Università
Ca'Foscari
Venezia

Master Degree Program in Computer Science

Final Thesis

Depth Completion

Mixing classical and learning-based methods for Depth Completion

Supervisor:
Asst. Prof. Filippo Bergamasco

Assistant Supervisor:
Mara Pistellato

Graduand:
Mengistu Shambel
882538

Academic Year:
2021/2022

ACKNOWLEDGEMENTS

First and foremost, I would like to praise and thank God, the Almighty, who has granted me countless blessing, support, and strength when I was passing through several hill and valleys during the whole period of my study.

I would like to express my deepest appreciation to my supervisor **professor Filippo Bergamasco** who has given me all his precious time to do this thesis. You relentlessly helped me in all the time that I needed help. I have benefited a lot from the extensive knowledge you have and the practical suggestion you shared me.

I must aslo thank **Mara Pistellato**, a post-doctoral student, who were on my side sharing her experience and knowledge since the beginning of the thesis work. She was always there when I asked her for any question.

I must not fail to sincerely appreciate all my families for their prayer, advice, and financial support.

I am also thankful to all the staff members of the Department of Computer Science for their full cooperation and help.

Abstract

Dense depth maps are important and have many applications in different computer vision tasks. To mention some of them, in autonomous driving, LIDAR sensors acquire a depth information to understand the environment around. Stereo vision, semantics segmentation, and optical flow are other tasks that uses a depth information. Depth completion aims to recover a dense depth map given sparse depth samples and optional additional data as input. While some methods take only sparse data as input, Others consider the corresponding RGB image as guidance to get a better dense depth representation. With the rise of data driven neural networks, most computer vision researchers moved away from classical methods and exploited the power of Convolutional Neural Network (CNN) for recovering accurate and dense depths. Some classical handcrafted methods also provide a commensurate result as that of modern deep neural network methods with a small computational time and computing resource requirement.

In this paper we have designed a depth completion algorithm that mixes a classical method with a modern learning-based method. The proposed method combines the two approaches to take advantage of the two methods by giving a more dense depth and can be trained from end-to-end. We evaluate our algorithm on the challenging KITTI depth completion benchmark dataset. Finally, we make a comparative analysis with some existing state of the art classical and modern depth completion approaches.

Keywords:

Depth Completion, Convolutional Neural Network, Sparsity Invariant CNN, depth up-sampling and down-sampling, Multi-scale Encoder-Decoder Network

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN Convolutional Neural Network

ANN Artificial Neural Network

HMS-Net Hierarchical Multi-Scale Network

ADM Adam Optimizer & Adaptive Learning Rate

IDW Inverse Distance Weighted

MSE Mean Square Error

IMSE Inverse Mean Square Error

MAE Mean Absolute Error

RMSE Root Mean Square Error

IRMSE Iverse Root Mean Square Error

LiDAR Light Detection and Ranging

STD Standard Deviation

Chapter 1

INTRODUCTION

1.1 Introduction

Depth Completion has become an important problem in recent years with the rapid growth of Computer Vision Application, such as autonomous driving, optical flow, stereo vision, and un-manned aerial vehicle control. To obtain a reliable depth completion, information from different sensors are used. e.g, Light Detection and Ranging (LiDAR) sensors and RGB cameras. Depth sensors such as LiDAR sensors, produce accurate depth measurements with high frequency. However, the depth data obtained is very sparse because of several reasons, for example because of hardware limitations. To solve such problems, there have been a lot of works to get a dense depth data given a sparse information. This process of getting a dense depth map from a sparse depth map is called Depth Completion. Early depth completion problems[1] were based on basic image processing operations that does not rely on large data and high computing power.

Modern depth completion algorithms are neural network based[2][3] , and highly dependant on large amount of data and usually requires high computing resources. Different kinds of neural network based methods have been proposed in the literature for depth completion problems. They range from approaches taking sparse depth input only to guided approaches which uses an additional RGB image.

This paper aims to show mixing some classical and learning based methods could give a better result. In particular, we have used a learning-based Sparsity Invariant Convolutional Neural Network (CNN)[3], and HMS-Net[2] to mix with classical approach which uses IDW. We showed that mixing the two approaches would allow us to take advantage of the two methods.

1.1.1 Depth Completion

Depth completion starts from a sparse set of known depth values and estimates the unknown depths for the remaining image pixels. The given sparse depth input is acquired/captured by different sensors, such as a LiDAR (laser scanner) sensor, and RGB camera.

Depth completion can refer to a range of related problems with different input modalities. e.g. exemplar-based depth inpainting, object-aware interpolation, and Nearest Neighbour Interpolation with CNN. However, depth completion can also refer to the problem of generating dense depth information from a scene when only a sparse representation of the scene depth is available. In this way we learn the structural composition of the scene.

Depth completion has got a particular interest in many applications, including stereo vision, optical flow, autonomous vehicles, and 3D reconstruction from sparse LiDAR data.
Depth completion can be categorized into different groups based on the principles they used. Based on input modalities they can be grouped into guided[2] and non-guided[3][1]. Guided depth completion problems are those that use RGB image as a guidance, together with the sparse input depth, where as non-guided approaches takes the sparse input only. Based on whether they require training or not, they can also be divided as trainable[2][3][4] and non-trainable[1][5]. Non-trainable methods generate dense depth maps from sparse inputs based on hand-crafted rules, but trainable methods are mainly based on deep neural networks. In this paper we combine a non-guided trainable methods called sparsity invariant CNN, and HMS-Net with a non-trainable method called IDW.

1.1.1.1 Formulation of Depth Completion from a mathematical perspective

The problem of Depth Completion can be described as follows:
Given an image $\mathrm{I} \in \mathbf{R}^{\mathbf{M}\times\mathbf{N}}$ and a sparse depth map $\mathbf{D}_{sparse} \in \mathbf{R}^{\mathbf{M}\times\mathbf{N}}$ find $\hat{f}$ that approximates a true function f: $\in \mathbf{R}^{\mathbf{M}\times\mathbf{N}} \times \in \mathbf{R}^{\mathbf{M}\times\mathbf{N}} \rightarrow \in \mathbf{R}^{\mathbf{M}\times\mathbf{N}}$ where $f(I, \mathbf{D}_{sparse}) = \mathbf{D}_{dense}$. the problem can be formulated as:

$$\parallel \hat{f}(I, \mathbf{D}_{sparse}) - f(I, \mathbf{D}_{sparse}) \parallel^2 = 0 \tag{1.1}$$

Here $\mathbf{D}_{dense}$ is the output dense depth map , and has the same size as $\mathbf{I}$ and $\mathbf{D}_{sparse}$ with empty values replaced by their depth estimate. In the case of non-guided depth completion, the above formulation becomes independent of the image $\mathbf{I}$ as shown in figure 1.1.

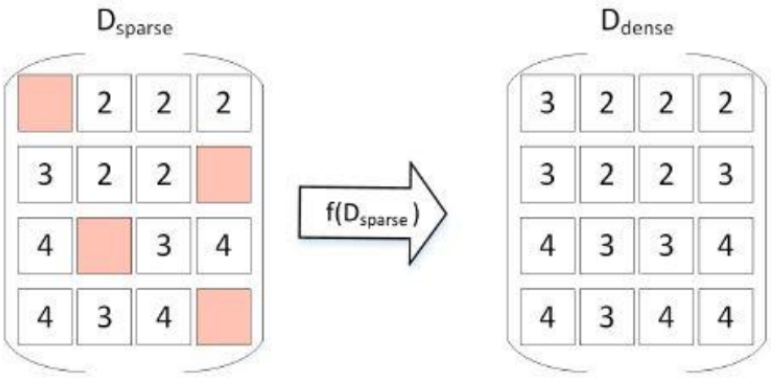

Figure 1.1: A toy example summarizing the problem formulation described in 1.1. Empty values are coloured in red, and filled by applying the function f to $\mathbb{D}_{sparse}$

1.1.2 Classical Depth Completion Approaches

With the rise of data driven neural networks, as a realization of universal function approximators, most research in computer vision has moved away from classical image processing that uses a hand crafted kernels to a learning based methods. Classical depth completions are Non-learning-based approaches that does not require training. Non-learning-based approaches generate dense depth maps from sparse inputs based on hand-crafted rules. Therefore, the outputs of these algorithms are generated based on assumed prior by humans. As a result, they are not robust enough to sensor noises and are usually specifically designed for certain datasets. But, with a well designed algorithm, some classical methods [1] [6] are capable of outperforming neural network based methods on the task of depth completion. In[1], a sequence of morphological operation are applied to get a dense depth map from a given sparse input.

An other classical method to apply for depth completion is to use a method of interpolation that estimates cell values by averaging the values of sample data points in the neighborhood of each processing cell, called IDW. The closer a point is to the center of the cell being estimated, the more influence, or weight, it has in the averaging process. Although all these classical methods are getting depreciated because of the rise of deep neural networks, they can be still used by mixing them with a modern depth completion problem. In this paper we have shown that by mixing IDW with a learnable neural network based methods, such as HMS-Net and Sparsity Invariant CNN, a better result can be gained.

1.1.3 Learning-based Depth Completion Approaches

For learning-based approaches, state-of-the-art methods are mainly based on deep neural networks. Various methods mainly utilize deep CNN [3] [7] [2], for generating dense depth maps from sparse inputs. Ma and Karaman [7] simply filled 0s to locations without depth inputs to create dense input maps, which might introduce ambiguity to very small depth values. Chodosh et al. [4] proposed to extract multi-level sparse codes from the inputs and used a 3-layer CNN for depth completion. However, those two methods used conventional convolution operations designed for dense inputs. Uhrig et al. [3] proposed sparsity-invariant convolution, which is specifically designed to process sparse maps and enables processing sparse inputs more effectively with CNN. However, the sparsity-invariant convolution in [3] only mimics the behavior of convolution operations in conventional dense CNNs. Its feature maps of later stages lose much spatial information and therefore cannot effectively integrate both low-level and high-level features for accurate depth completion. In this paper, in addition to Sparsity Invariant CNN, a multi-scale encoder-decoder network, HMS-Net[2], which adopts a series of sparsity-invariant convolutions with downsampling and upsampling to generate multi-scale feature maps and shortcut paths for effectively fusing multi-scale features has been adopted to overcome the problems mentioned above.

1.1.4 Classical Versus Modern Depth Completion Approaches

Many different approaches have been proposed for depth completion. These approaches range from handcrafted image processing methods[1] to end to end trainable ones [2][3]. The latter are very interesting as they do not require too much human involvement because of their data driven nature. However using deep learning approaches results in multiple consequences. First , learning based methods are mostly run on GPU power, and GPU power is always finite. GPUs are very power hungry, and deploying a GPU for each module to run is prohibitive. Second using deep neural network models require deep understanding of the model. lack of understanding of deep neural network models results in a poor design and sub-optimal result.
On certain problems, deep learning based approaches can still be outperformed by well designed classical image processing based algorithms[1]. These algorithms are non-learnable and usually can be run on CPU without requiring too much computation power. They relies on image processing operations, and do not have a problem of over-fitting which is a sensitive issue in learning based approaches. Especially in situation where there is no enough training data, it is wise to use this methods.

1.1.5 Application of Depth Completion

- Autonomous driving
- 3D object detection
- Optical flow estimation
- Semantic segmentation
- Stereo vision
- Security surveillance,
- 3D reconstruction from sparse LIDAR data.

1.2 Motivation

Modern learning-based Depth Completion problems such as [2][3] have been widely used to obtain a dense depth map from a sparse depth information. Although they require large amount of training data and high computing power, several state of the art algorithms outperformed classical approaches. The performance of most state of the art learning based methods depends on the sparsity level of the input depth. The depth maps are acquired through sensor devices such as RGB camera, and LiDAR which gives a sparse input. Some problems use RGB image as a guidance in addition to the sparse input to get a better result.
In this paper we have designed a problem that allows us to tackle the problem of sparse input by mixing a learning based depth completion method with a classical method. We have showed that mixing the results of the two approaches would give a better dense depth information.

1.3 Challenges

To complete the task of depth completion various depth sensors such as LiDAR have been invented to acquire depth information. However, current depth sensors are not able to obtain dense maps for outdoor scenes, which are essential in various applications, especially autonomous driving. Besides the depth map obtained by such sensors is highly sparse and irregularly spaced, and as a result depth completion from sparse depth maps and RGB images has attracted intensive attention. For example, in the KITTI dataset [8], there are only 5.9% pixels with depth information obtained by the Velodyne HDL-64e (64 layers) LiDAR in the whole image space.

Recent studies such as [3] [8], have demonstrated great advantages of deep Convolutional Neural Networks (CNNs) on depth completion. By extending the convolutional operation with sparsity-invariant, these deep methods can achieve way better performance than traditional methods. One of the challenges of depth completion problem in this paper was the sparsity of the input data. A properly designed algorithm is required to handle several levels of sparsity. The other challenge was there was no common methods to visualize the output depth maps.

1.4 Problem Definition

Depth completion aims to recover a dense depth map from the sparse depth data and an optional RGB image. The observed pixels provide a significant guidance for the recovery of the unobserved pixels' depth. Additional input mask is also used to keep track of the visibility state and make it available to the next layers of the network. In this paper we have designed a depth completion problem by mixing a learning-based method, such as HMS-Net and Sparsity Invariant CNN with a classical method, IDW. The mixing is performed by taking a pre-trained network in[2][3], and mixing it with IDW. The mixed result is then further passed through other convolutions layers.

1.5 Objectives

The objective of this paper is to investigate some learning-based and classical depth completion problems and to design a problem that mixes the two approaches. In particular we have adopted HMS-Net [2] and Sparsity Invariant CNN[3] to mix with IDW[5], and designed two depth completion problems , one is mixing HMS-Net with IDW, and the other is mixing Sparsity Invariant CNN with IDW.

1.6 Outline of the Thesis

The remaining part of the document is organized as follows,

Chapter 2 describes the classical depth completion approach used in this paper to design our depth completion problem. We have used a handcraft image processing method called Inverse Distance Weighted. The IDW is mixed with HMS-Net, and Sparsity Invariant CNN to design two kind of depth completion problems.

Chapter 3 is one of the main topic of this paper which gives a brief and detailed explanation about HMS-Net and Sparsity Invariant CNN. HMS-Net and Sparsity Invariant CNN are learning based depth completion methods.

Chapter 4 gives a brief overview of our depth completion problem design. We have designed two models. one is the mixture of Sparsity Invariant and IDW, and the other is designed by mixing HMS-Net and IDW. The two designs follow the same procedure except that their internal network is different.

Chapter 5 describes the experiment and results of the models. In this section we mentioned the parameters in the experimental setups , dataset, metrics, and the experimental results. The result of different algorithms are also analysed and their performance is compared using different metrics.

Chapter 6 is conclusion which summarizes the basic works done so far and the main talking points from the result we saw. It also includes personal recommendation for further work.

Chapter 2

Classical Depth Completion

2.1 Two-dimensional interpolation Functions

In many fields using empirical areal data, there arises a need for interpolating from irregularly-spaced data to produce a continuous surface. These irregularly spaced locations, hence referred to as "data points", may have diverse meanings: in meteorology, weather observation stations; in geography, surveyed locations; in city and regional planning, centers of data-collection zones; in biology, observation locations; in computer vision, depth information or values. It is assumed that a single unique number is assigned to each data point such as rain fall in meteorology, depth values in computer vision.

In order to display these data in some type of contour map or perspective view, to compare them with data for the same region based on other data points, or to analyze them for extremes, gradients, or other purposes, it is extremely useful, if not essential, to define a continuous function fitting the given values exactly. Interpolated values over a fine grid may then be evaluated.
In essence, it is desired to find a function that finds the value for interpolated point from from the given irregularly-spaced data points. It is assumed that a finite number N of triplets (x_i, y_i, z_i) are given, where, x_i and y_i are Cartesian coordinates of the corresponding data values. An interpolation function $z = f(x, y)$ to assign a value to any location $P(x, y)$ in the plane is sought.

2.2 Existing Approaches for Two-dimensional Interpolation

Although many solutions to related problems in two dimensional interpolation have been in long use, interpolation functions making an exact fit for irregularly spaced data are rare. When the data points already form a regular lattice, many solutions are possible. Among the most significant solutions for a rectangular grid are fitting a hyperbolic paraboloid to each four data points by double linear interpolation [9], or employing bi-cubic spline interpolation[10]. For a triangular lattice, fitting a plane to each three points is easy and effective. Downing, has developed a computer contouring program which interpolates intervening points from a square lattice, making possible planar interpolation over a triangular grid[11].
An other approach, easier to use and very elegant, would be to fit a polynomial or trigonometric function in two variables with enough coefficients so that it assumes exactly all the data values. Berezin, suggests a general formula for fitting a polynomial of degree $N-1$ in x and y to N data points[12]. Though it meets all the criteria, the computations became exceedingly long with large numbers of data points. To find an interpolated value given 1000 data points, for example, would require evaluating and multiplying 999 scalar products of two-dimensional vectors.

2.3 Inverse Distance Weighted Interpolation

IDW is a spatial interpolation approach that is used commonly to estimate an unsampled or unmeasured variable at any location in a study area [5]. IDW is a deterministic interpolation approach which considers the distance of an unsampled point towards a set of surrounding sampling points in weights determination stage. An initial inverse distance function was tested; in it the value at any point $\mathbf{P}$ in the plane was a weighted average of the values at the data points $\mathbf{D}_i$.

2.3.1 Methodology

To estimate an unsampled point using IDW method, a number of sampling points are required. Based on the distance between an unsampled point to each sampling point that involves in the calculation process, the weight will be determined with Shepard's rule [5] as in equation 2.1, where d_i is the Euclidean distance from the unsampled to the sampling point which can be calculated using equation 2.2. Lastly, $\mathbf{P}$ stands for power. The default value in shepard rule is 2.

Let z_i be the value at data point $\mathbf{D}_i$ and $d[P, D_i]$ be the Cartesian distance between P and D. Where the reference point P is understood, $d[P, Di]$ will be shortened to d_i. The interpolated value at $\mathbf{P}$ using this first interpolation function is:

$$f_1(P) = \begin{cases} \dfrac{\sum_{i=1}^{N} d_i^{-u} z_i}{\sum_{i=1}^{N} d_i^{-u}}, & \text{if } d_i \neq 0 \text{ for all } \mathbf{D}_i \\ 0, & \text{if } d_i = 0 \text{ for some } \mathbf{D}_i \end{cases} \tag{2.1}$$

The distance d_i is computed using the following equation:

$$d_i = \sqrt{(x_i - x_z)^2 + (y_i - y_z)^2} \tag{2.2}$$

In equation 2.1, notice that as $\mathbf{P}$ approaches a data point $\mathbf{D}_i, d_i \to 0$, and the i$^{\text{th}}$ terms in the numerator and denominator exceed all bounds while the other terms remain bounded. Therefore $\lim_{P \to \mathbf{D}_i} f_1(P) = z_i$ as desired, and the function $f_1(P)$ is continuous

In this paper, we used IDW to compute a set of kernels for each power values used. we have used seven power values between $2 and 4$, and a kernel of size 31×31 is computed using equation 2.2. The given sparse depth map is used as input, and seven feature depth maps are computed by applying conventional convolution using seven different kernels. A detailed description on how we mix the outputs of IDW with HMS-Net and Sparsity Invariant CNN, will be given in 4 section.

2.3.2 Choice of Exponent

As mentioned above, weights are proportional to the inverse of the distance between the data point and the prediction location raised to the power value u. As a result, as the distance increases, the weights decrease rapidly. The rate at which the weights decrease is dependent on the value of u. If $u = 0$, there is no decrease with distance, and because each weight $\frac{1}{d_i^u}$ is the same, the prediction will be the mean of all the data values in the search neighborhood. As u increases, the weights for distant points decrease rapidly.

If the u value is very high, only the immediate surrounding points will influence the prediction. Figure 2.1 shows the relationship between distance and weight as a function of power value.

Empirical tests showed that higher exponents $(u > 0)$ tend to make the surface relatively flat near all data points, with very steep gradients over small intervals between data points. Lower exponents produce a surface relatively flat, with short blips to attain the proper values at data points. An exponent of $u = 2$ not only gives seemingly satisfactory empirical results for purposes of general surface mapping and description, but also presents the easiest calculation. In this paper we have used a set of power values between 2 and 4, and we let the network to choose based on some probability estimates.

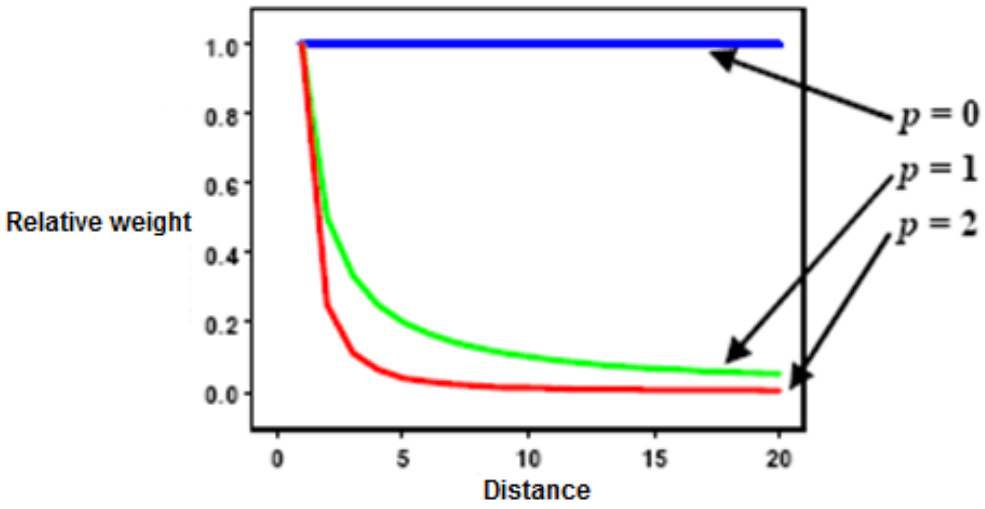

Figure 2.1: Decrease of weight with distance illustration

2.3.3 The search neighborhood

There are two approaches that can be chosen in selecting sampling point around an estimated point location [5]. The first approach is based on an arbitrary distance criterion, all data points within some radius r of the point P, and the second one is using an arbitrary number criterion, the nearest n data points, could be employed. The former choice, though computationally easier, allowed the possibilities that no data points, or an unmanageable large number of data points, might be found within the radius r.

The latter choice required a more detailed searching and ranking procedure for data points, and presupposed that a single number of interpolating points was best, regardless of the relative location and spacing of the points. In our paper, in order that the interpolation work reasonably, the data points were gridded, a set of data points was chosen in a square patch, and the value for the interpolated point is computed using the conventional convolution operation.

2.3.4 Shortcomings of Pure Inverse-Distance Weighting

Though the above method is sufficiently simple, and effective for our depth completion problem design , it did have several shortcomings.

1. When the number of data points is large, the calculation of $z = f_1(P)$ becomes proportionately longer. Eventually the method will become inefficient or impractical.

2. Only the distances to P from the data points $\mathbf{D}_i$, and not the direction, are considered. Therefore the following two configurations of co-linear points, for example, would yield identical interpolated value.

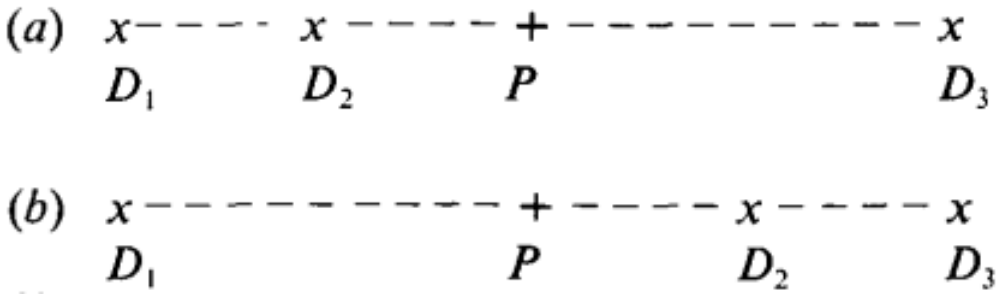

Figure 2.2: Different configuration of colinear points

3. Computational error becomes significant in the neighborhood of points $\mathbf{D}_i$, as the predominant term results from the difference of two almost equal numbers.

2.3.5 Chapter Summary

In many fields using empirical areal data there arises a need for interpolating from irregularly-spaced data to produce a continuous surface. These irregularly spaced locations, are referred to as data points. There are different kinds of function that finds the value for interpolated point from the given irregularly-spaced data points. Among the most significant solutions are, double linear interpolation, bi cubic spline interpolation, fitting a polynomial or trigonometric functions, and inverse distance weighted (IDW) interpolation. In this paper we have adopted IDW[5] as a classical methods of interpolating missing depth values, and this result will be mixed with learning based depth completion methods, such as Sparsity Invariant CNN and HMS-Net.

IDW is a deterministic interpolation approach which considers the distance of an unsampled point towards a set of surrounding sampling points in weights determination stage. Based on the distance between an unsampled point to each sampling point that involves in the calculation process, the weight will be determined with Shepard's rule. In IDW there are some parameters that has to be selected carefully, such as, power value, and the number of neighboring points considered in the calculation of interpolated point value.

Chapter 3

Learning-based Depth Completion

3.1 Introduction

The popularity of using depth sensing devices for dense depth prediction is constantly increasing in recent years. Though, some of these sensors give accurate depth measurement, e.g. LiDAR sensors, they suffer from the deficiency in producing a dense depth map. For example, in LiDAR sensors, only about 5% of pixels in generated depth map are observable [13]. Such a sparse output is insufficient for real world applications such as 3D reconstruction and mapping. Thus, it is of great value to research on the topic of sparse depth completion.

Currently, many methods have been proposed for this problem. These works can be categorized into two groups: depth-only completion and image guided completion. The depth-only methods use sparse depth map as input and infer the missing depth values to generate a full resolution depth map[2] [3]. However, as the observable depth points are unevenly scattered on the depth map, it is difficult to estimate the depth of non-observable area where the available measurements are too sparse. Hence, the accuracy of the depth-only method is still insufficient for practical usage. The image guided methods take sparse depth along with the synchronized RGB image as input [2]. The rich semantic information from image domain provides strong depth cues to assist the completion. Thus, these methods achieves better performance than depth-only approaches. In general, the image guided completion focuses on two topics: how to process sparse input and how to fuse data from different modalities. As the Convolutional Neural Network (CNN) is a powerful tool to perceive image data , the state-of-the-art depth completion works are mainly based on CNN.

Normally, the invalid pixels on a depth map are encoded by a default value, i.e., zero. This setting makes the traditional convolution operation difficult to distinguish between the valid and non-valid depth points. Besides, when the input to the network is sparse and irregular (e.g., when only 10% of the pixels carry information), it becomes less clear how the convolution operation should be defined as for each filter location the number and placement of the inputs varies. To solve these problems, different depth completion tasks that use a Sparsity Invariant Convolution [3] [2] are proposed, which utilizes an observation mask to instruct convolution calculation and normalize output features. However, the sparsity-invariant convolution in [3] only mimics the behavior of convolution operations in conventional dense CNNs. Its feature maps of later stages lose much spatial information and therefore cannot effectively integrate both low-level and high-level features for accurate depth completion(see 3.1 for illustration).

On the other hand, there exist effective multi-scale encoder-decoder network structures for dense pixel-wise classification tasks (see 3.2), such as [14], and Full Resolution Residual Network [15]. Direct integration of the sparsity invariant convolution[3] into the multi-scale structures is infeasible, as those structures also require other operations for multi-scale feature fusion, such as sparsity-invariant feature upsampling, average, and concatenation.In this paper we have adopted both Sparsity Invariant CNN, and a multi-scale encoder-decoder network, HMS-Net, which adopts a series of sparsity-invariant convolutions with downsampling and upsampling to generate multi-scale feature maps and shortcut paths for effectively fusing multi-scale features.

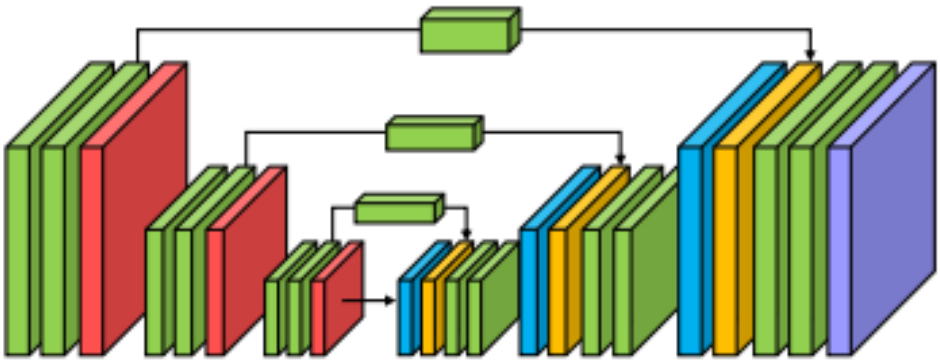

Figure 3.2: Sparsity Invariant Encoder-Decoder Network

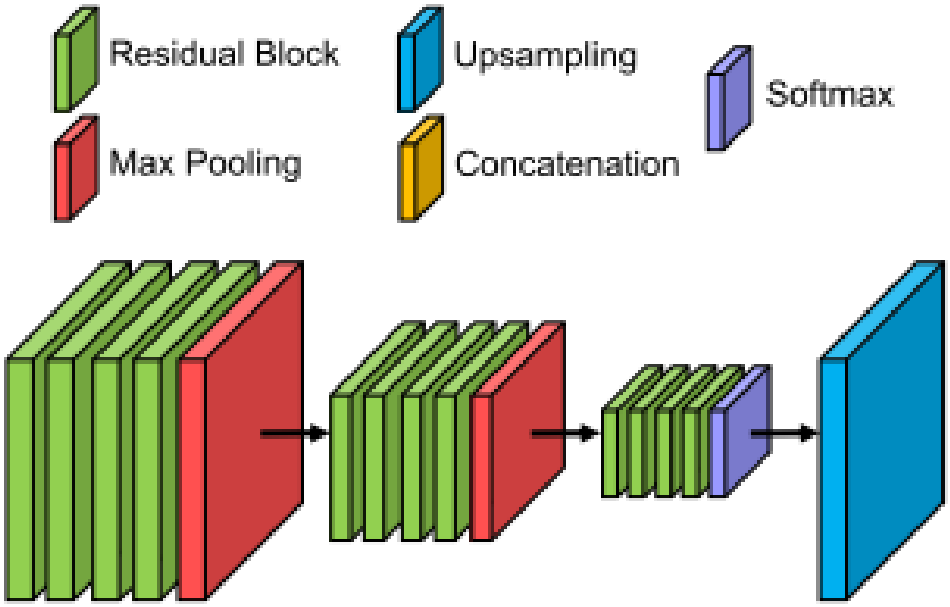

Figure 3.1: CNN-with Sparsity Invariant Convolution only

3.1.1 Depth Completion Related Tasks

(A) Depth Completion
Depth completion is an active research area with a large number of applications. According to the sparsity of the inputs, current methods could be divided into two categories: sparse depth completion and depth enhancement. The former methods as the name indicates, aim at recovering dense depth map from a sparse depth input, while the later methods work on conventional RGB-D depth data(RGB image and corresponding sparse depth map) and focus on filling irregular and relatively small holes in input dense depth maps. In addition, if the input depth maps are regularly sampled, the depth completion task could be regarded as a depth upsampling (also known as depth super-resolution) task. In other words, depth upsampling is a specific task of depth completion. According to whether RGB information is utilized, depth upsampling methods could be divided into two categories: guided depth upsampling and depth upsampling without guidance.

(B) Sparse Depth Completion
In sparse depth completion, to handle sparse inputs and sparse intermediate feature maps, Uhrig et al. [3] proposed sparsity-invariant convolution to replace the conventional convolution in convolution neural networks (CNN). The converted sparsity-invariant CNN keeps track of sparsity masks at each

layer and is able to estimate dense depth maps from sparse inputs. There also exist works utilizing RGB images as additional information to achieve better depth completion. Huang et al. [2] combined both an RGB image and and a sparse depth map to get a dense depth maps. In this paper we have adopted a similar algorithm, but it only uses a sparse depth input.

(C) Depth Enhancement
The inputs of depth enhancement or depth hole-filling methods are usually dense depth maps with irregular and rare small holes. The input depth maps are usually captured with RGB images. Matyunin et al. [16] used the depth from the neighborhoods of the hole regions to fill the holes, according to the similarity of RGB pixels.

(D) Guided Depth Upsampling
Depth upsampling methods take low-resolution depth maps as inputs and output highresolution ones. As the guidance signals, the provided RGB images bring valuable information (e.g., edges) for upsampling. Li et al. [17] proposed a CNN to extract features from the low-resolution depth map and the guidance image to merge their information for estimating the upsampled depth map.

(E) Depth Upsampling Without Guidance
Depth upsampling could also be achieved without the assistance of corresponding RGB images. they are are closely related to those for single image superresolution.

3.2 Sparsity Invariant Convolution

In this section we describe about one of the learning based depth completion problem that we adopted in our design of depth completion problem. Jonas Uhrig et. al[3] proposed this algorithm to overcome the problem of traditional convolutional networks when applied to sparse data. they propose a simple yet effective sparse convolution layer which explicitly considers the location of missing data during the convolution operation.
The naive approach to sparse depth completion problem is to assign a default value to all non-informative sites[18] [19]. Unfortunately, this approach leads to suboptimal results as the learned filters must be invariant to all possible patterns of activation whose number grows exponentially with the filter size. By introducing a novel sparse convolutional layer which weighs the elements of the convolution kernel according to the validity of the input , Sparsity Invariant CNN can handle

large levels of sparsity without significantly compromising accuracy. In addition, there is a second stream that carries information about the validity of pixels to subsequent layers of the network.

3.2.1 Conventional Convolution

The output of a standard Convolutional layer in a CNN is computed as follow:
Let f denote a mapping from input domain $\mathbf{X}$ (e.g., intensity, depth) to output domain $\mathbf{Y}$ (e.g., depth, semantics), implemented via a Convolutional neural network. Given an partially observed input(sparse input) $x = x_{u,v} \in X$, and corresponding binary mask $m_x(u, v)$, the convolution output is given by:

$$f_{u,v}(x) = \sum_{i,j=-k}^{k} x(u+i, v+j)w(i, j) + b \tag{3.1}$$

with kernel size $2k + 1$, weight w and bias b. If the input comprises multiple features, $x_{u,v}$ and $w_{i,j}$ represent vectors whose length depends on the number of input channels. As it can be shown from 3.1, conventional convolution does not handle invalid inputs.

3.2.2 Naive Approach

There are two naive ways to deal with unobserved inputs. The first approach is encoding invalid inputs $x_{u,v}$ using a default value, e.g., zero. This approach is very difficult to apply as network must learn to distinguish between observed inputs and those being invalid. this causes the number of possible binary patterns to grow exponentially with the kernel size. Alternatively, $\mathbf{M}$ can be used as an additional input to the network in the hope that the network learns the correspondence between the observation mask and the inputs. Unfortunately, both variants struggle to learn robust representations from sparse inputs.

3.2.3 Sparse Convolution

Sparsity Invariant Convolution overcomes the above problems using binary observation mask by evaluating only observed pixels and normalizing the output appropriately. The convolution output is given by:

$$z(u,v) = \frac{\sum_{i,j=-k}^{k} m_x(u+i,v+j)x(u+i,v+j)w(i,j)}{\sum_{i,j=-k}^{k} m_x(u+i,v+j) + \epsilon} + b \tag{3.2}$$

Here a small value, ϵ is added to the denominator to avoid division by zero at locations where $x(u+i, v+j)$ is invalid. The sparse feature map x and the observation mask m_x have the same spatial size $H \times W$. For each location (u, v), the convolution produces output $z(u, v)$. At each spatial location (u, v), the binary sparsity mask $m_x(u, v)$ records whether there are input features at this location, i.e., 1 for existing features and 0 otherwise. The convolution kernel w is of size $(2k+1) \times (2k+1)$, and b represents a learnable bias vector. Note that the kernel weights w and bias vector b are learned via back-propagation, while the sparsity mask m_x is specified by the previous layer and is not trainable. The observation mask at each layer is computed using a max pooling operation as follow:

$$m_x(u,v) = max_{i,j=-k,\ldots,k}(o_{u+i,v+j}) \tag{3.3}$$

which evaluates to 1 if at least one observed variable is visible to the filter and 0 otherwise.

The primary motivation behind sparse convolution operation is to render the filter output invariant to the actual number of observed inputs which varies significantly between filter locations due to the sparse and irregular input. Note that in contrast to other techniques [20] [21], which artificially upsample the input (e.g., via interpolation), This approach operates directly on the input and does't introduce additional distractors. The operation of a Sparse convolution is illustrated in Fig.3.3. In the figure , $\odot$ denotes elementwise multiplication, $\star$ convolution, $1/x$ inversion and "max pool" the max pooling operation. The input feature can be single channel or multi-channel.

3.2.4 Sparsity Invariant Convolution Architecture

The architecture of sparsity invariant network is shown in figure 3.4. The input to the network is a sparse depth map (yellow) and a binary observation mask (red). It passes through several sparse convolution layers (dashed) with decreasing kernel sizes from 11×11 to 3×3. all the convolution layers produce a 16 channel depth features.

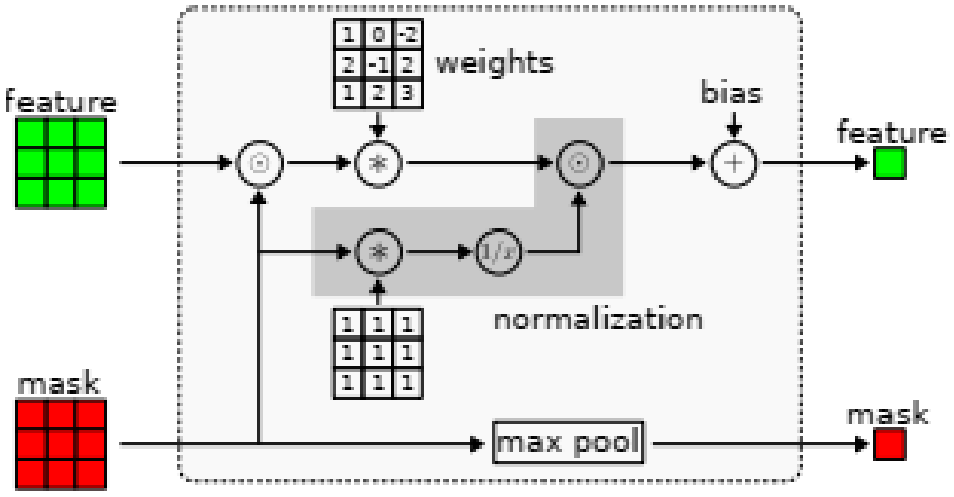

Figure 3.3: Sparse Convolution

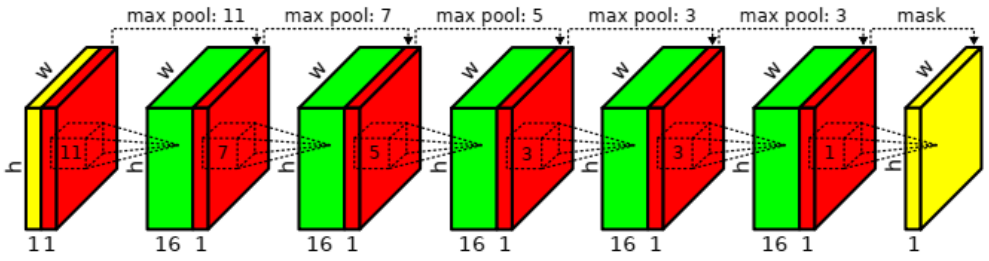

Figure 3.4: Network Architecture

3.3 Hierarchical Multi-scale Encoder Decoder Network

Neural networks that utilize multi-scale feature maps for pixelwise prediction (e.g. semantic segmentation) were widely investigated. Combining both low-level and high-level features was proven to be crucial for making accurate pixelwise prediction. However the Sparsity Invariant CNN proposed in [3] can not be used with HMS-Net as the HMS-Net can not be converted to handle sparse inputs. This is because there exist many operations that do not support sparse feature maps. By adding some operations, called sparsity invariant operation, sparsity invariant convolution proposed in[3] can be adopted with HMS-Net. The HMS-Net network structure consists of the three novel sparsity invariant operations, and two basic building blocks called two scale block and three scale block (see figure 3.5 for detail illustrations).

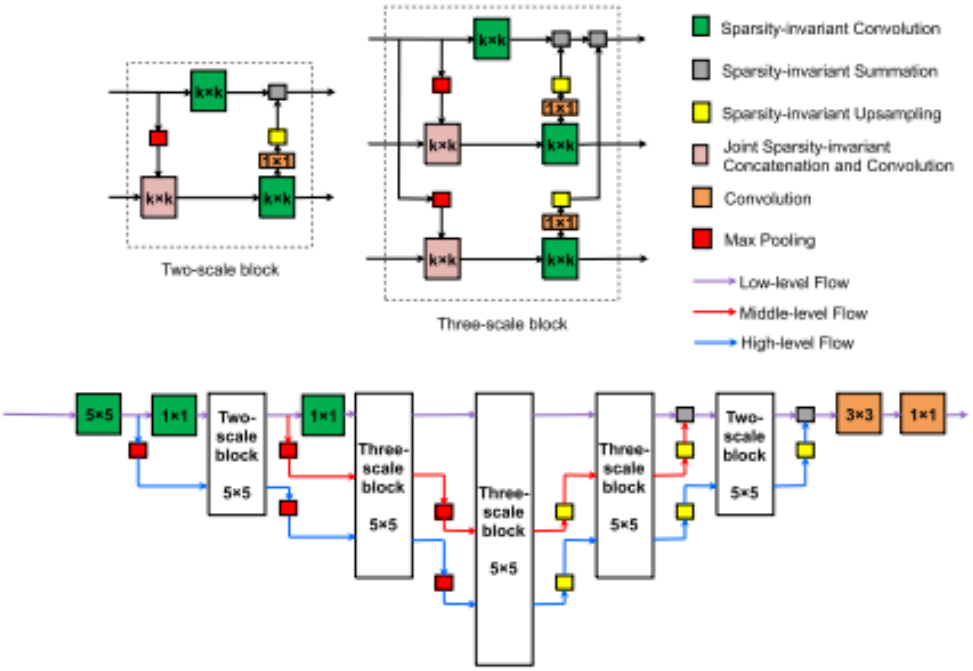

Figure 3.5: Illustration of adopted multi-scale encoder-decoder network structure for depth completion based on the sparsity-invariant operations

3.3.1 Basic Building Blocks of HMS-Net

The two basic building blocks of HMS-Net, a two-scale block and a three-scale block, consisting of a sequence of sparsity-invariant operations. The two-scale block has an upper path that non-linearly transforms the full-resolution low-level features by a $K \times K$ sparsity-invariant convolution. The lower path takes down-sampled low-level features as inputs for learning higher-level features with another $K \times K$ convolution. The value of K is set to 5. The resulting higher-level features are then upsampled and added back to the full-resolution low level features. Compared with the two-scale block, the three scale block fuses features from two higher levels into the upper low-level feature path to utilize more auxiliary global information. In this way, the full-resolution low-level features are effectively fused with higher-level information and are non-linearly transformed multiple times to learn more complex prediction functions. All feature maps in the network are of 16 channels regardless of scales.

3.3.2 Sparsity Invariant Operations In HMS-Net

The sparsity-invariant convolution successfully converts conventional convolution to handle sparse input features and is able to stack multiple stages for learning highly non-linear functions. However, only modifying convolution operations is not enough if one tries to utilize state-of-the-art multi-scale encoder-decoder structure for pixelwise prediction. The multi-scale encoder-decoder networks cannot be directly converted to handle sparse inputs.For this reason, Huang et.al [2] proposed the three novel sparsity invariant operation.

The three novel Sparsity Invariant Operations, namely, Sparsity Invariant upsampling, Sparsity Invariant Averaging, and Joint Sparsity Invariant Concatenation and Convolution are adopted to allow effectively handling sparse feature maps across the entire encoder-decoder network. As sparsity invariant convolution, a single-channel sparsity masks to track the validity of feature map locations. The sparsity masks could be used to guide and regularize the calculation of the operations.

3.3.2.1 Sparsity-Invariant Bilinear Upsampling

In multi-scale encoder-decoder network, Sparsity invariant upsampling is one of the basic operations in the decoder part. Upsampling is an operation used to increase the number of samples which are small in number or size by some factor. There are different upsampling operations, such as, linear, bilinear, and cubic.

Here we used the sparsity-invariant bilinear upsampling operation. Let x and m_x denote the input sparse feature map and the corresponding input sparsity mask of size $H \times W$. The operation generates the output feature map z and its corresponding sparsity mask m_z of size $2H \times 2W$. Let $\mathbf{F}$ represents the conventional bilinear upsampling operator, which bilinearly upsamples the input feature map or mask by two times. The proposed sparsityinvariant bilinear upsampling can be formulated as

$$z = \frac{F(m_x \bigodot x)}{F(m_x) + \epsilon}, \tag{3.4}$$

$$m_z = 1[F(m_x) \neq 0] \tag{3.5}$$

Where $\cdot$ denotes the spatial elementwise multiplication, ϵ is a very small number to avoid division by zero, and $1[\cdot]$ denotes the indicator function, i.e. $1[\text{ true }] = 1$ and $1[\text{ false }] = 0$. The mask m_z is computed using a max pooling operation given in equation 3.3. The proposed sparsity-invariant bilinear upsampling operation is illustrated in figure 3.4.

The sparsity invariant upsampling operation takes a single sparse depth x and the corresponding mask m_x. As shown in equation 3.4, first input x is elementwise multiplied with its corresponding mask m_x to give $x \bigodot m_x$, then a bilinear upsampling operator F is applied on both the resulting feature map, $x \bigodot m_x$ and the binary mask m_x. The upsampled sparse features $F(m_x \bigodot x)$ are then normalized at each location according to the upsampled sparsity mask values $F(m_x)$. The final sparsity mask m_z is obtained by putting 1 on the non-zero locations of the upsampled sparsity mask $F(m_x)$, otherwsie zero. practically this operation is performed by applying max pooling operation on the upsampled mask.

An other operation used in the HMS-Net is called sparsity-invariant max-pooling or downsampling. It is used to downsample the feature maps that we upsampled using sparsity invariant upsampling operation. It could be calculated the same as Eqs. 3.4 and 3.5 by replacing the upsampling function F with maxpooling or downsampling operators.

3.3.2.2 Sparsity-Invariant Average or Summation

Pixelwise average of two feature maps of the same spatial sizes is needed for fusing features from different levels without increasing the output channels. For average of sparse input feature maps, however, specifically designed average operation is required. The sparsity-invariant average operation, takes two input sparse

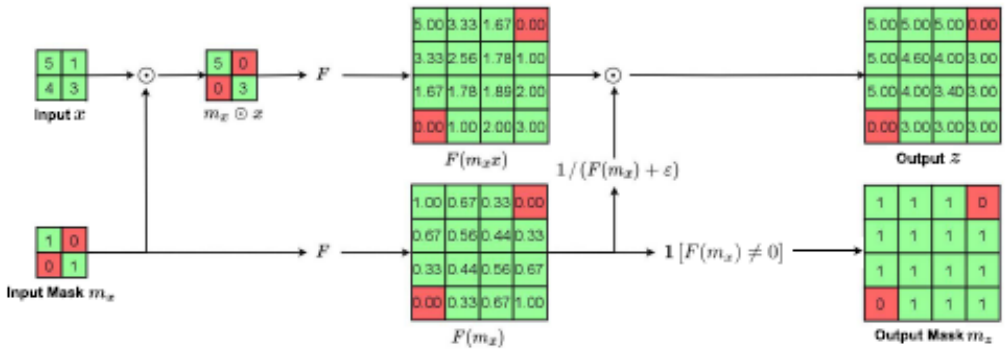

Figure 3.6: Illustration of the proposed sparsity-invariant upsampling operation. F stands for Bilinear Upsampling

feature maps, x and y, with their corresponding sparsity masks, m_x and m_y, as inputs. sparsity-invariant average, which takes two input sparse feature maps, x and y, with their corresponding sparsity masks, m_x and m_y, as inputs. A sparse feature map z, and corresponding mask m_z is generated by fusing the two sparse inputs. The main difference between sparsity invariant upsampling or downsampling, and sparsity-Invariant Average is the number of sparse features taken as input. The sparsity-invariant average can be computed as follow:

$$z = \frac{(m_x \bigodot x) + (m_y \bigodot y)}{(m_x) + m_y \epsilon}, \tag{3.6}$$

$$m_z = m_x \bigvee m_y \tag{3.7}$$

where $\bigvee$ denotes elementwise alternation i.e., logical 'or' function, $\bigodot$ represents elementwise multiplication, and ϵ is a very small number to avoid division by zero.

Figure 3.7 illustrates the sparsity invariant average operation. On one side the two sparse feature inputs, x and y are elementwise multiplied with their corresponding masks, m_x and m_y, and the output features are pixelwise added to generate a feature, $(m_x \bigodot x) + (m_y \bigodot y)$. The output feature $(m_x \bigodot x) + (m_y \bigodot y)$ is normalized by $(m_x + m_y)$, which is obtained by pixelwise addition the two masks. On the other side, a logical 'or' operation is performed on the masks $m_x, and m_y$ to generate the mask, m_z of the output feature z. For the output sparsity mask m_z, at each location, if the location is valid for either of the input feature maps, the mask is set to 1 for this location. At each location of the output feature map, the output feature vector is the mean of two input feature vectors if both input maps

are valid at this location. If only one input feature is valid at a location, the valid single input feature vector is directly copied to the same location of the output feature maps.

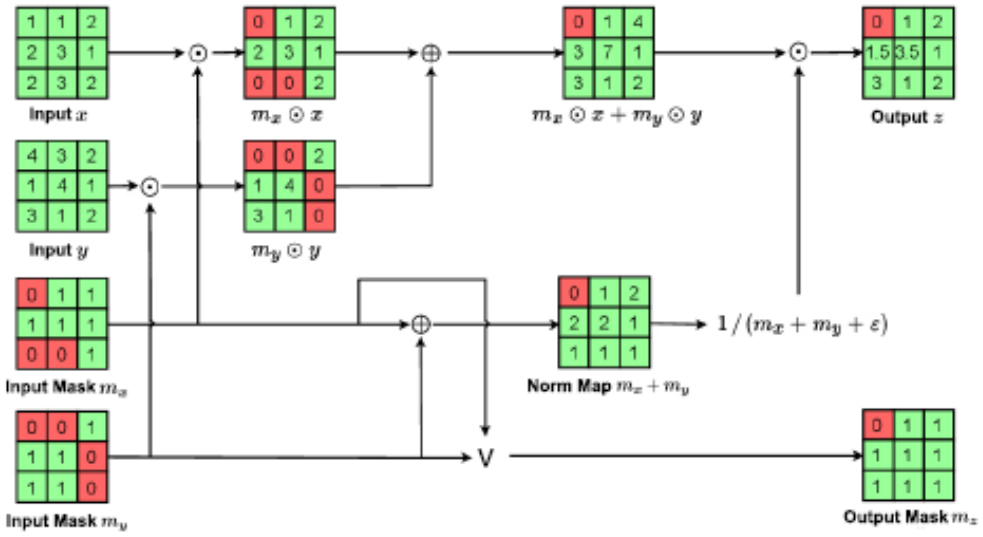

Figure 3.7: Illustration of the proposed sparsity-invariant average.

3.3.2.3 Joint Sparsity-Invariant Concatenation and Convolution

Another commonly used approach of fusing two feature maps with the same spatial size is to concatenate features along the channel dimension. However, different from aforementioned other operations, concatenation would introduce sparsity in both the spatial dimension $(H \times W)$ and the feature dimension (C), and the latter actually prevents us from simply extending the idea of sparsity-invariant average in the previous subsection. If we have two feature maps with shape $C_1 \times H \times W$ and $C_2 \times H \times W$, concatenating them in to one results in a shape of $(C_1 + C_2) \times H \times W$ as illustrated in figure 3.8. The concatenation is further followed by 1×1 convolution layer. As we know convolution performs filtering on a location by extracting one local feature vector with length C and summing all entries up into a number with learnable weights. Then, the convolution kernels iterate over the whole feature map, treating every location equally.

In sparsity-invariant convolution, the feature vector for a certain location only have two possible sparsity patterns-the whole vector of length C is valid, or all

the entries of this vector are zeros. Note that the contribution of the latter situation to the output as well as the gradient of kernels is zero, so it does not affect the training. Therefore, it is enough for us to use one set of convolution kernels, equally for every valid location.

However, when we are convolving on the feature maps after concatenation, we have four different types of vectors or sparsity patterns for each location: the first C_1 feature channels of the vector is valid while the latter C_2 feature channels are not; or C_2 is valid while C_1 is not, or both of them are valid/invalid. Therefore, we need three different sets of kernels to tackle these four different sparsity patterns. In other words, to effectively handle different scenarios at different locations of the concatenated feature maps, we propose to use an adaptive-kernel version convolution to solve the difficulty and combine it with concatenation together.

Another advantage of combining them is that all convolution kernels would generate outputs with the same spatial sparsity patterns. Therefore the output mask is still of single channel, which is computationally efficient and reduces the model complexity significantly. The joint sparsity-invariant concatenation and convolution is described and explained formally as following and illustrated in Fig.3.9.

Given the two input sparse feature maps x and y with their sparsity masks m_x and m_y, the proposed joint concatenation and convolution operation is formulated as

$$z = [x; y] \star k_a, \tag{3.8}$$

$$m_z = m_x \bigvee m_y \tag{3.9}$$

where $[;]$ denotes the concatenation of two feature maps along the channel dimension, and $\star$ denotes the conventional convolution operation. Note that the output sparsity mask is calculated exactly the same as that in sparsity-invariant average. The key of the proposed operation is a 1×1 convolution with an adaptive convolution kernel k_a that handles three different scenarios of concatenating sparse feature maps, which is formulated as

$$k_a(u,v) = \begin{cases} k_a^{(1)} m_x(u,v) = 1, m_y(u,v) = 0; \\ k_a^{(2)} m_x(u,v) = 0, m_y(u,v) = 1; \\ k_a^{(3)} m_x(u,v) = 1, m_y(u,v) = 1, \end{cases} \tag{3.10}$$

where $k_a(u,v)$ are the 1×1 adaptive convolution kernel at location (u,v) of the concatenated feature maps $[x;y]$. $k_a^{(1)}$v $k_a^{(2)}$, and $k_a^{(3)}$) are the three sets of learn-

able convolution weights for the three different feature concatenation scenarios: at each location (u, v), either both input feature vectors are valid (i.e., $m_x(u, v) = 1$ and $m_y(u, v) = 1$), or only one of the input feature vectors is valid (i.e., either $m_x(u, v) = 1$ or $m_y(u, v) = 1$).The key reason for using different sets of kernel weights instead of the same set of convolution weights, as we briefly introduced before, is to avoid invalid input features in the concatenated feature maps into feature learning process. For example, if the current 1×1 convolution kernel is on the location (u, v) and find that the first mask here $m_x(u, v) = 1$ and the second mask $m_y(u, v) = 0$), we choose kernel $k_a^{(1)}$. thus, this kernel is used both in forward pass and backward propagation. In this case, we know the second chunk of the feature vector, of which the length is fixed, is always zero. And because it's consistently processed by the first kernel, this kernel would naturally learn how to adapt to this pattern.
In other words, by adopting the proposed adaptive convolution kernel k_a, the three sets of kernel weights $k_a^{(1)}$v $k_a^{(2)}$, and $k_a^{(3)}$) are able to handle different sparse feature concatenation scenarios. With joint training, the different kernels are learned to best adapt each other to generate appropriate feature representations for further processing. In this way, the sparse feature maps could be effectively fused with proposed concatenation.

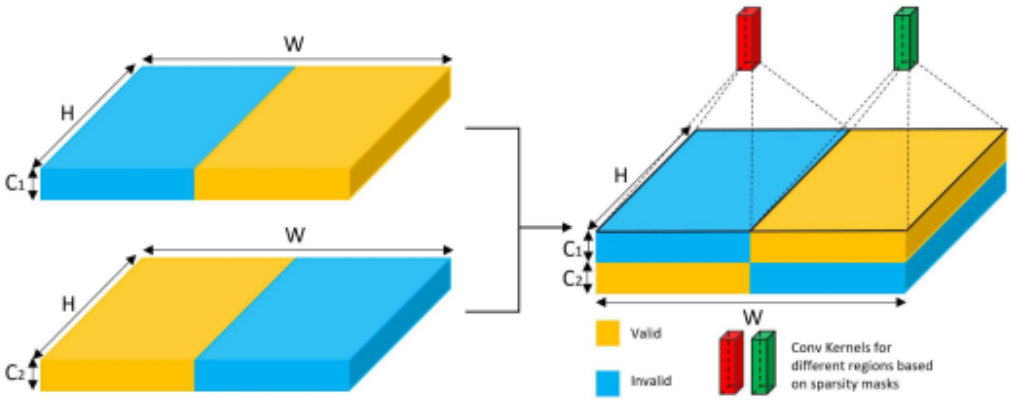

Figure 3.8: Sparsity patterns vary from regions, thus we need several different kernels to deal with out feature maps after concatenation.

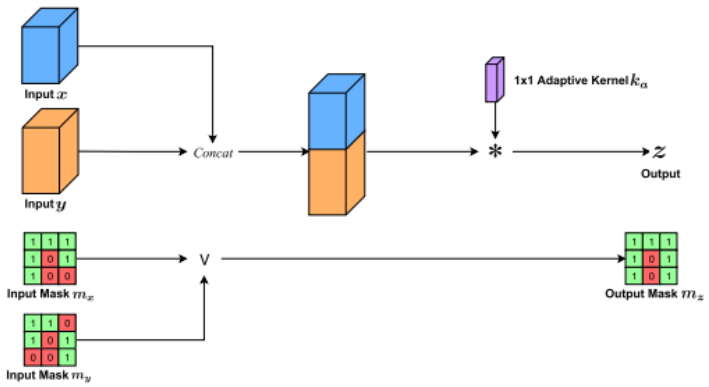

Figure 3.9: Illustration of the proposed joint sparsity-invariant concatenation and convolution.

3.4 Chapter Summary

Many deep learning based methods have been proposed for depth completion problem in the literature. These problems can be grouped in different categories based on some criteria. For example, based on input modalities, they can be categorized into two groups: depth-only completion and image guided completion. The depth-only methods use sparse depth map as input and infer the missing depth values to generate a full resolution depth map. The image guided methods take sparse depth along with the synchronized RGB image as input.

In this chapter we have discussed the depth only learning based depth completion problems, including HMS-Net and Sparsity Invariant CNN. Sparsity Invariant CNN handle different level of sparsity by using s a sparsity convolution layer. This is achieved by incorporating an observation mask in each convolution operation. The observation masks track the valid inputs along the convolution layers. At each layer , they are updated using a soft max function. However, the sparsity-invariant convolution only mimics the behavior of convolution operations in conventional dense CNNs. Its feature maps of later stages lose much spatial information and therefore cannot effectively integrate both low-level and high-level features for accurate depth completion.

Neural networks that utilize multi-scale feature maps for pixelwise prediction (e.g., semantic segmentation) were widely investigated. Combining both lowlevel and high-level features was proven to be crucial for making accurate pixelwise prediction. There exist effective multi-scale encoder-decoder network structures for dense pixelwise prediction tasks. Hierarchical multi-scale network is different from sparsity invariant CNN in that it can be used with multi-scale encoder-decoder network. Using only sparsity-invariant convolution, the multi-scale encoder-decoder networks cannot be directly converted to handle sparse inputs. This is because there exist many operations that do not support sparse feature maps. HMS-Net solve the problem by using three novel sparsity invariant operations and allow encoder-decoder networks to be used for sparse data.

Chapter 4

Mixing Learning Based and Classical Depth Completion Problems

4.1 Introduction

As we explained in the previous chapters, there are different ways of categorizing depth completion problems, based on the approach they adopted. These include guided or unguided, and learning based or non-learning based depth completion problems. While learning based methods require training to learn some parameters, non-learning methods entirely depend on a hand crafted image processing, which does not require training. In this paper we have covered two learning based methods, HMS-Net and Sparsity Invariant CNN in chapter 3 and one classical depth completion approach, IDW in chapter 2.

It is not customary to mix learning based approaches and non-learning classical approaches for depth completion task. In this paper we have designed a depth completion model that combines learning based problems with classical one. We demonstrate that by mixing the output depth features of a learning based method with a classical depth completion method, a better dense depth map could be produced. The first model is designed by mixing HMS-Net and IDW, where as the second one is obtained by mixing Sparsity Invariant CNN and IDW.

4.2 Mixing Sparsity Invariant CNN and Inverse Distance Weighting

In this section we describe how to mix Sparsity Invariant CNN with IDW for depth completion task. We adopted the network of Sparsity Invariant CNN in[3], and IDW in[5] to design our depth completion problem.
IDW has no any learnable parameters, and used only to modify the output depth feature of Sparsity Invariant CNN in the middle stages of the convolutional layers. It can be used to create a more dense depth feature maps in the training process. Our model of mixing Sparsity Invariant CNN and IDW is illustrated in figure 4.1.

As it can be seen in the figure, the sparse input depth is passed through a pre-trained Sparsity Invariant CNN adopted in[3], and the result is given to the first layer of our new convolutional layer. on the other side, the sparse input depth is modified in such a way that , the depth values of invalid pixels are replaced by depth values obtained from the pre-trained model at the corresponding positions. Then the modified sparse input depth is given as input to IDW. The invalid pixel depth values are modified using equation4.2. Then the outputs from IDW and the first layer of the newly added convolution layer, are mixed and passed to the next convolutional layers.

$$input_{IDW} = output * (1 - mask) + input \tag{4.1}$$

Where $input_{IDW}$ is input to IDW, output is output of the pre-trained model, input is the given sparse input depth, and mask is the mask of the sparse input depth.

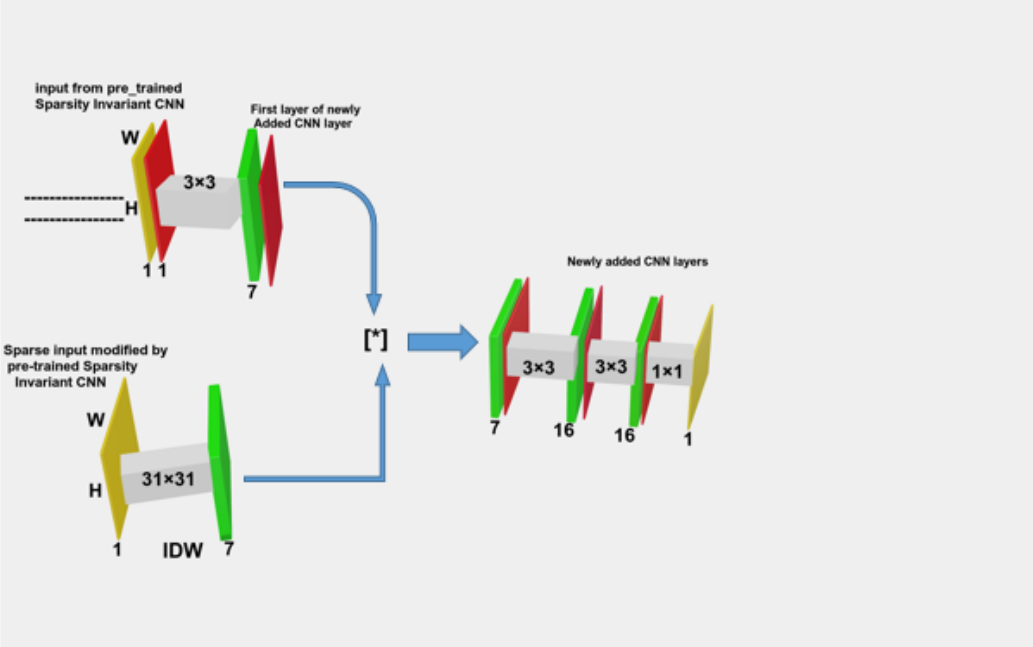

Figure 4.1: Mixing Sparsity Invariant CNN and IDW. $[\star]denoteschannelwisemultiplication$

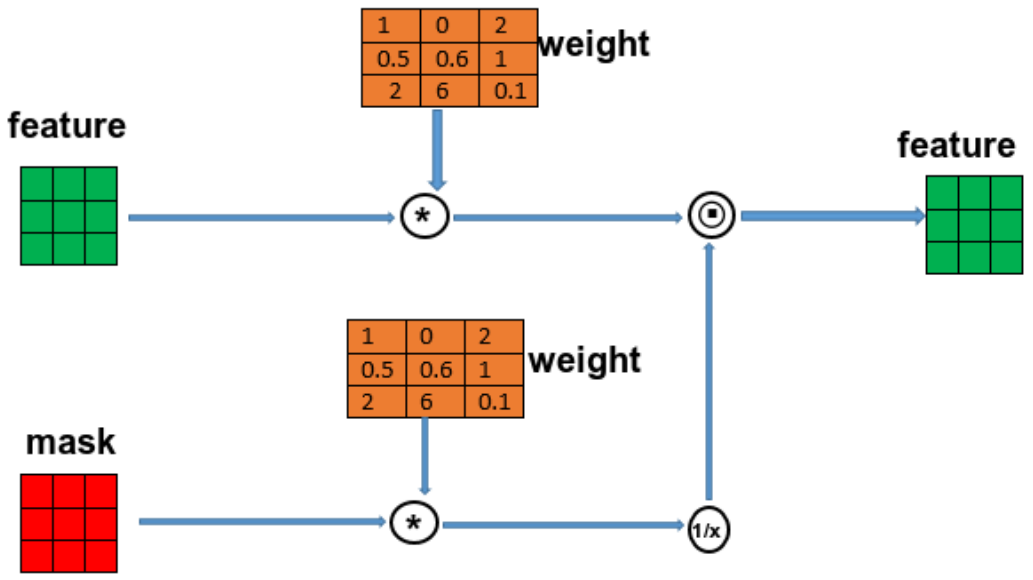

Figure 4.2: Operation of Inverse Distance Weighted

4.2.1 Methodology

The first step in our design of mixing Sparsity Invariant with IDW is to precompute a 31×31 kernel for the IDW interpolation using the formula describe in chapter 2 section 2.3. A kernel of size 31×31 will be computed for each power value between 2 and 4, producing 7 channels kernel or 7 different kernels. In particular we have used power values 2, 2.5, 2.8, 3, 3.2, 3.5, and 4. The power values are selected in such a way that their Root Mean Square Error (RMSE) is small. We have computed a depth feature maps for each power values from 1000 sparse depth inputs by convolving 31×31 kernel. Then the RMSE is computed from groundtruth depth values and the depth feature maps computed through IDW. The plotted RMSE graph is illustrated in figure 4.3.

Given a sparse depth input x and a corresponding mask m_x, a convolution operation is applied through the pre-trained model. Then the outputs of pre-trained model is used to modify the sparse input depth that will be used as input to IDW. Using a 7 channel kernel, IDW nework produces a 7 channel depth feature map. There are no learnable parameters in IDW, and the kernels are computed offline. The IDW operation is showed in figure 4.2. In the figure $\bigodot$ denotes channelwise multiplication , and $\circledast$ denote a convolution operation.

In the second stream of our model, the outputs of the pre-trained model and it's corresponding mask will be given as input to the first layer of the new convolution layer. The outputs of this layer and IDW are mixed (channelwise multiplied) and produce a 7 channel depth feature map. Then the mixed output depth is further processed by the remaining convolutional layers. In each layer a ReLu activation function is applied except the first newly added convolution layer. We aplied a softmax function for the layer that is directly mixed with IDW.
The reason that we have used softmax function is to create some sort of probability distributions for each pixel along the channel dimension. This lets the network to choose the power values based on some probability. The operation of Sparsity Invariant CNN is illustrated in chapter 3 section 3.2.3 figure 3.3.

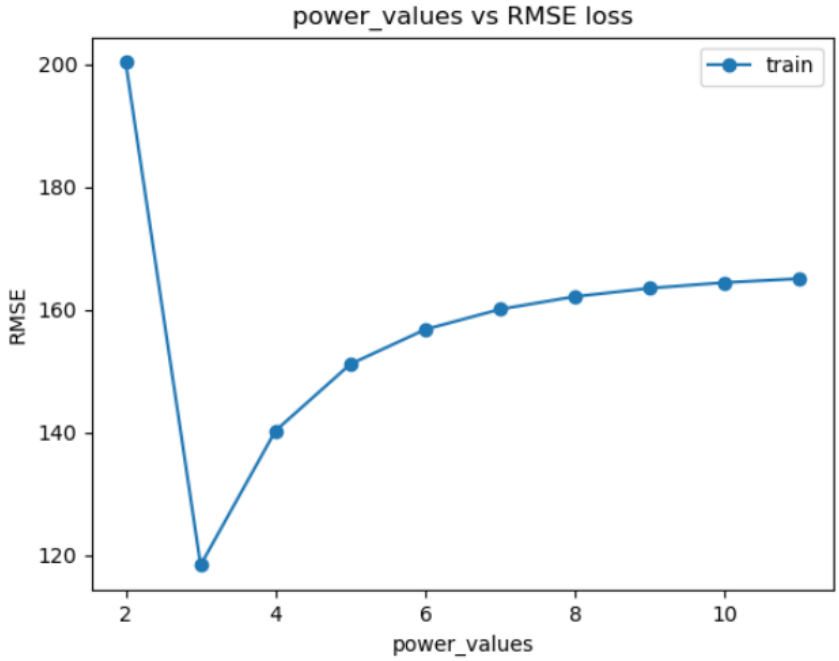

Figure 4.3: Illustration of RMSE error for output depth features computed by IDW for different power values

4.3 Mixing Hierarchical Multi-Scale Network and Inverse Distance Weighting

The second design of our work is made by mixing HMS-Net and IDW. The procedure is similar with the previous design, except that here we used HMS-Net instead of Sparsity Invariant Convolution as a pre-trained model. The new convolution layers added on top of the pre-trained model are similar with the one used in our first mixed model.

4.4 Chapter Summary

In this chapter we have described our new designs for depth completion problem. The first model mixes IDW and Sparsity Invariant CNN, where as the second model is obtained by mixing HMS-Net and IDW. We first trained Sparsity Invariant CNN and HMS-Net, and the pre-trained models are used to mix with IDW. after the mix is performed, the result is further convolved through some convolutional layers. The extra convolutional layers added on top of the pretrained network and IDW are simple sparsity invariant convolutions. Basically, we have used the pretrained network as feature extractors.

Chapter 5

Experiment and Results

In the previous chapters, we have discussed the depth completion problems that we have adopted to design our depth completion problem, and the proposed depth completion problems. These are Inverse distance weighted interpolation (IDW), Hierarchical multi-scale encoder-decoder network(HMS-Net), and Sparsity Invariant CNN.
HMS-Net and Sparsity Invariant CNN are a learning based depth completion methods that uses deep convolution neural network for extracting depth features at different layers, where as, IDW is a simple interpolation technique that uses distance metric to compute the weights. In this chapter we describe about the experimental setup, different evaluation metrics, and the dataset and experimental results.

5.1 Experimental Setups

In this section we will look into the different parameters and hyper parameters setting for our experiment. We have used the same network parameters and hyper-parameters for each of our depth completion problem designs, in order for us to compare them under the same setup. We first implemented HMS-Net and Sparsity Invariant CNN, and these pre-trained models were used to train our newly designed network. ALL the models were trained with GPU.

5.2 Training Scheme

As we briefly explained in the previous chapters, our first design for depth completion problems is to mix Sparsity Invaiant CNN and IDW, and the second one

is mixing HMS-Net and IDW. Both models are trained with the same network setup. The models learns only the parameters of newly added convolution layers, as Sparsity Invariant CNN, and HMS-Net are pre-trained and has no any learnable parameters.
We adopt the mean squared error (MSE) loss function to train our proposed models. Since some datasets could only provide sparse ground-truth depth maps, the loss function is only evaluated at locations with groundtruth annotations, which could be formulated as

$$L(x, y) = \frac{1}{|V|} \sum_{u,v \in V} |o(u, v) - t(u, v)|^2 \tag{5.1}$$

where $\mathbf{V}$ is the set containing coordinates with ground-truth depth values, $|V|$ calculates the total number of valid points in $\mathbf{V}$, and o and t are the predicted and ground-truth depth maps.
For the newly added network training, all parameters are randomly initialized. We adopt the learning rate, optimizer, learning rate decay factor of HMS-Net in[2]. We adopt the ADAM optimizer with an initial learning rate of 0.01. The networks are trained for 50 epochs. To gradually decrease the learning rate, it is decayed according to the following equation,

$$Learningrate = 0.01 \times (1 - \frac{iter\_epoch}{50})^{0.9} \tag{5.2}$$

where $iter\_epoch$ denotes the current epoch iteration. As we mentioned in the previous chapters, sparsity masks are generated for every input directly depending on the network structure, without any learnable parameters.
In Inverse Distance Weighting, we have chosen a kernel size of 31×31, and 7 power values between 2 and 4 are used to produce a stack of 7 kernels. The number of power values are chosen in such a way that they produce optimal result for our experiment.

5.3 Dataset and Evaluation Metrics

We evaluate our proposed approach and all the other adopted methods on the KITTI depth completion benchmark [8]. Following the experimental setup , 33,200 depth maps are used for training, 1,000 for validation and 1,000 for test. The input depth maps generally contains less than 10 % sparse points with depth values and the top $1/3$ of the input maps do not contain any depth measurements.
According to the benchmark, all algorithms are evaluated according to the following metrics, root mean square error (RMSE in mm), mean absolute error (MAE in mm), root mean squared error of the inverse depth (iRMSE in 1/km), and mean absolute error of the inverse depth (iMAE in 1/km). The metrics are computed using the following formulas.

$$RMSE == (\frac{1}{|V|}\sum_{u,v\in V}|o(u,v)-t(u,v)|^2)^{0.5} \tag{5.3}$$

$$MAE == \frac{1}{|V|}\sum_{u,v\in V}|o(u,v)-t(u,v)| \tag{5.4}$$

$$IRMSE == (\frac{1}{|V|}\sum_{u,v\in V}|\frac{1}{o(u,v)}-\frac{1}{t(u,v)}|^2)^{0.5} \tag{5.5}$$

$$IMAE == \frac{1}{|V|}\sum_{u,v\in V}|\frac{1}{o(u,v)}-\frac{1}{t(u,v)}| \tag{5.6}$$

where o and t represent output and ground-truth depth values.

For RMSE and MAE, RMSE is more sensitive to large errors compared. This is because even a small number of large errors would be magnified by the square operation and dominate the overall loss value. RMSE is therefore chosen as the main metric for ranking different algorithms in the KITTI leaderboard. Since large depth values usually have greater errors and might dominate the calculation of RMSE and MAE, IRMSE and IMAE are also evaluated, which calculate the mean of inverse of depth errors. In this way, large depth values' errors would have much lower weights on the two metrics.

5.4 Evaluation and Experimental Results

In this section we will discuss the results of the experiments and the metrics values computed. Based on the experimental setups that we discussed in the previous section, we will demonstrate the training and validation losses, RMSE and MAE error, and the predicted output depth.

5.4.1 Sparsity Invariant CNN with IDW

Now, let's review the results of the first mixed model we mentioned for depth completion problem. Figure 5.1 illustrates the training and validation loss for sparsity invariant CNN and our mixed models of sparsity invariant CNN with IDW. From the figure, it is shown that the mixed model of sparsity Invariant CNN and IDW has a lower training and validation loss. Sparsity invariant CNN has a smooth convergence for the validation loss compared to the mixed model. The training loss has a smooth convergence in both models. Both models are trained for 50 epochs, but sparsity invariant CNN found its best loss earlier than the mixed model.

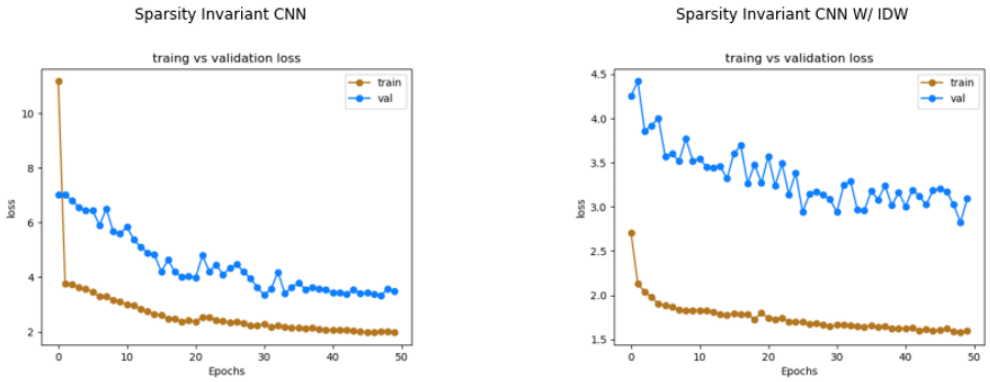

Figure 5.1: Sparsity Invariant CNN and Sparsity Invariant CNN mixed with IDW Training/Validation Loss

In Figure 5.2, we have plotted the histogram and kernel density estimation(KDE) of the RMSE for 1000 testing set depth maps. This allows to figure out the RMSE distribution of the two models for the whole test dataset. As you can see, the distribution of errors follow the typical bell curve pattern of normal distributions, except that there are some outliers. Sparsity Invariant CNN has an average RMSE of 1358, and standard deviation of 516, while the mixed model has an average RMSE of 1275 and standard deviation of 470. The standard deviation is computed on the RMSE.

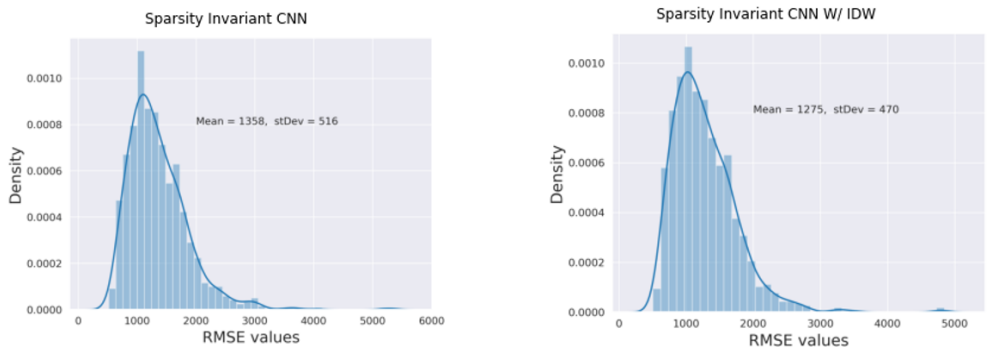

Figure 5.2: Histogram and Kernel Density estimation(KDE) of the RMSE for the test dataset

The output depth maps of the two models are shown in Figure 5.3. From top to down, the first row is the image, second row the raw-data, third row the ground-truth, fourth row the output depth map of sparsity invariant CNN, and the fifth row is the output depth map of our newly mixed model. It is very difficult to visually identify the output depth maps of the two models. But, in the mixed model, the depth maps are a little bit sharper, it recovered depth values for very far objects from the camera plane. Besides, in the mixed model, some artifacts that we see in the sparsity invariant CNN are removed. The zoomed output depth maps of the second column in 5.3, is re-shown in Figure 5.4

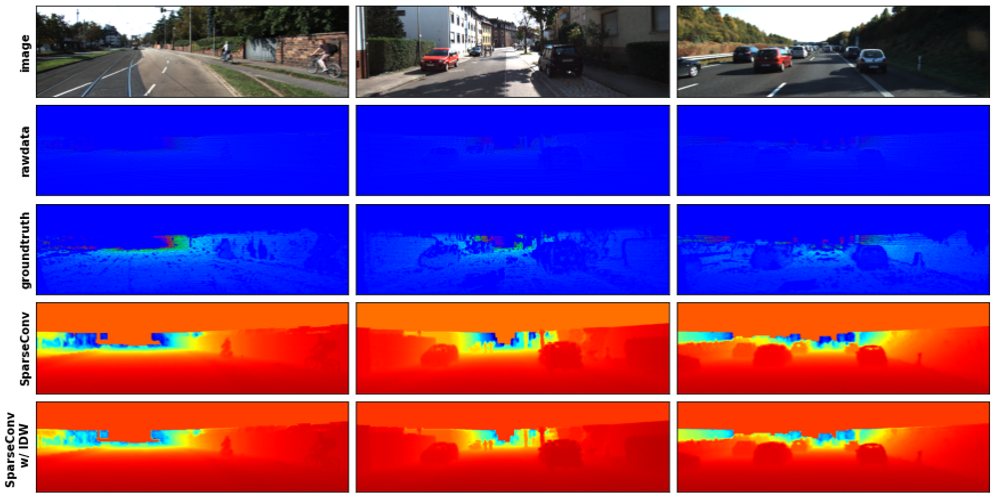

Figure 5.3: Predicted depth map of Sparsity Invariant CNN and Sparsity Invariant CNN with IDW

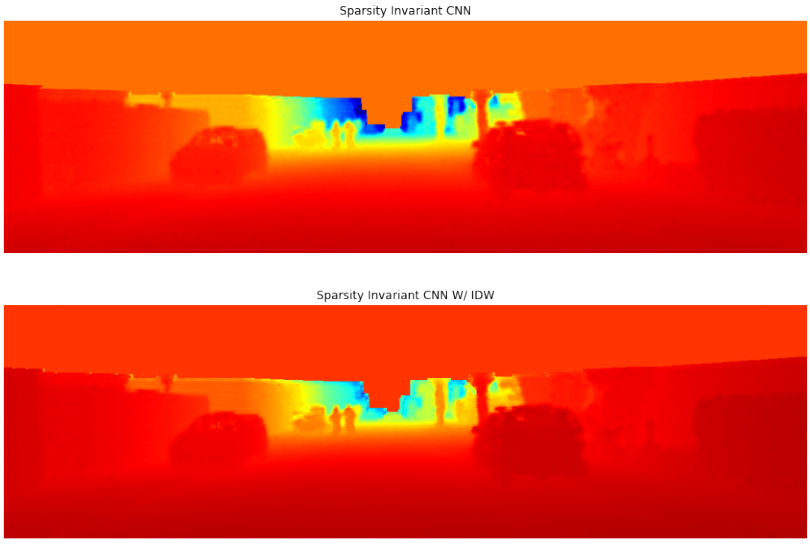

Figure 5.4: Predicted depth map of Sparsity Invariant CNN(Top) and Sparsity Invariant CNN with IDW(bottom)

Finally, we have plotted the error, groundtruth and output depth as an image to better visualize the errors at each pixel location. Though, it is possible to see some color differences, it is generally difficult to visually identify the error maps of the two models. Figure 5.6 shows the error maps shown in figure 5.5, second column, widely zoomed.

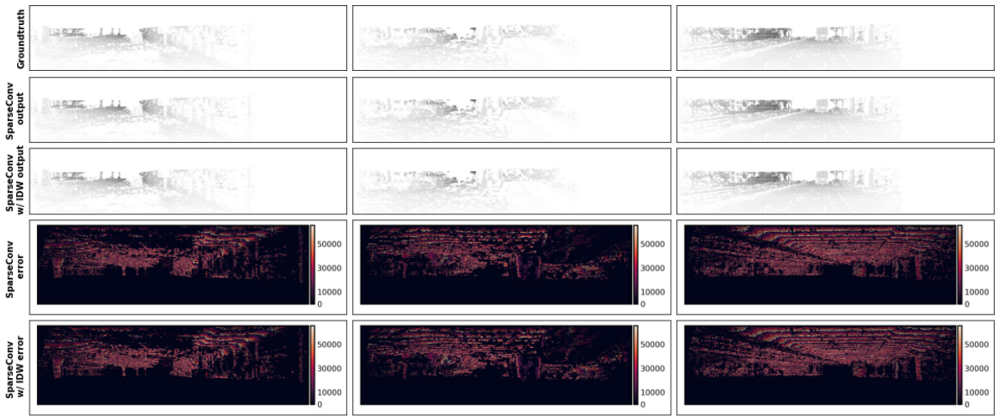

Figure 5.5: The Errors, groundtruth and depth output of Sparsity Invariant CNN(SparseConv) and SparseConv mixed with IDW as an image

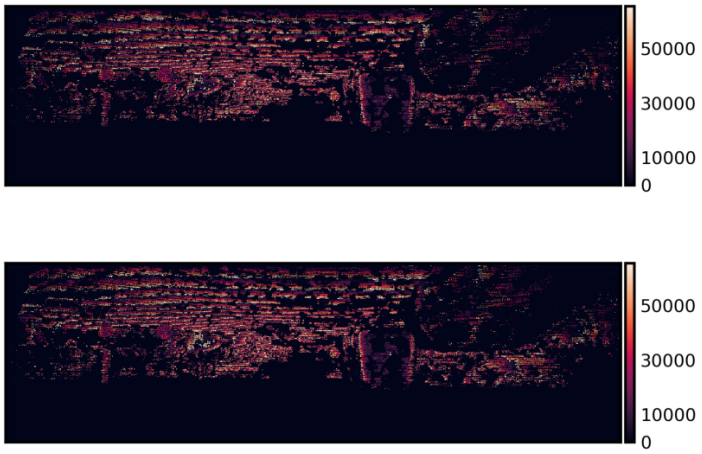

Figure 5.6: The Errors, groundtruth and depth output of Sparsity Invariant CNN(top) and SparseConv mixed with IDW(bottom) as an image

5.4.2 HMS-Net with IDW

The second design of our model is HMS-Net mixed with IDW. In this section we will look at the experimental results of the mixed model and make some comparative analysis with HMS-Net. Similar to the first mixed model, we have also plotted the training / validation loss, the RMSE for 50 randomly selected output depth maps, and the predicted results of HMS-Net mixed with IDW. The training / validation loss is illustrated in Figure 5.7. While HMS-Net has a smooth training and validation loss curve, the mixed model has a very noisy curve for validation loss. HMS-Net is started from a high training / validation loss and quickly dropped at the beginning of the iteration, and not much improvement has seen in the next iterations. On the other hand, the mixed model has started from a very low training / validation loss, and it has seen a slow drop in loss in the subsequent epochs.

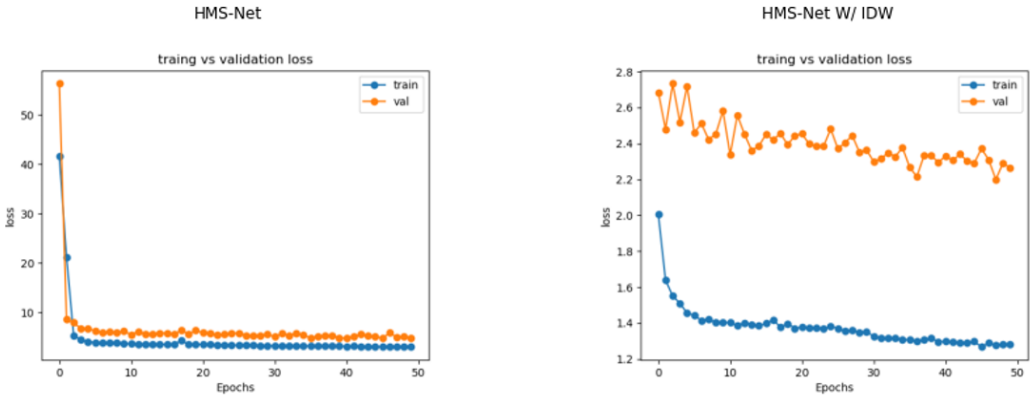

Figure 5.7: HMS-Net and HMS-Net mixed with IDW Training/Validation Loss

Figure 5.8 below, illustrates the histogram and kernel density estimation(KDE) of HMS-Net, and HMS-Net mixed with IDW. The two models have a very close average RMSE and standard deviation values. Most errors are close to the average error. As the previous models, The standard deviation is computed on the RMSE.

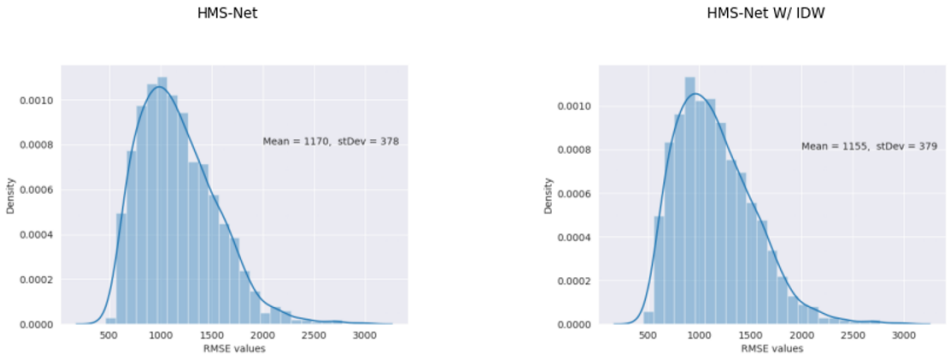

Figure 5.8: Histogram and Kernel Density estimation(KDE) of the RMSE for the test dataset

The below figure 5.9 shows the predicted depth maps for the given sparse depth inputs. For better visualization, the zoomed output depth map of Figure 5.9, first column, is redrawn in Figure 5.10.

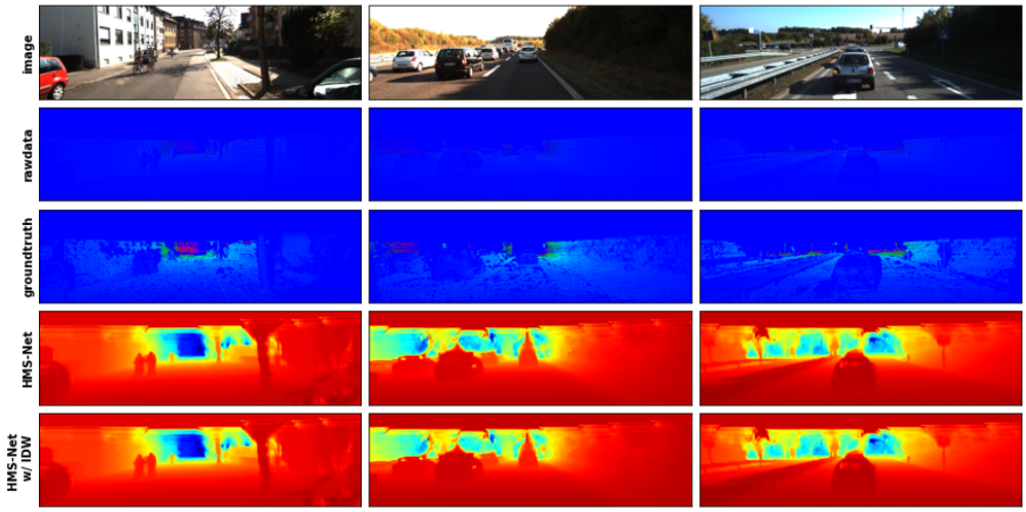

Figure 5.9: Predicted depth map of HMS-Net and HMS-Net mixed with IDW

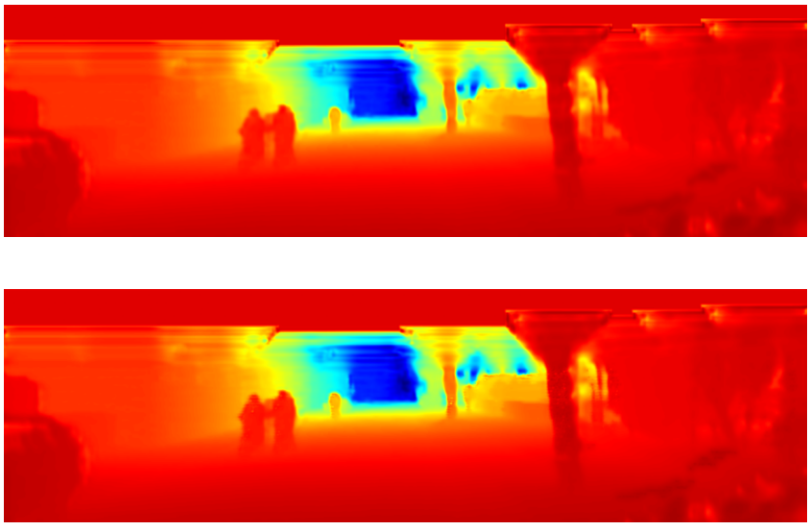

Figure 5.10: Predicted depth map of HMS-Net(Top) and HMS-Net with IDW(bottom)

5.4.3 Error Metrics

To evaluate the performance of our designed depth completion problems, and to make some comparative analysis with other state of the art problems, we have computed several error metrics. In addition to the error metrics that we described in section 5.3, we have also used Standard Deviation (STD). The STD in this table is computed using Mean Absolute Error (MAE). Among all the metrics used, RMSE is chosen as the main metric for ranking different algorithms in the KITTI leader-board. In the table, we have included the metrics for In Defense of Classical Image Processing: Fast Depth Completion on the CPU[1](FastDepth), Sparsity Invariant CNN[3](SparseConv), Hierachical multi-scale encoder-decoder network[2](HMS-Net), and our two mixed models, Sparsity Invariant CNN mixed with IDW and HMS-Net mixed with IDW.
Based on the experimental results, our mixed models have beaten both Sparsity Invariant CNN[3], and HMS-Net[2]. Especially, we have improved the RMSE value of Sparsity Invariant CNN[3] from 1359 to 1275 through our Sparsity Invariant CNN mixed with IDW model. The second design of our mixed model has also improved the RMSE value of HMS-Net[2] form 1171 to 1155.

Table 5.1: DEPTH COMPLETION ERRORS BY DIFFERENT METHODS ON THE TEST SET OF KITTI DEPTH COMPLETION BENCHMARK.

Methods	**RMSE**(in mm)	**MAE**(in mm)	**STD**	**IRMSE**(in m)	**IMAE**(in m)
FastDepth[1]	1345	304	126.2	–	–
SparseConvs[3]	1359	415	127.3	5.4	2.1
HMS-Net[2]	1171	347	107.7	4	1.5
SparseConvs w/ IDW	1275	373	118.3	–	–
HMS-Net w/ IDW	1155	332	105.9	4.19	1.49

Chapter 6

Conclusion

In this paper, we have designed two depth completion problems, one is mixing Sparsity Invariant CNN with IDW, and the other is mixing HMS-Net with IDW. We were able to evaluate and compare our mixed models with other selected algorithms. By using the KITTI depth completion benchmark [8] as our training and evaluation dataset, We compare their performance using different metrics. And we showed the quantitative results of the metrics.

We saw that our mixed model of Sparsity Invariant CNN and IDW have well performed compared to the mixed model of HMS-Net and IDW. It has improved the results obtained in Sparsity Invariant CNN[3] in wide range. The second model, HMS-Net mixed with IDW has also improved the results in HMS-Net[2] in a narrow range. The reason that the improvement seen in this model was not significant, could be because of the new CNN layers added after mixing it with IDW. We have used the same CNN layers as that of the first mixed model.
The absence of common visualization method for the predicted depth maps makes it difficult to visually compare very small details.

References

[1] J. Ku, A. Harakeh, and S. L. Waslander, "In defense of classical image processing: Fast depth completion on the cpu," in *2018 15th Conference on Computer and Robot Vision (CRV)*, pp. 16–22, IEEE, 2018.

[2] Z. Huang, J. Fan, S. Cheng, S. Yi, X. Wang, and H. Li, "Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion," *IEEE Transactions on Image Processing*, vol. PP, pp. 1–1, 12 2019.

[3] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," 08 2017.

[4] F. Ma, G. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera," 07 2018.

[5] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM national conference*, pp. 517–524, 1968.

[6] M. Michalak, "On interval estimating regression," *Computer Science Information Technology*, vol. 4, pp. 45–53, 06 2014.

[7] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," 09 2017.

[8] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *International Conference on 3D Vision (3DV)*, 2017.

[9] S. C. M. M. R. E. HEITMAN, "Statistical analyses of ocean terrain and contour plotting procedures," 04 1964.

[10] C. deBOOR, "Bicubic spline interpolation," 1962.

[11] J. A. DOWNING, "automatic construction of contour plots with applications to numerical analysis resear," 01 1966.

[12] B. N. P. ZHIDKOV, "Computing methods addison-wesley publishing company inc," 01 1965.

[13] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," 12 2018.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[15] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," 11 2016.

[16] S. Matyunin, D. Vatolin, Y. Berdnikov, and M. Smirnov, "Temporal filtering for depth maps generated by kinect depth camera," *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1–4, 05 2011.

[17] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep joint image filtering," in *European Conference on Computer Vision*, pp. 154–169, Springer, 2016.

[18] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," 2017.

[19] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," *arXiv preprint arXiv:1608.07916*, 2016.

[20] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof, "Image guided depth upsampling using anisotropic total generalized variation," in *International Conference on Computer Vision*, ., 2013. International Conference on Computer Vision : ICCV 2017 ; Conference date: 24-10-2017 Through 27-10-2017.

[21] G. Riegler, M. Rüther, and H. Bischof, "Atgv-net: Accurate depth super-resolution," *CoRR*, vol. abs/1607.07988, 2016.