The Dissertation Committee for Janice S. Pan
certifies that this is the approved version of the following dissertation:

Perceptual Monocular Depth Estimation

Committee:

Alan C. Bovik, Supervisor

Joydeep Ghosh

Qixing Huang

Martin Mueller

Haris Vikalo

Perceptual Monocular Depth Estimation

by

Janice S. Pan

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2019

Dedicated to my parents.

Acknowledgments

I wish to express my deepest gratitude for those who guided and supported me. I would not have completed this work without the support of my advisor, teachers, mentors, family, and friends:

- Prof. Al. Bovik, my advisor, who took a chance and accepted me into his lab after I did my undergraduate Senior Design Project with him – I relied heavily on his advice, guidance, and patience throughout my graduate career, and he taught me how to think independently and confidently about my research. He always gave me enough space to explore my own ideas, while also checking in to make sure I was heading in the right direction.

- My other teachers and mentors, both at UT Austin and at Texas Instruments, who kindly funded my project for most of my graduate career, and where I had the honor of interning one summer – I want to thank Prof. Joydeep Ghosh and Prof. Haris Vikalo for their general guidance and encouragement from the first classes I took with them; Prof. Qixing Huang for his honest feedback and his helpful advice when I was trying to find footing with using deep learning in my research; Vikram Appia, my supervisor at Texas Instruments, who was always patient with me and taught me how to organize and give effective presentations; and

Martin Mueller and Tarek Lahlou, also formerly with Texas Instruments, who always made me feel that my work was important and provided invaluable feedback and suggestions early on.

- Past and present LIVE members (particularly Deepti, Zeina, Praful, Todd, Christos, Leo, Lark, Tony, Meixu, Yize, Jerry, and Zhengzhong) and other WNCGers (Debarati and Jason) for their friendships, collaborations, discussions, and mental support during my years in graduate school – Deepti was both a mentor and a friend to me when I first started my research. We collaborated on a few projects, and her kindness and patience taught me so much about how to work and communicate with others. Zeina inspired me (and still does) to take more initiative in setting my own path. Lark is one of the most selfless people I have had the privilege of meeting. He was always eager to help in any way he could, and even since he graduated and left Austin, he has continued to inspire me to work hard not only for myself but for those I care about as well. And Praful and Jason were with me until the end, available anytime to lend an ear or offer advice. It was always a comfort to come to work and see any of these people's faces.

- My family – My parents, Tinsu Pan and Dershan Luo, instilled in me a work ethic and sense of discipline to set goals and strive to achieve them. They are my biggest cheerleaders and encouraged me to pursue any opportunity I was so fortunate to receive. And my younger sisters, Cassie and Emily, are my two biggest inspirations. When I see their dedication to accomplish whatever they set their minds to, how highly they achieve through sheer hard work and without complaints, and how they manage the stress and pressure they put on themselves, I aspire to have the same drive and strength of mind.

- Na and Vi, who many times believed in me more than I believed in myself – I am lucky to have such supportive, encouraging, and caring friends. They are the reasons I was able to develop and maintain some semblance of balance in my life between working and everything else.

- And last but never least, Jonathan, who has inspired me to think more deeply about who I want to be, to not be afraid of setting higher goals, and to be more confident in my own abilities. He has undoubtedly made me a better person.

Perceptual Monocular Depth Estimation

Publication No. ____________________

Janice S. Pan, Ph.D.
The University of Texas at Austin, 2019

Supervisor: Alan C. Bovik

Monocular depth estimation (MDE), which is the task of using a single image to predict scene depths, has gained considerable interest, in large part owing to the popularity of applying deep learning methods to solve "computer vision problems". Monocular cues provide sufficient data for humans to instantaneously extract an understanding of scene geometries and relative depths, which is evidence of both the processing power of the human visual system and the predictive power of the monocular data. However, developing computational models to predict depth from monocular images remains challenging. Hand-designed MDE features do not perform particularly well, and even current "deep" models are still evolving. Here we propose a novel approach that uses perceptually-relevant natural scene statistics (NSS) features to predict depths from monocular images in a simple, scale-agnostic way that is competitive with state-of-the-art systems. While the statistics of natural photographic images have been successfully used in a variety of image and video processing, analysis, and quality assessment tasks, they have never been applied in a predictive end-to-end deep-learning model for monocular depth. Here we accomplish this by developing a new closed-form bivariate model of image luminances and use features extracted from this model and from other NSS models to drive a novel deep learning framework for predicting depth given a single image.

We then extend our perceptually-based MDE model to fisheye images, which suffer from severe spatial distortions, and we show that our method that uses monocular cues performs comparably to our best fisheye stereo matching approach. Fisheye cameras have become increasingly popular in automotive applications, because they provide a wider (approximately 180 degrees) field-of-view (FoV), thereby giving drivers and driver assistance systems more visibility with minimal hardware. We explore fisheye stereo as it pertains to the problem of automotive surround-view (SV), specifically, which is a system comprising four fisheye cameras positioned on the front, right, rear, and left sides of a vehicle. The SV system perspectively transforms the images captured by these four cameras and stitches them together in a birdseye-view representation of the scene centered around the ego vehicle to display to the driver. With the camera axes oriented orthogonally away from each other and with each camera capturing approximately 180 degrees laterally, there exists an overlap in FoVs between adjacent cameras. It is within these regions where we have stereo vision, and can thus triangulate depths with an appropriate correspondence matching method. Each stereo system within the SV configuration has a wide baseline and two orthogonally-divergent camera axes, both of which make traditional methods for estimating stereo correspondences perform poorly. Our stereo pipeline, which relies on a neural network trained for predicting stereo correspondences, performs well even when the stereo system has limited overlap in FoVs and two dissimilar views. Our monocular approach, however, can be applied to entire fisheye images and does not rely on the underlying geometry of the stereo configuration. We compare these two depth-prediction methods in both performance and application.

To explore stereo correspondence matching using fisheye images and MDE in non-fisheye images, we also generated a large-scale photorealistic synthetic database containing co-registered RGB images and depth maps using a simulated SV camera configuration. The database was first captured using fisheye cameras with known intrinsic parameters, and the fisheye distortions were then removed to create the non-fisheye portion of the database. We detail the process of creating the synthetic-but-realistic city scene in which we captured the images and depth maps along with the methodology for generating such a large, varied, and generalizable dataset.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The human visual system (HVS) uses both binocular and monocular cues to perceive depths and instantaneously reconstruct the 3D world. Even from a single image, without binocular cues, humans are able to easily gain a good understanding of the 3D geometry of the scene, e.g., relative distances and object sizes. Recently, modeling depth from monocular luminance has gained interest, and deep learning approaches to depth estimation have shown promising results. However, designing computational models to estimate depth from monocular images, or Monocular Depth Estimation (MDE), remains an ill-posed, challenging problem.

Recent work in estimating depth from monocular images has made use of deep network architectures and training on large databases [8, 17, 21, 27, 28, 51, 53, 56, 57, 114]. Some models were trained with stereo or multiple views [27, 28, 51], but the amount of multi-view ground-truth data that is available is limited. Other approaches require post-processing, refinement steps, or multi-phase training [17, 56] or use networks having tens of millions of trained parameters [8, 21, 53, 57, 114]. One previous approach used natural image statistic features in a Bayesian framework to regress depth using a simpler learner (Support Vector Regressor) [102], however that model requires training two distinct sub-models, and depth-prediction is subsequently a two-part process. Here we show that perceptually available information provided by the local, scale-invariant statistics of natural scenes, or natural scene statistics (NSS), can be used to predict depth in a simple end-to-end network, yielding significantly improved results.

Natural Scene Statistics (NSS) have proven to be useful in understanding the evolution of the HVS and for exploring complex visual problems [7, 78, 91, 104, 112]. In a related study, the authors of [58] deployed statistical models of luminance/chrominance and depth/disparity NSS, and the relationships between them, to drive a Bayesian stereo algorithm to predict disparity. The authors of [97] extended that work by introducing NSS models of the marginal and conditional distributions of luminances/chrominances and depths/disparities of natural scenes and used them to improve a chromatic Bayesian stereo algorithm. More recently, bivariate and correlation NSS models for natural images and depth maps were developed, but these only operate on spatially adjacent bandpass responses [99, 103]. NSS models of spatially adjacent luminance coefficients have also been shown to provide strong features on visual problems [66, 67], but they have yet to be applied to the problem of depth prediction method, monocular, stereo, or otherwise.

Here we advance the problem of perceptual depth prediction, by first introducing a new closed-form correlation model of local bivariate luminance statistics. We then use features derived from this model, along with other univariate and bivariate NSS features, to train a deep neural network (DNN) to predict inverse-range patches from luminance patches. We choose to predict inverse-range (versus depth) because the goal of traditional stereo correspondence matching methods is to estimate disparity, which is inversely proportional to depth, so our use of inverse-range is natural in a depth-prediction context. Therefore, we refer to our monocular inverse-range prediction model throughout this report as a 'depth'-prediction model.

When building this model, we represent each inverse-range patch using a sparse code, each element of which supplies a weight on each patch within a dictionary of image patches, which was learned from patches sampled from depth perception databases of ground-truth luminance-depth images. We discuss the construction of our network and emphasize connections and parallels to perceptual models and concepts. We explain the new correlation NSS model, the dictionary construction, and the network design, including the data processing steps, the network architecture, the training and testing details, and the results. We show that our simple network trained using perceptually-relevant features and a perceptually-relevant loss function is able to predict dense depth maps in a manner that is competitive with state-of-the-art methods.

Motivated by the popularity of using wide-angle lenses in automotive applications, we also extend our model for MDE to fisheye images. We first detail our algorithm for predicting depth from a wide-baseline orthogonally-divergent stereo fisheye system, four of which comprise the automotive surround-view (SV) system, and then compare the depths predicted with both the monocular and binocular methods. Fisheye cameras are able to provide a wider (near-180^o) field-of-view (FoV) compared to traditional rectilinear cameras, thereby giving more visibility using very minimal hardware. Stereo systems are typically comprised of a pair of rectilinear cameras with parallel camera axes and narrow ($\sim$ 10 cm) baselines. The use of fisheye lenses enables stereo configurations to use wider baselines and non-parallel camera axes, since they capture larger FoVs that may still overlap under disparate camera positions and orientations. However, using fisheye lenses for stereo applications is inherently more challenging because wide-angle lenses produce spatial image distortions that get more severe closer to image edges, which are coincidentally where FoVs overlap. Additionally, trying to change the fisheye stereo problem to be a rectilinear stereo problem so that traditional stereo methods can be applied presents its own challenges, because information can be lost, particularly at the image edges, when wide-angle images are converted to rectilinear images, and distortions in the 'undistorted' image can still be present due to the spatial spreading of pixels. Therefore, we adapt our NSS-based MDE model for native fisheye images and demonstrate its efficacy in predicting depths without requiring co-registered fisheye stereo images or first removing the radial distortion. We compare these results with our method for estimating depth from fisheye stereo using a neural network trained for correspondence prediction, first published in [76], and show that the monocular method performs better in both accuracy and efficiency.

Chapter 2

Background

We first review related work in developing natural scene statistics (NSS) models, because the monocular depth estimation network we designed and trained utilizes such NSS features extracted from the input image patches. We also discuss related work in estimating depth from monocular images before reviewing previous work in depth estimation for fisheye images, which, thus far, has been constrained to stereo approaches, as no monocular work has been done in the fisheye space.

2.1 Natural Scene Statistics

Extensive work has been done analyzing and modeling the natural statistics of 2D and 3D scenes and how the HVS processes this information. In particular, many models have been developed that make use of statistical models of bandpass responses of luminance and depth/disparity of real-world images [19,58,79,84,97,102,110]. Using locally normalized *luminance coefficients* when modeling such relationships has been explored to a lesser extent, though NSS computed in this domain are also perceptually relevant [66, 67].

There exists a strong relationship between luminance and co-located depth in natural scenes [79]. Statistical models have been established that reliably capture the univariate [58, 97] and bivariate [99, 103] statistics of real-world photographs quite well. There are a variety of useful established closed-form NSS models of bandpass images [20, 91, 93, 100, 101], and recent work using them for the monocular depth estimation problem has shown promising results [102]. However, a bivariate model of luminance statistics as a function of orientation and separation between pixels would better capture the relationships between pixels and scene geometry. We develop and introduce a closed-form bivariate model for luminance coefficients in Section 4.

2.2 Monocular Depth Estimation

Estimating dense depth maps of naturalistic scenes is useful for many computer vision tasks, such as scene reconstruction and object detection/recognition. Estimating scene depths given only a single monocular image is a much more challenging problem than multi-view (stereo) or multi-frame (video) depth estimation. The latter two problems have been studied extensively [31, 40, 47, 76, 81, 119], however the problem of monocular depth estimation is a much more poorly-posed problem, and progress towards developing a solution has been more difficult.

Recent approaches to MDE that utilize DNNs have shown promising results [8, 17, 21, 43, 53, 56, 57, 83, 114]. However, some of the highest-performing methods limit the resolution of the output [17], require post-processing (correction) steps [8], require multi-phase training [114], or reformulate the problem as a variation of a classification task [8,21]. Some recent methods for MDE rely on stereo or multi-view data for training [22,27,28], thus potentially limiting their generalizability.

Instead of predicting depth, we predict the inverse of depth, which is proportional to disparity, and which we refer to as *inverse-range*. The task of predicting the inverse of depth follows the objective of stereo correspondence methods. Stereo algorithms aim to estimate disparity, which can be used to compute depth directly if the baseline and focal length of the stereo cameras are known. Our method uses a novel modular DNN architecture that relies solely on monocular training data. It can be trained end-to-end, and because it operates in a patch-wise manner, it is able to produce predictions for images of any resolution.

2.3 Depth Estimation using Fisheye Images

We now discuss past work in estimating depth using fisheye images, beginning with stereo approaches using the more traditional rectilinear cameras, so that we can then understand how using spatial distortions in images add challenges and complexity to the problem that require modifications to these methods.

2.3.1 Stereo Depth Estimation

The goal of stereo algorithms is to determine point correspondences between a stereo image pair, which means finding which pairs of points in the stereo images correspond to the same world points. Stereo matching is governed by epipolar geometry, which relates correspondences between views. Potential matches for any point in the reference (typically left) image, must lie on a corresponding *epipolar* line in the matching image. With rectilinear cameras, image rectification is almost always applied in a preprocessing step, because rectification transforms the pair to align on a common image plane, thereby aligning the epipolar lines, and as a result, corresponding points align along the same rows. Image rectification, therefore, simplifies the correspondence matching problem to be one-dimensional. In other words, for each point in the reference image, the stereo algorithm only needs to search along the same row in the corresponding image for the matching point. Once a matching pair of points is found, disparity can be computed as the horizontal displacement between the points in each view. Stereo algorithms typically output a disparity map, where a disparity value is given for each pixel in the reference image. Depth can then be computed from disparity, as they are inversely proportional. Figure 2.1 shows an example stereo pair that was taken with traditional rectilinear cameras. The images have also been rectified such that corresponding points lie along the same rows. Disparity is then simply the difference between the x-coordinates, $\text{disp} = |x_1 - x_2|$, and $\text{depth} = \frac{fB}{\text{disp}}$, where f and B are the lens focal length and stereo baseline, respectively.

There has been a lot of work on developing and improving traditional stereo methods for camera configurations involving rectilinear cameras, and there is no shortage of robust approaches, which is why the first step to many

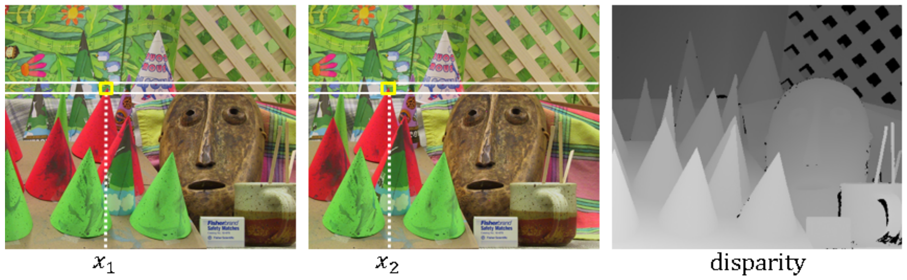


Figure 2.1: Example from [87]. A stereo pair of images taken with rectilinear cameras with their corresponding disparity map. The left image is taken as the reference. After applying image rectification, epipolar lines align along rows, so matching points can be found via a 1D search. The yellow boxes indicate an example pair of matching points, and their horizontal displacements are x_1 and x_2, which are used to compute disparity and, subsequently, depth.

fisheye stereo methods is to remove the fisheye distortion. Some traditional stereo-matching algorithms rely on local [118] and/or global costs [36, 42], a good example being the popular Semi-global Matching (SGM) [36] which is commonly used in Advanced Driver Assistance Systems (ADAS) [71] and autonomous driving applications [124]. The SGM method uses a local pixelwise matching of Mutual Information (MI), while at the same time minimizing a global energy function. The global energy function relies on complete disparity map estimates, and therefore, requires that the input image pair be rectified, which does not work when using native fisheye image pairs.

A variety of more recent stereo matching approaches have relied on neural networks. Networks trained for correspondence prediction have been proven to perform better [61, 64, 119, 121] than traditional methods, and even networks trained on synthetic data have performed well on real-world data [64]. Convolutional neural networks (CNNs) have become increasingly popular for learning both high-level vision tasks (e.g., semantic segmentation, object detection, classification) [26,35,50,80] and low-level vision tasks (e.g., correspondence and optical flow prediction) [14, 61, 64, 119].

The works in [121] and [119] use CNNs to compute matching costs between image patches using binary cross-entropy and SGM [121] or a hinge-based loss [119]. They were able to show that learned image features can outperform manually-designed features in assessing patch similarities. The work in [61] follows [119, 121], but they instead learn a probability distribution over disparity values, thereby capturing correlations between disparities and avoiding the task of independently processing patches. All three of these approaches, however, assume the stereo image pair has been rectified, and by extension, all image patches can be assumed to lie in the same image plane. The work in [64] presents a deeper CNN to predict disparity maps given an input stereo image pair. Unlike previous work [61, 119, 121], they train their network end-to-end, with the inputs being the stereo images and the output being a predicted disparity image, and they also assume the inputs are rectified stereo images from rectilinear cameras. Our work with fisheye stereo closely follows the work in [119], which uses rectilinear stereo. We adapt their model architecture and retrain it on image patches extracted from the new database.

2.3.2 Stereo and Monocular Depth Estimation using Fisheye Images

There has been no previous work in developing a method to estimate monocular fisheye depth. Thus, we briefly discuss past work in estimating depth using stereo fisheye images and the constraints of those systems. Past work exploring fisheye stereo has typically involved fisheye correction, or some form of spatial distortion removal, followed by stereo rectification [15,23,24,34,45,70, 88,116,123]. The variation among fisheye lens models is wide, but they share the common advantage of having wider FoVs than rectilinear lenses, so each is able to capture more information. However, in fisheye stereo applications, after the removal of fisheye, the transformed views are usually cropped so that the new images looks rectilinear, which results in a loss of information at the FoV boundaries. In cases where the 'undistorted' views are not cropped, the edge content may still suffer from horizontal distortions and lower resolution due to spatial spreading of the original fisheye pixels.

Few methods do not remove fisheye distortion prior to stereo computation [44,68,69,76], but the work in [44] used stereo images that they designed to be densely textured, so they could then effectively apply a correspondence method based on template-matching. The authors in [68,69,76] take an approach based on an analytical model of the fisheye lens and the stereo camera geometry. They perform stereo matching along computed epipolar curves, which provides an additional matching constraint in any correspondence-prediction algorithm. Additionally, [68,69] use a stereo configuration of fisheye cameras with parallel vertical camera axes, and the points they attempt to match are primarily the tops of buildings, which have very large depths and do not suffer from widely disparate representations within a stereo pair.

All previous approaches to the challenging fisheye stereo problem have required imposing constraints, e.g., on the stereo configuration itself, the search space, and so on. A patch-based method that does not rely on stereo data and that requires only computation of NSS features is undoubtedly much more flexible and generalizable.

Chapter 3

LIVE Surround-View Database[1]

We now introduce the LIVE Surround-View (LIVE SV) database that was initially built for automotive surround-view, but can be used in a wide range of stereo or monocular depth estimation methods as well. We first presented this database in [76], and instructions to download the database can be found at `https://github.com/janicepan/live-sv/`.

Surround-view is a common feature of Advanced Driver Assistance System (ADAS) technology and uses four to six outward-facing fisheye cameras placed on the front, right, rear, and left sides of a vehicle [13,33,106,117,122]. By exploiting the wide Field-of-View (FoV) of fisheye lenses, such a configuration allows for the generation of a complete 360^o birdseye-view image, also referred to as the *surround-view* or *top-view* output. An example set of synthetically generated fisheye images along with a "true" simulated SV image are shown in Fig. 3.1.

Because there are currently no surround-view or fisheye stereo databases,

[1]Content presented in this chapter was first published in [76]. Co-authors were Martin Mueller, Tarek Lahlou, and Alan C. Bovik. Janice S. Pan did the work presented in [76] of generating the database while following the guidance and feedback provided by her co-authors.

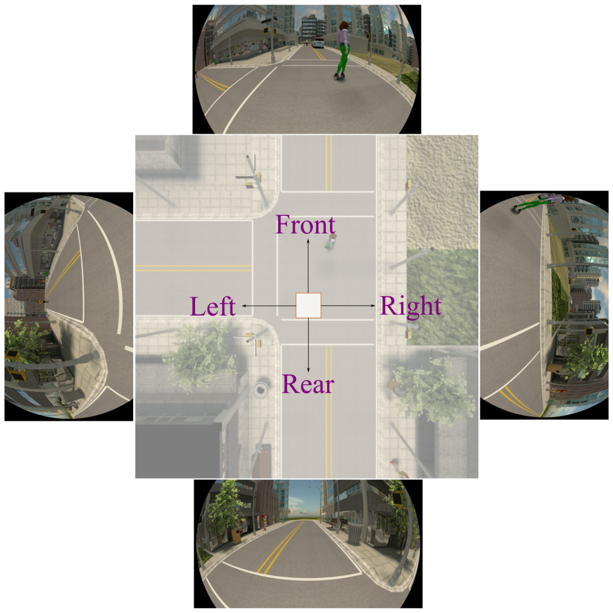


Figure 3.1: Example set of fisheye captures and the associated true SV.

and gathering enough real data for a machine-learning-based approach is infeasible, we built a large 3D synthetic database initially to support data-driven approaches to the SV and fisheye stereo problems. We built a city scene in Blender [6] using SceneCity [12], which provides the tools to construct a single city with realistic roads and buildings. An overhead shot of the city we designed is shown in Figure 3.2. Blender is also compatible with Python scripting, which enables automation in setting up and capturing scenes.

Data-driven approaches in computer vision are proving to outperform state-of-the-art methods [115]. We captured our database using a photo-realistic city scene, because we wanted our data to model real-world geometry, as opposed to past work that used synthetically generated random scenes,

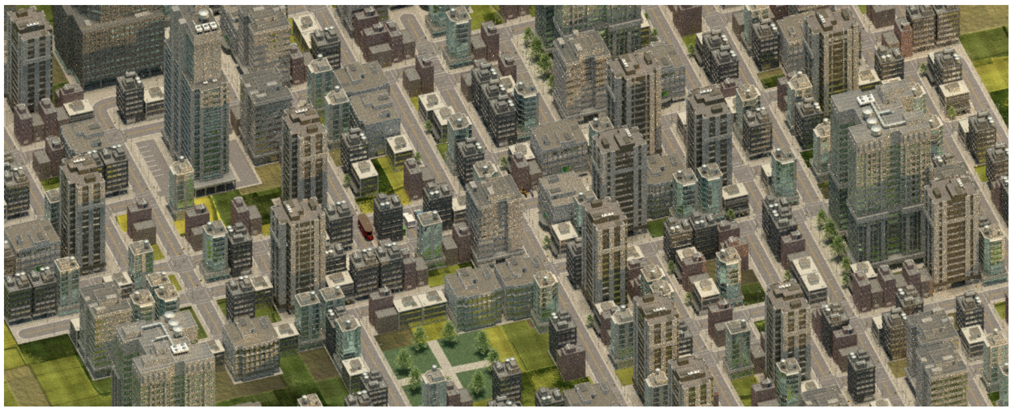


Figure 3.2: Overhead screen-capture of the city designed using SceneCity [12].

e.g., [14, 64]. We detail the camera parameters and configurations used to capture this dataset, which will make clear the variety of applications in which LIVE SV may be useful.

3.1 Blender city

The city we built using SceneCity is comprised only of buildings and streets, including road surfaces, sidewalks, street lights, signal lights, and trees. To add variety and to create scenes more representative of a true city, we populated the city in a structured and realistic manner. We added vehicles, pedestrians, and sidewalk furniture (i.e., benches and trash cans) in two steps: (1) populating the *roads* with vehicles and pedestrians and (2) populating the *sidewalks* with pedestrians and sidewalk furniture.

To obtain the 3D models with which to populate the city, we used Archive3D [1], a large online database containing tens of thousands of 3D models of people, furnishings, appliances, animals, plants, buildings, doors, windows, and more. All models on Archive3D, however, were contributed by different artists, so while there is a wide variety of looks, poses, and styles for any object category, there were also a lot of inconsistencies in terms of Blender compatibility and rendering capabilities. All models are also different sizes, so it was infeasible to automatically import, scale, and position them. Each model we chose to use had to be individually imported, and for most, we also had to manually apply colors and/or textures to different parts of the model. All objects we use to populate the city are shown in Figure 3.3.

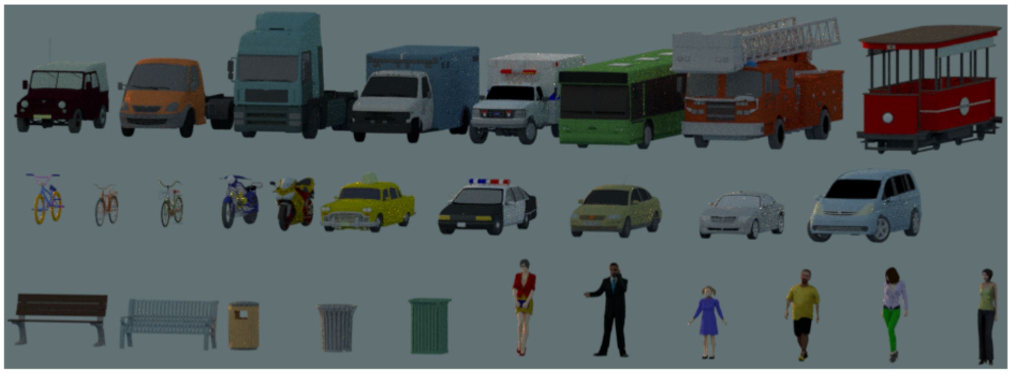


Figure 3.3: Objects added to the city scene built with SceneCity [12].

The collection of added objects includes 16 vehicles (1 trolly, 1 firetruck, 1 bus, 1 ambulance, 3 trucks, 5 cars, 2 motorcycles, and 3 bicycles), 6 pedestrians, 3 trashcans, and 2 benches. Road objects in SceneCity are all a 10m $\times$ 10m with sidewalks, street lights, and signal lights. There are 5 road types (Figure 3.4): straight, turn, T-intersection, 4-way intersection, and deadend. To maintain realism in the scene, we added all objects with a structured randomness. To add a vehicle to a random road, we determine the location and orientation of the road, randomly pick the side of the road on which to place the object, and translate and rotate the vehicle so it faces in the correct direction of traffic flow. When adding a pedestrian to a random road, we randomly rotate it about the z-axis, thereby not constraining them to face in any particular direction. Since sidewalks are located on different sides/corners for each road type, to be able to easily and automatically populate the sidewalks, we decided to only add objects to straight roads, which make up the majority of the road network. Sidewalks on the straight roads simply run down the two sides, so computing positions for objects is straight-forward. Additionally, each of the two sidewalks that borders straight roads are divided in half by street light poles, so when populating the sidewalks, we just add a new object to each of the 4 sidewalk segments. Details for adding objects are as follows:

1. Randomly select 50% of all the road segments to populate. To the first 70% of these roads, add a randomly-selected vehicle. To the rest, add a randomly-selected pedestrian.

2. Randomly select 33% of the all straight roads to populate their sidewalks. For each for the four sidewalk segments, randomly select a pedestrian, trash can, or bench, and translate and rotate the object so it faces the middle of the road (a constraint primarily for the benches).

By adding objects in this manner, we believe we have created enough variation and realism so the database can benefit both the trained network and the SV visualization.

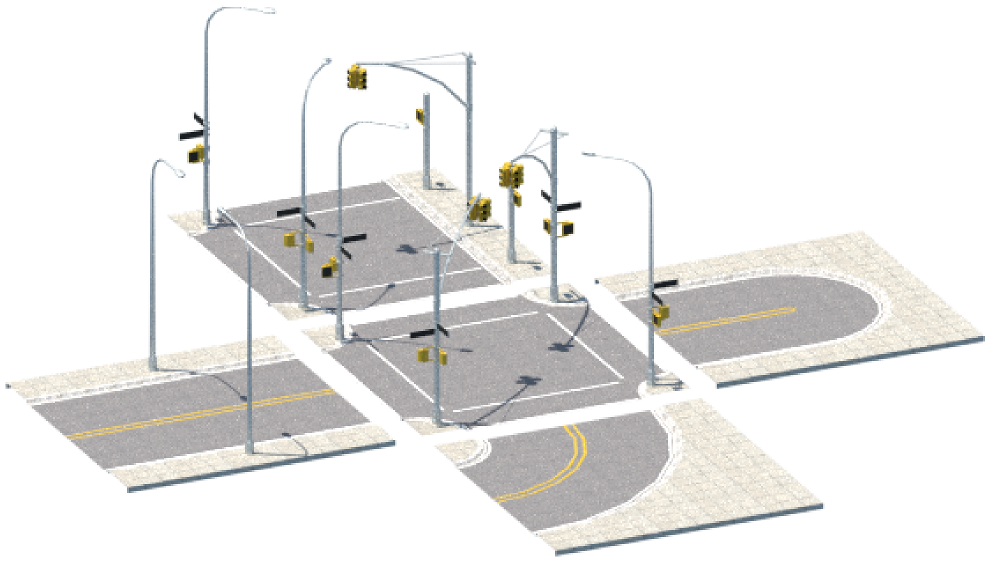


Figure 3.4: SceneCity [12] road types: 2 intersections, 1 straight, 1 turn, 1 deadend. These pieces are duplicated and combined to create the entire road network.

3.2 Data collection

To determine positions within the city from which to capture data, we manually selected 222 locations in the city that have varying concentrations of surrounding road and sidewalk objects. Additionally, we ensured that this set of capture locations contains several sets of contiguous coordinates, to provide some temporally continuous realistic road/driving data, which can help in developing temporal SV methods. In Figure 3.5, we show two examples of some of the selected locations. Selecting the locations such that they form sequences also allows us to gather different perspectives of the same scene, which helps increase the variation in our data.

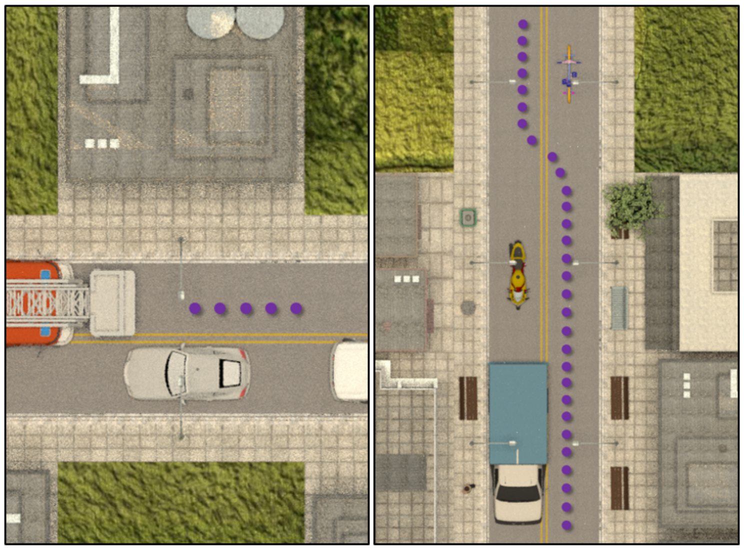


Figure 3.5: Two example sequences of capture locations. The sequence on the left contains 5 contiguous locations, and the sequence on the right contains 30. At each purple dot will be the set of four SV fisheye cameras to capture the scene from that location. Above each purple dot will also be a rectilinear camera to capture a true virtual top-view image.

For every captured fisheye and true top-view image, we also capture a depth map, which specifies for each pixel in the image the Euclidean distance between the 3D world point and the 3D location of the camera center. This measurement, computed in Blender, provides ground-truth depth for every scene. In gathering depth data, we captured each image three times with three different render seeds, and then combined them via a `min` filter. Such processing is necessary, because the Blender rendering engine we used, Cycles, works by projecting light off surfaces, thereby mimicking the physical properties of light. However, Cycles is error-prone in projecting light from the various light sources in the scene and can create burnt-out pixels, which incorrectly have infinite value in the depth map and appear as little bright spots, which we refer to as *fireflies*. Figure 3.6 shows, for an example scene, the depth map captured using a single render seed along with the depth map that results from combining three depth maps captured using three different render seeds. By combining maps captured with different seeds, we are able to eliminate almost all occurrences of these erroneous depth measurements.

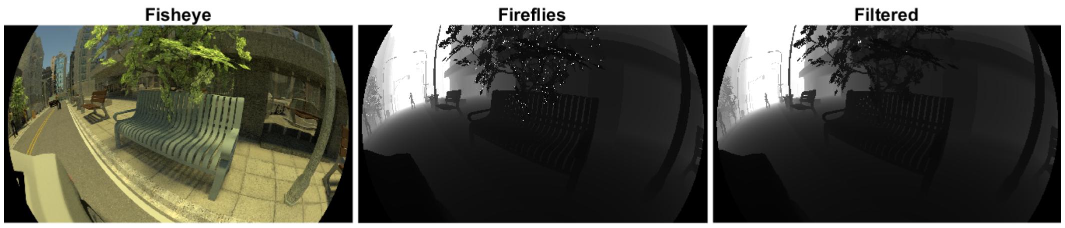


Figure 3.6: By capturing the same scene using different render seeds, we can filter out the noise inherent in all captures: (left) the captured fisheye image; (middle) the corresponding depth image captured using one render seed value; (right) the result after combining the depth maps captured with three different seeds. The 'fireflies' (infinite luminance values) appear at different pixels, and by using a `min` filter, most or all can be removed to generate a cleaner depth map.

In Figure 3.7, we show seven surround-view image sets from the database. Each set contains the four fisheye captures, their corresponding fisheye depth maps, as well as a true SV, i.e., birdseye-view, capture. Due to the range of depths within each scene, to effectively display the depth maps, we only show depths in the range [0,25] meters.

3.2.1 Multiangle captures

In addition to capturing the four SV fisheye images and a true virtual top-view image at each scene location, we also expanded our database to include fisheye images captured at different angles and their corresponding depth maps. Figure 3.8 shows the new angles (in red) by which we will rotate each fisheye camera to expand our database. Each camera was rotated an angle $\theta = \{60^o, 45^o, 30^o, 0^o, -30^o, -45^o, -60^o\}$ about its vertical axis (normal to the ground plane). By gathering data at so many different angles, we are able make our database useful in developing methods for solving other multi-view computer vision tasks. With the additional data, we can also explore fisheye stereo with different degrees of divergence between the camera axes.

The set of captures from when each of the four cameras is at its own *default* 0^o-rotation is the SV configuration, and the adjacent camera pairs in this configuration correspond to the orthogonal fisheye stereo case. Figure 3.9 shows a sample set of images captured at each of these seven angles. By using stereo pairs with different combinations of rotations about their vertical axes, we can control the degree of stereo toe-in or toe-out to build a generalizable fisheye stereo engine. For instance, a pair of adjacent cameras with the left capturing at -45^o and the right capturing at $+45^o$ (refer to Table 3.1 for possible camera pairs) can be used to represent the parallel camera axes case. In other words, with the left/right camera pair being any of the combinations listed in Table 3.1, if the left image was captured at its -45^o orientation, and the right was captured at its $+45^o$ orientation, then the angle between the camera axes is 0^o, i.e., they form a *parallel* fisheye stereo pair.

Table 3.1: Combinations of SV camera pairs that form a stereo pair

Left stereo camera	**Right stereo camera**
Front	Right
Right	Rear
Rear	Left
Left	Front

3.2.2 Data augmentation

The proposed database includes 222 unique manually selected scenes captured within the city, each of which was captured 21 times by the four fisheye cameras and true birdseye-view camera, and each capture has an associated ground-truth depth image, also acquired in Blender. Each of the 21 captures is a different augmentation, chosen by randomly rotating about the vertical axis the 5-camera configuration, which includes the four fisheye cameras and an overhead rectilinear camera to capture a true virtual SV image. The data was also diversified by randomly applying different lighting conditions, shown in Figure 3.10 to the scene. Overall, the database includes 4,662 surround-view image sets, each of which includes the front, right, rear, left, and overhead captures (see Figure 3.1) along with corresponding depth maps for each captured image. So that users of the database can develop and test fisheye stereo methods, the database contains $18,648$ unique stereo pairs. The true SV image is only used in this work to evaluate the performance of our SV distortion correction method.

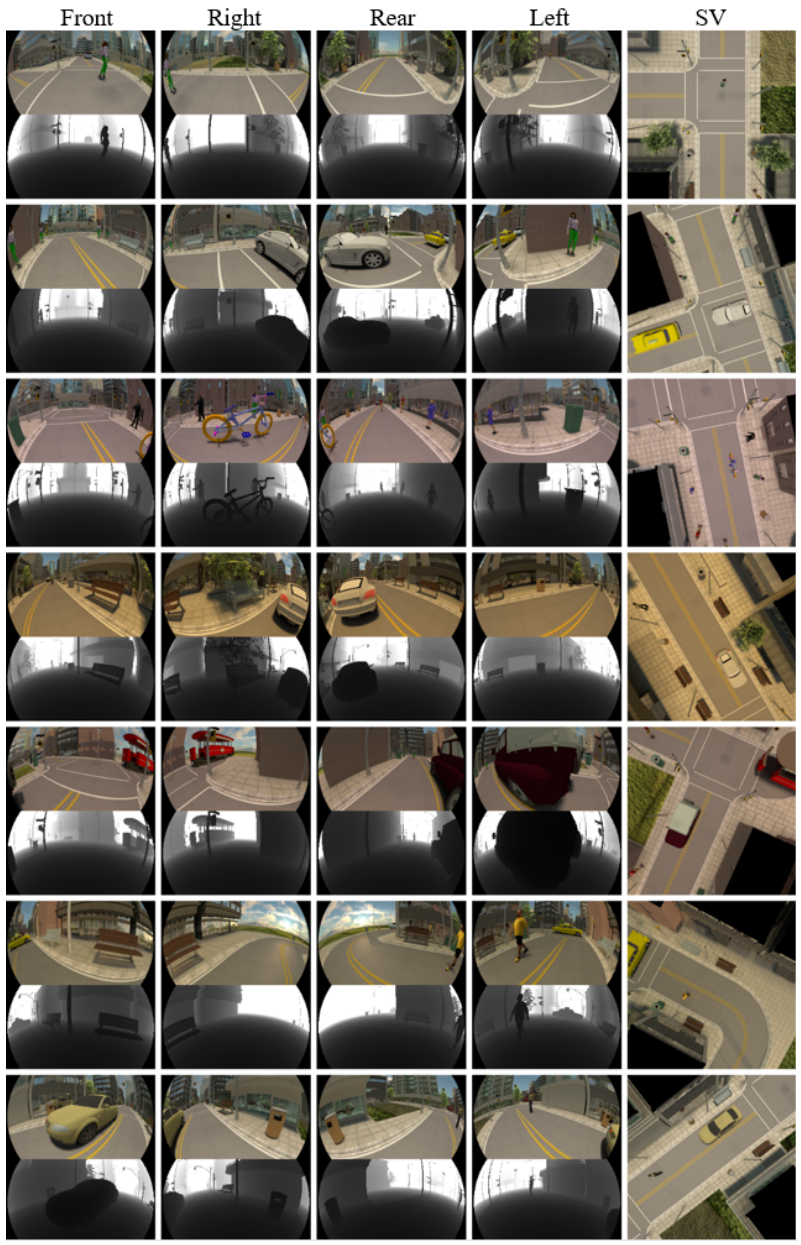


Figure 3.7: Example scenes from the database. Each set of images includes four fisheye images, their corresponding depth maps, and a true birdseye view capture. Note: for display, only depths <25 meters are shown.

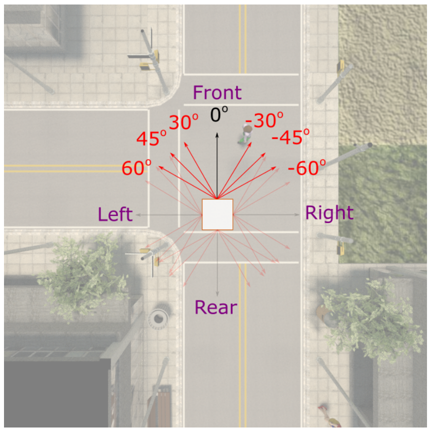


Figure 3.8: New camera angles (in red). Images captured at 0^o correspond to the traditional SV camera configuration, i.e., orthogonal fisheye stereo. To be clear, the front camera's 0^o position is towards the top edge of the page, whereas the right, rear, and left default 0^o positions are towards the right, bottom, and left edges of the page, respectively.

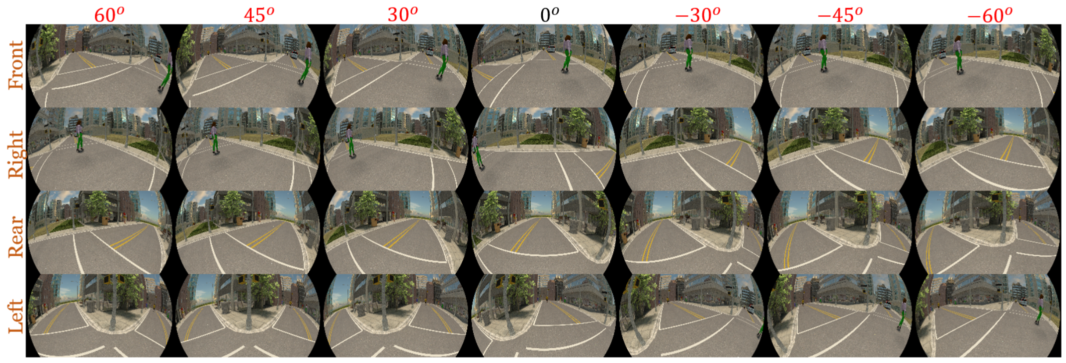


Figure 3.9: Example of an expanded set of fisheye captures for one scene. The camera labels are on the left, and each column displays the images captured when each camera is rotated about its vertical axis from its default 0^o position.

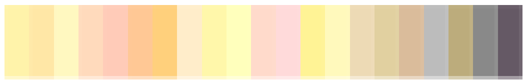


Figure 3.10: Ambient lighting colors

Chapter 4

Luminance and Depth NSS

Before discussing our proposed MDE method, we must first discuss natural scene statistics (NSS) models and introduce a new bivariate correlation model, because these NSS features are integral to our depth-estimation approach. One hypothesis of vision science is that the sensory systems have become, through evolution and adaptation, matched to the statistical properties of the signals to which they are exposed [2, 82, 111]. When attempting to solve the MDE problem, the only available data from which to predict depth is from single-view pixel data. Thus, we begin by exploring the statistical relationships that exist between the bandpass luminance coefficients of images and their corresponding depths. We develop a model that is able to capture the correlations between spatially neighboring luminance values across multiple scales. We apply the bivariate model introduced in [99, 103] to the spatial domain, complete the correlation model, and present it in a closed form. Then in Chapter 5, we use features derived from the new correlation model, along with established univariate and bivariate luminance NSS features, to train a DNN to predict depths from monocular image patches.

4.1 Univariate NSS Model

In the 1980s, Ruderman [84] observed interesting outcomes that arose by applying a simple process of image modification. After digitizing a large set of outdoor photographs, he found that subtracting estimates of the local mean luminances from the pictures, then further processing by dividing by the estimates of local variance (an example of a 'divisive normalization transformation', or DNT), has a decorrelating and gaussianizing effect on the image data. The DNT process is perceptually significant and is a reasonable approximation to the nonlinear behavior of retino-cortical neurons [110]. We will start by reviewing simple spatial domain NSS models, beginning with the univariate model detailed in [66, 67]. Given an image I, define the normalized luminance values:

$$\hat{I}(i,j) = \frac{I(i,j) - \mu(i,j)}{\sigma(i,j) + C}, \tag{4.1}$$

where i, j index row and column pixels, respectively, C is a normalization constant that stabilizes the quotient when σ is small, and

$$\mu(i,j) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} w_{k,l} I_{k,l}(i,j), \tag{4.2}$$

and

$$\sigma(i,j) = \sqrt{\sum_{k=-K}^{K} \sum_{l=-L}^{L} w_{k,l} (I_{k,l}(i,j) - \mu(i,j))^2}, \tag{4.3}$$

where, as in [66], we use a patch sampling window $w = \{w_{k,l} | k = -K, ..., K, l = -L, ..., L\}$ that is a 2D circularly-symmetric Gaussian weighting function sampled out to 3 standard deviations and rescaled to unit volume. In the experiments, we use $K = L = 3$, but operate over multiple scales.

Although in principle, the empirical distributions of the DNT luminance coefficients should be Gaussian-shaped, we instead fit the histograms with the zero-mean generalized Gaussian distribution (GGD) function:

$$f(x; \alpha, \sigma^2) = \frac{\alpha}{2\beta\Gamma(1/\alpha)} \exp\left(-\left(\frac{|x|}{\beta}\right)^{\alpha}\right), \tag{4.4}$$

where

$$\beta = \sigma\sqrt{\frac{\Gamma(1/\alpha)}{\Gamma(3/\alpha)}}, \tag{4.5}$$

and

$$\Gamma(a) = \int_0^{\infty} t^{a-1}e^{-t}dt, \quad a > 0. \tag{4.6}$$

The GGD is parameterized by α and σ^2, which represent the shape and variance, respectively, of the distribution. We use the maximum-likelihood method detailed in [89] to estimate α and σ^2. One of the databases that we use is the large new LIVE Surround-View (SV) Database [76] (Chapter 3), which contains over $130,000$ pairs of synthetically-generated naturalistic color images and ground-truth depth maps. We have found that the parameters of the best

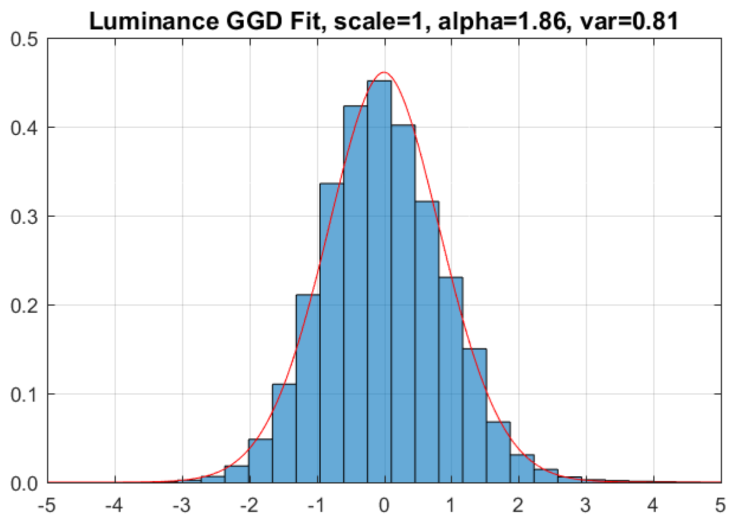


Figure 4.1: GGD fit to bandpass, divisively normalized luminance histogram from the LIVE SV Database [76].

GGD fits to the DNT luminance coefficients of LIVE SV support the use of natural picture statistics models on the images in the LIVE SV database. As shown in Figure 4.1, the distributions of the normalized luminance coefficients strongly tend towards a unit normal Gaussian [66, 84].

The LIVE SV database was originally generated for the purpose of exploring surround-view driver assistance imaging systems, where the feeds from four fisheye cameras placed around a vehicle are combined to generate a top-down view of the vehicle and its surroundings. The SV database, contains fisheye stereo pairs and their corresponding depth maps, which are also subjected to the fisheye distortion. The spatial distortion can be removed to obtain typical rectilinear photographic images of city scenes, which include pedestrians, bikes, motorcycles, a variety of vehicles, and other realistic content.

We performed this distortion removal step and use the resulting non-fisheye images in developing both our bivariate NSS model (Section 4.2) and our monocular depth estimation model (Chapter 5). We will refer to the set of non-fisheye data (images and co-registered depth maps) within this dataset as 'LIVE SV' and the set of fisheye data within this dataset as 'LIVE SV-F'. We now focus on using the LIVE SV data and will only use the fisheye data beginning in Chapter 6.

4.2 Bivariate NSS Model

Our bivariate NSS model extends the work done in [99, 103], which explored the NSS of spatially adjacent directional bandpass (wavelet) responses. We use a simpler isotropic bandpass/DNT model. As with [99, 103], we use a bivariate generalized Gaussian distribution (BGGD) to model the pairwise statistics of pixels having four relative orientations, separated by distance d, as shown in Figure 4.2: the horizontal (0^o), right diagonal (45^o), vertical (90^o), and left diagonal (135^o).

The bivariate generalized Gaussian distribution is:

$$f(\mathbf{x}; \mathbf{M}, \alpha_b, \beta_b) = \frac{1}{|\mathbf{M}|} g_{\alpha_b,\beta_b}(\mathbf{x}^\top \mathbf{M}^{-1} \mathbf{x}), \tag{4.7}$$

where $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{M}$ is a 2×2 covariance matrix, α_b and β_b are scale and shape parameters, respectively, and $g_{\alpha_b,\beta_b}(\cdot)$ is:

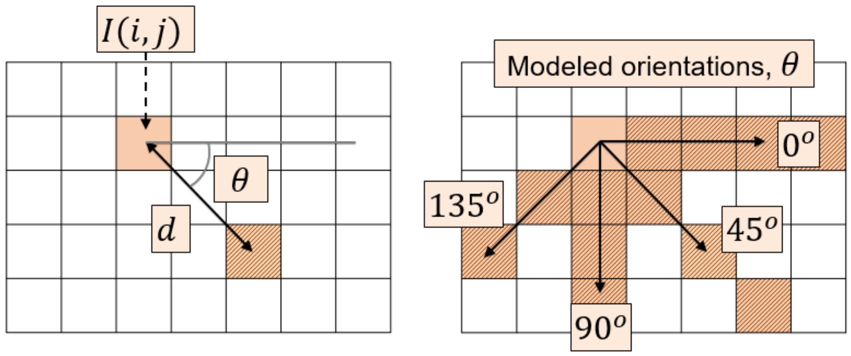


Figure 4.2: Visualization of distances and orientations between pairs of pixels on which we model bivariate statistics. Left: distance is measured in pixels, while orientation is measured clockwise from the positive horizontal. Right: the four orientations considered in our bivariate statistical model.

$$g_{\alpha_b,\beta_b}(y) = \frac{\beta_b \Gamma(1)}{\left(2^{\frac{1}{\beta_b}} \pi \alpha_b\right)^{\frac{1}{2}} \Gamma\left(\frac{N}{2\beta_b}\right)} e^{-\frac{1}{2}\left(\frac{y}{\alpha_b}\right)^{\beta_b}}, \quad y \in \mathbb{R}^+. \tag{4.8}$$

The elements of matrix $\mathbf{M}$ are pairwise covariances, for which we seek a simple closed-form correlation model so that the overall second-order statistical model, described by (4.7) and (4.8), is also of closed form. Figures 4.3 and 4.4 plot the correlations against the distance between the pixels for four different orientations, on two different databases. Figure 4.3 was generated on the LIVE Color+3D Database Release 2 [102], which contains 98 sets of high-definition (1920×1080) stereo color image pairs with co–registered dense ground-truth depth maps. The images in this database are pristine, natural images, i.e., with minimal distortion. The depth data was captured using a RIEGL VZ-400 terrestrial range scanner calibrated to the camera.

As previously mentioned, we also utilized the LIVE SV Database [76],

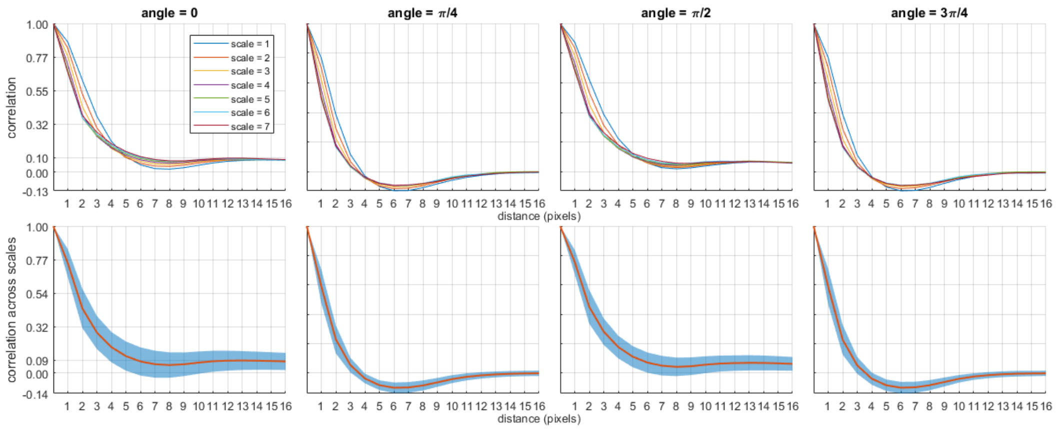


Figure 4.3: Empirical luminance correlations computed on the LIVE Color+3D Release 2 Database. Top row: correlation as a function of distance for (from L to R) $\theta = \{0, 45^o, 90^o, 135^o\}$ and 7 different scales. Bottom row: correlation functions averaged over scales showing confidence bands.

because of its size, which greatly facilitates the training of deep models, and the guaranteed accuracy of its ground-truth range measurements. Large amounts of ground-truth co-registered luminance and depth data are very difficult to obtain, and this dataset contains over $130,000$ pairs of naturalistic color images and corresponding depth maps. Since the data was generated synthetically, it contains perfectly calibrated co-registered luminance-depth data. The correlation plots generated using this synthetic data nicely agree with those computed on pristine images captured by high-definition cameras, which, along with the univariate GGD fits (Figure 4.1), supports the use of LIVE SV for developing MDE models of natural photographic + depth images.

We have found that a difference-of-exponentials can be used to accurately model the empirical correlation values:

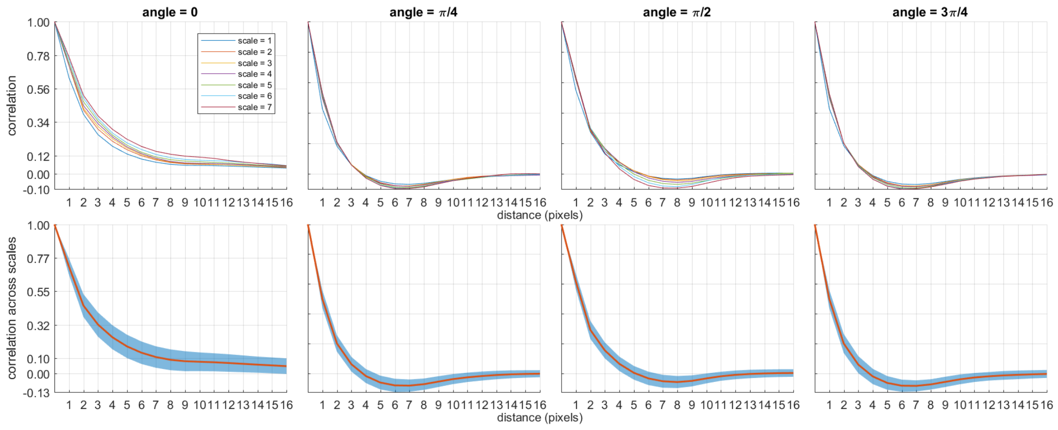


Figure 4.4: Empirical luminance correlations computed on the LIVE SV Database. Top row: correlation as a function of distance for (from L to R) $\theta = \{0, 45^o, 90^o, 135^o\}$ and 7 different scales. Bottom row: correlation functions averaged over scales showing confidence bands.

$$\rho(d) = 2\mathrm{e}^{ad} - \mathrm{e}^{abd}, \tag{4.9}$$

where d is the pixel separation, a controls the decay steepness (and thus, the location of the dip that is evident in the plots), and b controls the width of the dip (or lack thereof). Figure 4.5 shows possible shapes the model (4.9) can exhibit when varying the parameters a and b. In the left plot, a is fixed at -0.5, while b varies, and in the right plot, b is fixed at 0.5, while a varies.

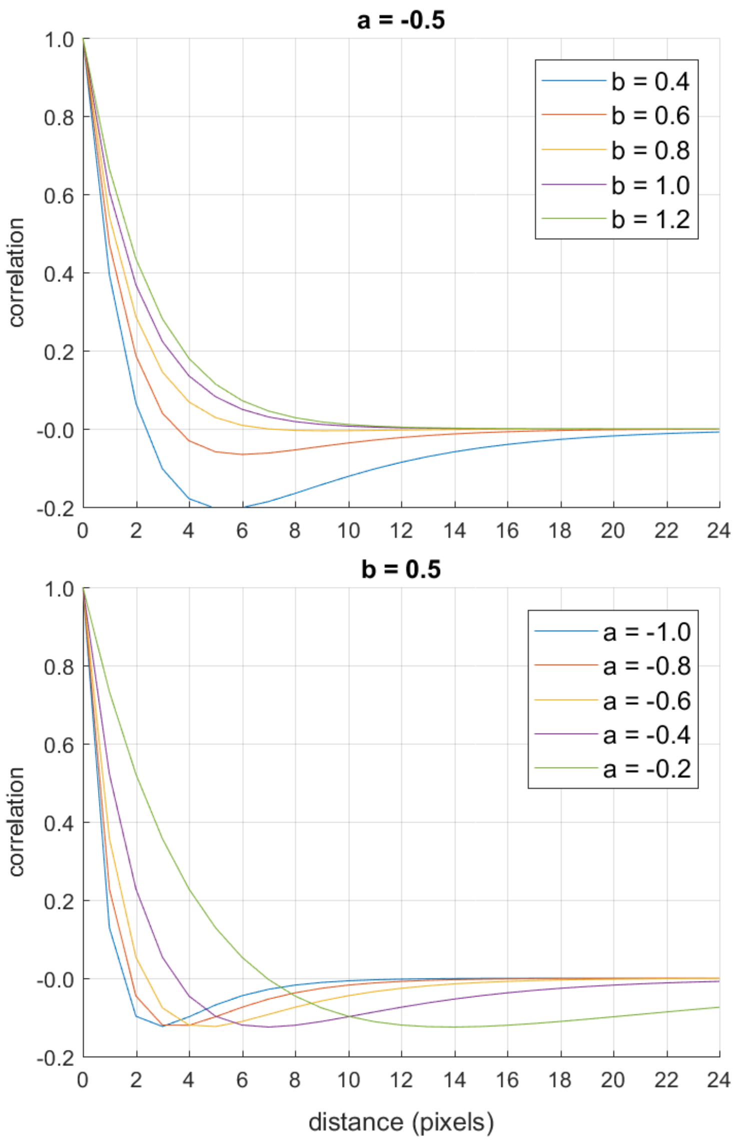


Figure 4.5: Difference-of-exponentials (correlation model (4.9)) with (top) fixed a and varying b and (bottom) fixed b and varying a.

Chapter 5

NSS-Based Monocular Depth Estimation

We now detail our method for predicting depth from monocular images using the univariate and bivariate NSS models discussed in Chapter 4. We use features extracted from the NSS models in a deep neural network (DNN) to predict inverse-range patches from luminance patches. We represent each inverse-range patch using a sparse code, each element of which supplies a weight on each patch within a dictionary of image patches, which was learned from patches sampled from depth perception databases of ground-truth luminance-depth images. We first explain how these patch dictionaries were constructed before discussing the construction of our network while emphasizing the connections and parallels to perceptual models and concepts. In the sections following, we explain the dictionary training and the network design, including the data processing steps, the network architecture, the training and testing details, and the results. We show that our simple network trained using perceptually-relevant features and a perceptually-relevant loss function is able to predict dense depth maps in a manner that is competitive with state-of-the-art methods.

5.1 Sparse Representation for Inverse-Range Maps

It is strongly believed that the HVS extracts statistical elements of visual stimuli to produce efficient representations [2, 82, 111]. Early on, Barlow [4] proposed that early visual processing in the brain acted to remove much of the considerable redundancy present in visual signals by generating statistically independent neural responses. Further, a significant amount of prior work has demonstrated the connection between NSS and sparse representations [3, 19, 72, 73]. In particular, Field [19] showed that simple cells in the primary visual cortex (V1) create sparse representations of the natural world, which is largely made up of regular structures of 3D objects and surfaces. Following Field's paper, there has been extensive work showing that receptive fields similar to those of simple cells can be optimized to produce sparse representations of natural scenes [5, 37, 72, 108, 109]. Regularities in depth are strongly tied to luminance regularities, and depth and luminance statistics can be similarly modeled [38, 58, 59, 95–97, 99, 103].

Su, *et. al.*, [102] postulated that depth maps tend to be composed of simple, regular patterns, and they used a small dictionary of five canonical patterns to represent priors in a Bayesian depth prediction framework. They achieved promising results on predicting dense depth maps. We adopt a similar approach to what was proposed in [102], but with important differences. We use a significantly larger collection of picture + depth data to learn a larger, more descriptive dictionary of inverse-range patterns. Instead of using steerable pyramids, we build dictionaries of divisively-normalized inverse-range patches, computed by applying (4.1) to the inverse-range maps:

$$\hat{I}_{ir}(i,j) = \frac{I_{ir}(i,j) - \mu_{ir}(i,j)}{\sigma_{ir}(i,j) + C_{ir}}. \tag{5.1}$$

Again, in traditional stereo-matching algorithms, the goal is typically to compute disparity, which is inversely proportional to depth (and directly proportional to the camera focal length and baseline of the stereo configuration), so our use of inverse-range is natural in a depth-prediction context. Additionally, the scale of inverse-range values within each map is smaller and more manageable than the potential range of depths.

On each database that we used to develop and test our models, we randomly extracted 200,000 DNT inverse-range patches of size 31×31 across a range of scales. To do so, we first scaled each image to multiple resolutions, the choice of which we discuss in Section 5.2.2, and for each image, regardless of resolution, we extracted patches of constant size 31×31. We then applied least angle regression (LARS) [16], which is a stepwise regression algorithm that, at each step, selects a patch from a collection of possible dictionary elements that has the strongest correlation with the data, and then iteratively selects patches based upon a refitting of residuals to build the dictionary. The candidates for the dictionary patches were initialized using singular value decomposition (SVD) to be the orthogonal basis for $\mathbb{R}^{961}$ (because $31^2 = 961$). In other words, if we have $n \times p$ data matrix A, where n is the number of image patches used to learn the dictionary (200,000), and $p = 31^2 = 961$, then applying SVD gives:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathbf{T}}, \tag{5.2}$$

and the candidate dictionary elements are the rows of $\mathbf{S}\mathbf{V}^{\mathbf{T}}$. By iteratively selecting the strongest patch to include in the dictionary, we obtain a small subset of orthogonal patches that can be used to reconstruct any DNT inverse-range image, thereby removing a significant amount of redundancy in the representation of the input data. LARS is implemented in Python in the `MiniBatchDictionaryLearning` function from the decomposition module within the scikit-learn library [77]. By including patches from multiple scales during construction of the dictionary, we were able to build a reasonably scale-agnostic model.

Using this method, we obtained a set of 64 31×31-sized basis patches, so that any image patch within the database can be represented using a linear combination of these 64 bases. Each patch from each image in the database can thus, instead of being represented by 31^2 pixels, be represented by a sparse 64-length code, each element of which represents a weight for the corresponding dictionary patch. Figure 5.1 shows the learned dictionaries of inverse-range patches on each of the three datasets we tested. We also tried learning dictionaries of different sizes (36 and 81) and found that the smaller 36-element dictionary yielded image reconstructions with significantly greater loss, whereas the 64- and 81-element dictionaries yielded comparable high-accuracy image reconstructions.

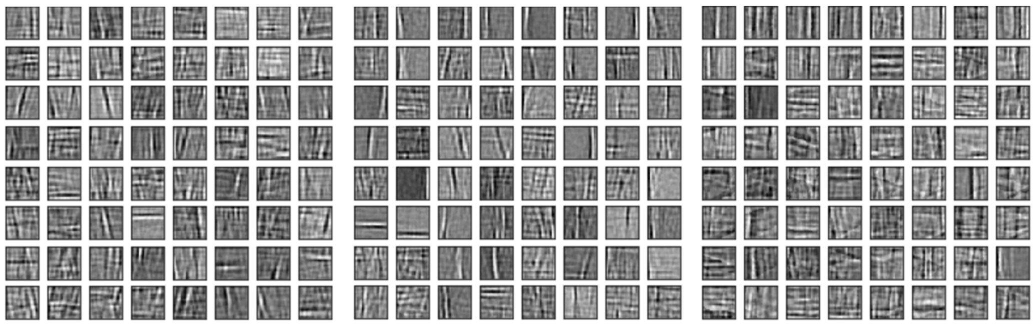
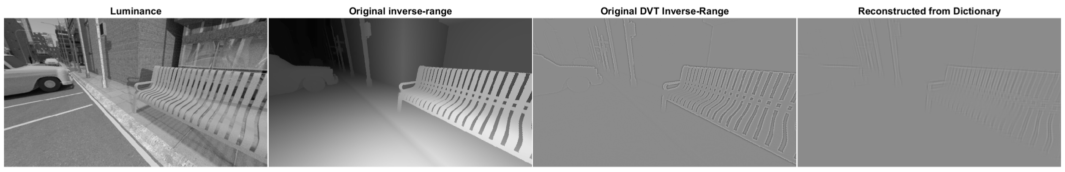

(a) LIVE SV [76] (b) LIVE Color+3D [102] (c) NYU Depth V2 [90]

Figure 5.1: Dictionaries of learned sparse inverse-range patches computed on each of the three tested databases.

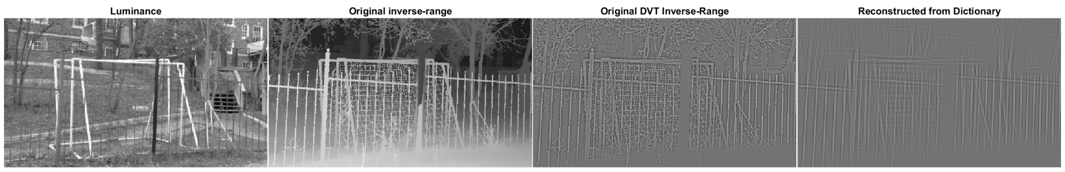
Figure 5.2 shows examples of reconstructions of one DNT inverse-range image from each database. To demonstrate the applicability of the dictionaries to images of different scales, the examples in Figure 5.2 are of varying resolutions. The example from the LIVE SV Database in Figure 5.2a is 360×640, which is 25% of the resolution of the original (720×1280); the example from the LIVE Color+3D Database in Figure 5.2b is 270×480, or 1/16-th the original resolution of 1080×1920; and the example from the NYU Depth Dataset in Figure 5.2c is of full resolution 480×640 (460×620 with some invalid edge pixels removed).

5.2 Method

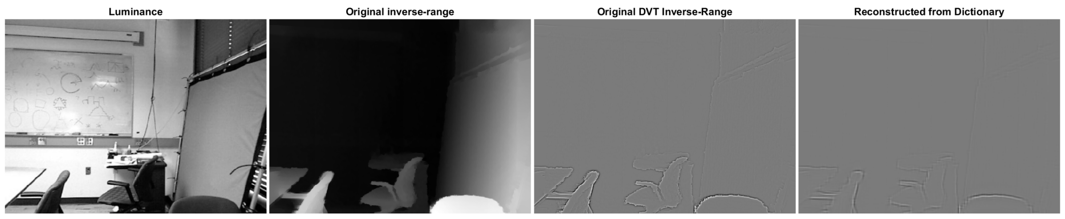
We now explain the components of our inverse-range patch prediction model. We start by describing the model input features in Section 5.2.1. Then we describe each of the relevant luminance + depth datasets, and how we



(a) Example from LIVE SV [76]



(b) Example from LIVE Color+3D [102]



(c) Example from NYU Depth V2 [90]

Figure 5.2: From left to right in each row: Original luminance image; ground-truth inverse-range map; divisively normalized bandpass ground-truth inverse-range; divisively normalized bandpass inverse-range image reconstructed from the dictionary elements in Figure 5.1a

processed each of them to extract the necessary features in Section 5.2.2, We detail the design of the network architecture in Section 5.2.3, and relate how we trained the model in Section 5.2.4.

5.2.1 NSS Features

The goal is to take an input luminance patch that is a projection of part of a 3D scene, and densely predict its corresponding depth patch. We use patches of size 31×31 throughout. From each patch within a luminance image, we extract a 982-length feature vector $\boldsymbol{F}$ comprising:

- DNT luminance coefficients: L_{ij} for $i, j \in 1, ..., 31$,
- Luminance patch mean: μ_L,
- Luminance patch variance: v_L,
- Normalized y-coordinate of the luminance patch: y,
- Luminance GGD parameters: $\alpha_{\text{GGD}}, \sigma_{\text{GGD}}$,
- Luminance BGGD parameters: $\alpha_{\text{BGGD}_k}, \beta_{\text{BGGD}_k}$ for $k = [1, ..., 4]$,
- Luminance correlation model parameters: a_k, b_k for $k = [1, ..., 4]$,

where k indexes the four pairwise orientations $\{0^o, 45^o, 90^o, 135^o\}$, and μ_L and v_L are extracted during the divisive normalization step. The GGD parameters are estimated using the moment-matching approach used in [66] and proposed in [89]; the BGGD parameters are estimated using the moment-matching approach for fitting multivariate generalized Gaussian distributions (MGGD) detailed in [29]; and the correlation model parameters are estimated using the Nelder-Mead simplex algorithm [52], as implemented in the `fminsearch` MATLAB function, to fit the constrained difference of exponentials (4.9) to the empirical correlations, for integer pixel separations ranging from $d = 1$ to $d = 16$ pixels.

The luminance patch mean and variance are computed during the process of divisively normalizing the luminance patch, and the normalized y-coordinate is simply the y-coordinate in pixels divided by the height of the image in pixels. This knowledge of vertical location within the image is a monocular cue, because pixels closer to the top generally correspond to world points with larger depths, and pixels closer to the bottom generally correspond to to world points that have smaller depths [102].

5.2.2 Datasets

Next we describe the datasets that we used and explain how the data was processed for both training and testing. On each dataset, we extract and use only the luminance channel, and compute inverse-range as the reciprocal of depth. We selected 20 images from each dataset that we set aside to evaluate the performance of the trained inverse-range prediction network on full images. These images were not included when generating patch data to train and test the network. We also list in Table 5.1 a few database features and processing choices. The variable C_{ir} represents the normalization constant used in (4.1) for inverse-range images, which is required to build the inverse-range dictionary specific to each dataset and to extract the sparse code for generating the training and testing data. The choice of C_{ir} is different for each database, because it depends on the variances of the luminance patches within the databases, which arises from various factors, such as the precision of the data, the complexity of the scenes, and any inherent noise in the data. We manually chose C_{ir} for each database to not overpower the variance in the DNT equation denominator.

Additionally, because we aim to build a scale-agnostic model, we extracted a range of scales of images within each database depending on the original resolution of images in that database. The images in the NYU dataset [90], for instance, are native 480×640, which is much smaller than the native images in both the LIVE SV [76] and LIVE Color+3D [102] databases, so we only extract five scales (including the original) on NYU, as opposed to the seven which we extract from the others.

LIVE SV Database The LIVE SV Database [76] is a large synthetic database with over 130,000 co-registered pairs of color images and ground-truth depth maps. Although all of the images in the database were captured with a fisheye lens, the intrinsic camera parameters are known, so we transformed all the images and depth maps to their rectilinear forms. To generate the training and testing data, we randomly sampled patches from 500 luminance-depth image pairs. In this database, some pixels in the sky or on complex object surfaces (e.g., tree leaves) have infinite ground-truth depth recorded, so they were excluded from the training and testing sets, and only depths less than 50 meters were estimated.

LIVE Color+3D Database Release 2 The LIVE Color+3D Database Release 2 [102] contains 98 sets of stereo color image pairs with co-registered depth maps of outdoor scenes. Because the left and right images and depths in each stereo pair are quite similar in nature, we randomly selected 20 left images for final evaluation and excluded their corresponding right images from both the training and testing data.

NYU Depth Dataset V2 The NYU Depth Dataset V2 [90] contains 1449 pairs of aligned color images and depth maps of indoor scenes. Due to occlusions and measurement limitations, the raw depth maps contain missing values. These missing depth measurements are often in key spatial locations, as they correspond to changes in object boundaries and sharp changes in depth. The authors provide interpolated measurements for missing values, and because it is important that our model be able to learn such geometries in a scene, we used these dense maps to extract training and testing data.

Table 5.1: Database Features and Processing

Database	**Resolution**	**Scales**	$C_{\mathbf{ir}}$	**Depths**
LIVE SV	720×1280	7	0.1	<50m
LIVE Color+3D	1080×1920	7	0.0001	-
NYU Depth V2	480×640	5	0.001	-

5.2.3 Neural Network Architecture

In order to optimize our predictions of inverse-range using NSS features, we built a modular network, motivated by our normalization transformation of the inverse-range images and NSS. If we consider any inverse-range map as being a reconstruction from its DNT form, as in (5.1), then we may formulate the inverse-range-prediction problem as a combination of four smaller problems: first, predicting the DNT patch, the patch mean, and the patch variance, and then combining those three predictions in a reverse divisive-normalization transformation step. We model our network on this formulation, as depicted in the network flow diagram in Figure 5.3a. One module, the ‘code’ block, is designed to predict a 64-length feature vector, which is intended to represent the code used to reconstruct the DNT inverse-range patch from dictionary patch elements. To obtain the inverse-range values from the DNT patch, we also require the patch mean and variance. However, instead of training separate networks to predict mean and variance separately, we use two normalizing scalar (NS) blocks in our network to predict scalar features to be passed to the combination module to predict the final inverse-range patch.

We show the specific architectures of the modules in Figure 5.3b. The code and NS modules use the four fully-connected layers, while the combination module uses the four convolutional layers. The luminance blocks also use the four convolutional layers with an additional fully-connected layer to obtain a feature vector of the appropriate dimensions to input to the intermediate ‘code’ block. We provide details of all layer dimensions for our highest-performing network in Table 5.2, and note that our network contains approximately 880K parameters, which is orders of magnitude smaller than other high-performing models [21, 53, 57].

We tested module architectures of different depths and widths and tried extracting luminance feature vectors of different lengths. We also tested fully-connected architectures, fully-convolutional architectures, and other combinations besides the one represented in Table 5.2, along with options of including residual connections. The chosen output length of the luminance feature-extraction block is 961, as displayed in Figure 5.3a, which indicates that the luminance CNN extracts a feature vector of the same dimensions as the input patch. We found that extracting shorter feature vectors compromised performance. Our final network also does not use any skip or residual connections. We did not find that adding them between fully-connected layers improved performance, and we obtained similar results using an adapted version of ResNet [35] as the combination module, however that network is more complex and takes much longer to train and execute on test data. Importantly, our proposed network requires far fewer parameters than previous high-performing networks designed and trained for monocular depth estimation. Much of the predictive power of the system lies in the use of perceptually relevant features.

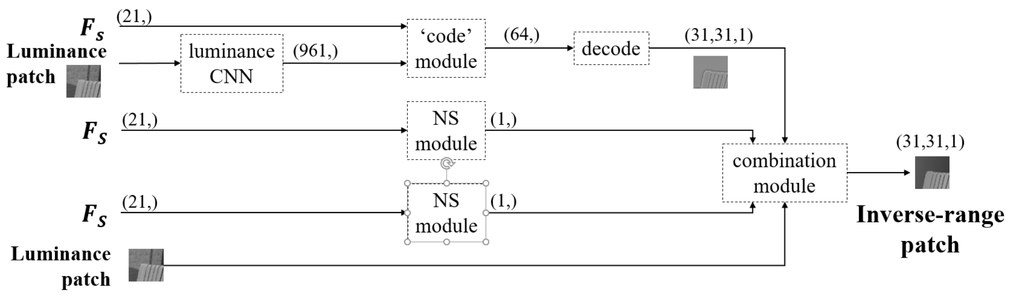
If the features listed in Section 5.2.1 are denoted as F, then F_s is a subset of F that excludes the DNT luminance coefficients. For the NS blocks, the only inputs are F_s, whereas, for the 'code'-prediction module, the DNT luminance patch is first passed through four convolutional layers to extract a

961-length feature vector before being concatenated with F_s to form the input feature vector of length 982. We also conducted experiments where we excluded the NSS features and only trained on luminance patches. In these experiments, the input to each of the three modules was a 961-length luminance feature vector, extracted using a separate CNN (each having the same architecture as the luminance CNN detailed in Table 5.2) for each module. We found that the results suffered significantly when NSS features were excluded from the model.

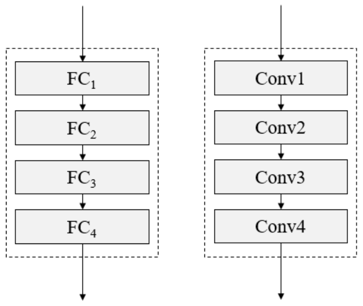
The network begins with these three independent branches to represent the components in the reverse divisive normalization process (5.1), because any image patch can be reconstructed from its DNT, mean, and variance. The 64-length vector resulting from the code-prediction module represents the sparse code, which is used to reconstruct the DNT patch from the dictionary elements. This step is represented by the 'decode' block in Figure 5.3a, which takes a 64-length inverse-range patch code as input and reconstructs an estimate of the DNT inverse-range patch, which is then combined with the outputs of the NS modules in a combination module, yielding the estimate of the final inverse-range patch.

5.2.4 Training

To train the network, which we did for each database separately, we randomly sampled 625,000 luminance-depth patch pairs. We reserved 20% (125,000 pairs) for testing, and used the remaining 80% (500,000 pairs) for training. During the training process, 10% of the training patches were reserved



(a)



(b)

Figure 5.3: (a) Network flow diagram, in which the four-layer fully-connected architecture shown in (b) is used in the code-prediction, mean-prediction, and variance-prediction modules, and the four-layer convolutional architecture shown in (b) is used in the combination module as well as to extract luminance features to input into the code-, mean-, and variance-prediction modules.

Table 5.2: Network Architecture Details

Module	**Input**	**Output**	**Layer Dimensions**			
			FC$_1$	**FC$_2$**	**FC$_3$**	**FC$_4$**
‘code’	(982,)	(64,)	524	262	131	131
NS (‘mean’)	(21,)	(1,)	11	6	3	3
NS (‘variance’)	(21,)	(1,)	11	6	3	3
			Conv$_1$	**Conv$_2$**	**Conv$_3$**	**Conv$_4$**
luminance	(31,31,1)	(961,)	(3,3,4)	(3,3,8)	(3,3,16)	(3,3,32)
combination	(31,31,4)	(31,31,1)				

for validation.

We evaluated a number of loss functions, including the mean squared error (MSE), mean absolute error (MAE), difference of structural similarity (DSSIM), [41,113], and weighted combinations of the three. The non-traditional loss, DSSIM, is based on the Structural SIMilarity index (SSIM), which considers luminance, contrast, and structural similarities between two images, taking a maximum of 1 when the images being compared are identical. To use this index in a loss function, we therefore subtract its value from 1, and in the Keras implementation we used, the DSSIM loss is clipped at 0.5. We found that the MAE preserved finer details, which is to be expected, while a combination loss (MAE-DSSIM) equally weighing MAE and DSSIM produced the best results.

We trained the network end-to-end, while also utilizing losses at the outputs of the ‘code’ and NS modules. For the first half of training, we used a combination loss comprising the overall MAE-DSSIM inverse-range loss as well as three intermediate layer MAE losses:

- the MAE loss between the output of the 'code'-prediction block and the ground-truth sparse code of the DNT inverse-range patch;
- the MAE loss between the output of one of the NS modules and the ground-truth inverse-range patch mean;
- and the MAE loss between the output of the other NS module and the ground-truth inverse-range patch variance.

The weight ratio between the overall loss and each of the three intermediate losses was $1 : 0.01$. The validation loss was computed at the end of each epoch. We used a patience threshold of 10 epochs to determine when to change or stop training, meaning that we monitor the validation loss to determine when it no longer improved (i.e., the validation loss decreased) for 10 consecutive epochs. The first time this event happened, the weights for the intermediate losses were dropped, and only the overall MAE-DSSIM inverse-range patch loss was used for the rest of training, which stopped when we no longer saw improvement in the validation loss for another 10 consecutive epochs. The final model was taken to be the one having the lowest validation loss.

We used this two-tier training scheme to initially gently guide the 'code' and NS modules to predict the quantities required for patch reconstruction from DNT components (5.1). We removed these guiding losses before finishing training, which we noticed had a small impact on the visual results, but systematically improved the quantitative results. We also tried training end-to-end with only the MAE-DSSIM loss, and found that the network sometimes struggled more to accurately predict the inverse-range values of object interiors.

5.2.4.1 Data Augmentation

We also observed that the inverse-range patch variances of natural images are heavily skewed toward zero, because scene depths are predominantly smooth and continuous with discontinuities mainly corresponding to object edges. To avoid biasing our model to predicting inverse-range patches having zero variance, we augmented the training data by sampling the patches based on their variances, and by combining these samples with the training samples. Specifically, we computed a sample weight for each training patch to be its variance over the sum of all training patch variances. We then sampled, with replacement, 500,000 patches, i.e., the same number of patches as in the training set, appended these patches to the original training data, and randomly selected 500,000 patches from this augmented set for training.

In Figure 5.4, we show, for a random sample of LIVE SV patches, the histogram of inverse-range patch variances on the left, where the severity of the bias toward zero is apparent. After resampling, the distribution is still skewed toward zero, but the bias is not as severe, as seen in the histogram on the right in Figure 5.4. By implementing this resampling approach, our training data distribution is still significantly representative of the true data distribution, but we place a greater emphasis on learning inverse-range values on patches having higher inverse-range variance, which are more likely to correspond to points of

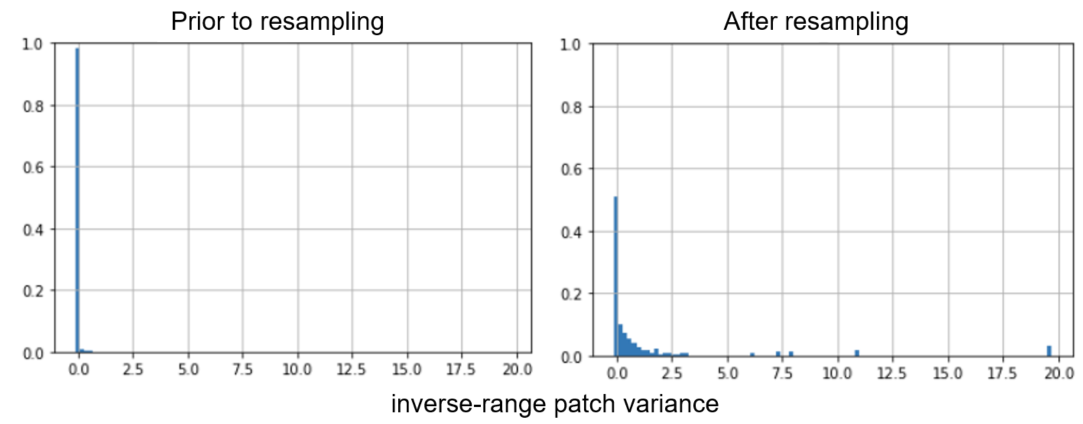


Figure 5.4: Histograms of inverse-range patch variances: (left) before resampling and (right) after resampling using the method detailed in Section 5.2.4.1. Plots are shown for a randomly sampled 100,000 patches from the LIVE SV data, but we observed similar trends on the other databases as well.

interest or object edges. We trained networks with and without this resampling step and found that the predictions resulting from using the resampled training data to be both quantitatively and qualitatively better.

5.3 Results

While the final models are trained to predict inverse-range in a patchwise manner, the true test of each model is how well it can predict dense inverse-range maps on entire images. To evaluate the final models, we used the 20 randomly-selected **evaluation** images from each database that did not contribute any training or testing patches to build the model. As previously mentioned, if the evaluation image was part of a stereo pair from the LIVE Color+3D database, the corresponding image also made no contribution to the sets of training and testing patches, because the image content between the stereo image pairs in that database is almost always nearly identical.

To evaluate performance, we compute the mean absolute error (MAE) and the normalized mean absolute error (nMAE), which is the MAE divided by the ground truth values:

$$\mathrm{nMAE}(G, P) = \frac{|G - P|}{|G|}, \tag{5.3}$$

where G and P are the ground-truth and predicted inverse-range maps, respectively.

In Table 5.3, we show the average and median of each error metric across the 20 evaluation images for each database when the model was trained on patches extracted from images of the same resolution. In other words, for each scale, 650,000 training and testing patches were extracted only from images of that scale. Thus, the scales and image resolutions listed in Table 5.3 correspond to both the model and evaluation data, and the model trained for each scale can be considered to be *scale-aware*. The results in Table 5.4, on the other hand, were obtained from models trained on patches extracted across all scales, i.e., 7 scales on both of the LIVE databases and 5 scales on the NYU dataset. These models can therefore be considered as *scale-agnostic*, and the scale and image resolutions listed in Table 5.4 correspond only to the evaluation data for which the results are listed, because a single multi-scale model was trained for each database.

The results reported in Tables 5.3 and 5.4 are errors, so smaller numbers indicate better performance. We considered the normalized MAE when comparing model performances, since it weighs errors based on the ground truth values. Therefore, major differences between depth distributions in scenes from different databases are normalized. Generally, the multi-scale models, which were trained on data across all scales, performed slightly worse than single-scale models trained on scale-specific data. However, the difference was small, and the multi-scale results strongly support the efficacy of using the scale-invariant NSS to learn scale-agnostic networks. We also observed that for any given database, models trained on higher-resolution data did not necessarily perform better in terms of nMAE. Similarly, models trained on data at different scales also did not necessarily perform better on higher-resolution data. In fact, there was no obvious relationship between resolution and performance, which supports the generalizability of the model and features.

Table 5.3: Results from Single-Scale Models

Database	**Scale Index**	**Image Resolution**	**Average MAE**	**Median MAE**	**Average nMAE**	**Median nMAE**
LIVE SV	0	720×1280	0.4251	0.3736	0.3347	0.3221
	1	509×905	0.4166	0.4050	0.3176	0.3014
	2	360×640	0.3986	0.3739	0.3204	0.3174
	3	255×453	0.4094	0.3750	0.3431	0.3470
	4	180×320	0.3820	0.3559	0.3096	0.3190
	5	127×226	0.4016	0.3543	0.3328	0.3191
	6	90×160	0.3804	0.3610	0.2934	0.2941
LIVE Color+3D	0	1080×1920	0.0432	0.0394	0.6060	0.5207
	1	764×1358	0.0414	0.0352	0.5470	0.4916
	2	540×960	0.0379	0.0314	0.5277	0.4627
	3	382×678	0.0396	0.0319	0.5394	0.4186
	4	270×480	0.0439	0.0310	0.5309	0.4693
	5	191×339	0.0453	0.0308	0.5489	0.4809
	6	135×240	0.0453	0.0342	0.5304	0.4553
NYU Depth V2	0	460×620	0.1500	0.1085	0.3094	0.3083
	1	325×438	0.1460	0.1074	0.3002	0.2951
	2	230×310	0.1402	0.1061	0.2894	0.2898
	3	163×219	0.1343	0.1033	0.2814	0.2810
	4	115×155	0.1260	0.1035	0.2697	0.2790

We also compared our model's performance on the test images of each database to the performance of the model presented by Laina, et. al [53].

Table 5.4: Results from Multi-Scale Models

Database	Scale Index	Image Resolution	Average MAE	Median MAE	Average nMAE	Median nMAE
	0	720×1280	0.4642	0.4341	0.3745	0.3642
	1	509×905	0.4471	0.4228	0.3757	0.3646
	2	360×640	0.4423	0.4100	0.3663	0.3668
LIVE SV	3	255×453	0.4474	0.4167	0.3537	0.3522
	4	180×320	0.4439	0.4263	0.3345	0.3387
	5	127×226	0.4643	0.4181	0.3228	0.3152
	6	90×160	0.4669	0.4671	0.3093	0.2904
	0	1080×1920	0.0497	0.0408	0.7224	0.6231
	1	764×1358	0.0518	0.0387	0.6617	0.5737
	2	540×960	0.0488	0.0340	0.6491	0.5653
LIVE Color+3D	3	382×678	0.0461	0.0321	0.6165	0.5274
	4	270×480	0.0459	0.0323	0.5963	0.5042
	5	191×339	0.0449	0.0341	0.5903	0.5150
	6	135×240	0.0437	0.0337	0.5778	0.5050
	0	460×620	0.1510	0.1107	0.3029	0.3035
	1	325×438	0.1464	0.1087	0.2967	0.2953
NYU Depth V2	2	230×310	0.1433	0.1077	0.2948	0.2886
	3	163×219	0.1383	0.1037	0.2877	0.2934
	4	115×155	0.1328	0.0966	0.2773	0.2816

Although another model [21] has been reported to provide better performance than the Laina model, we could not reproduce the reported results, and the authors of [21] do not provide training code to do so. In [21], the second best performance on the NYU dataset was obtained using Laina, so we used Laina as a benchmark to evaluate our model's performance. The Laina model is trained on full images and predicts complete depth maps, although at a lower resolution. Thus, for each database, we formed the training data using the images from which we extracted patches to train our patch-based model. We also resized the LIVE SV and LIVE Color+3D images to 480×640 to avoid modifying the original model. Additionally, because the model was initially presented to predict range, we trained and evaluated the Laina model on range images and took the inverse of the predictions for the sake of comparison. We tried training the models to predict inverse-range directly, but this approach yielded worse results. We show the performance results of this model along with those from our own scale-agnostic model in Table 5.5.

Table 5.5: With and Without NSS

Database	**Model**	**MSE**		**MAE**		**nMAE**	
		Mean	Median	Mean	Median	Mean	Median
LIVE SV	Pan	0.5538	0.3492	0.4642	0.4341	0.3745	0.3642
	Laina	5.8683	5.3368	1.9381	1.9143	0.6769	0.6672
LIVE Color+3D	Pan	0.0045	0.0028	0.0497	0.0408	0.7224	0.6231
	Laina	0.4941	0.4778	0.6717	0.6602	10.1756	9.7636
NYU Depth V2	Pan	0.0471	0.0221	0.1510	0.1107	0.3029	0.3035
	Laina	0.0360	0.0210	0.1455	0.1225	0.3361	0.3264

We show example predictions computed by both our scale-aware and scale-agnostic models, as well as the predictions produced by the Laina model [53] in Figures 5.5, 5.6, and 5.7. On each database, we chose predictions from two different scales of our models to demonstrate the generalizability of our scale-specific models to be scale-agnostic. (Refer to Table 5.3 or Table 5.4 for scale resolutions.) The Laina model performed quite well on the NYU dataset, however, our models outperformed the Laina model with respect to normalized MAE. The complexities contained in the LIVE SV Database and the insufficient amount of training data in the LIVE Color+3D Database likely contributed to the poor visual results delivered by the Laina model.

5.3.1 Module Outputs

As mentioned, the modular design of the network was motivated by the DNT deconstruction of the inverse-range map. The three components needed to reconstruct any inverse-range patch are the divisively-normalized patch, which can be reconstructed from the 64-length code, along with the

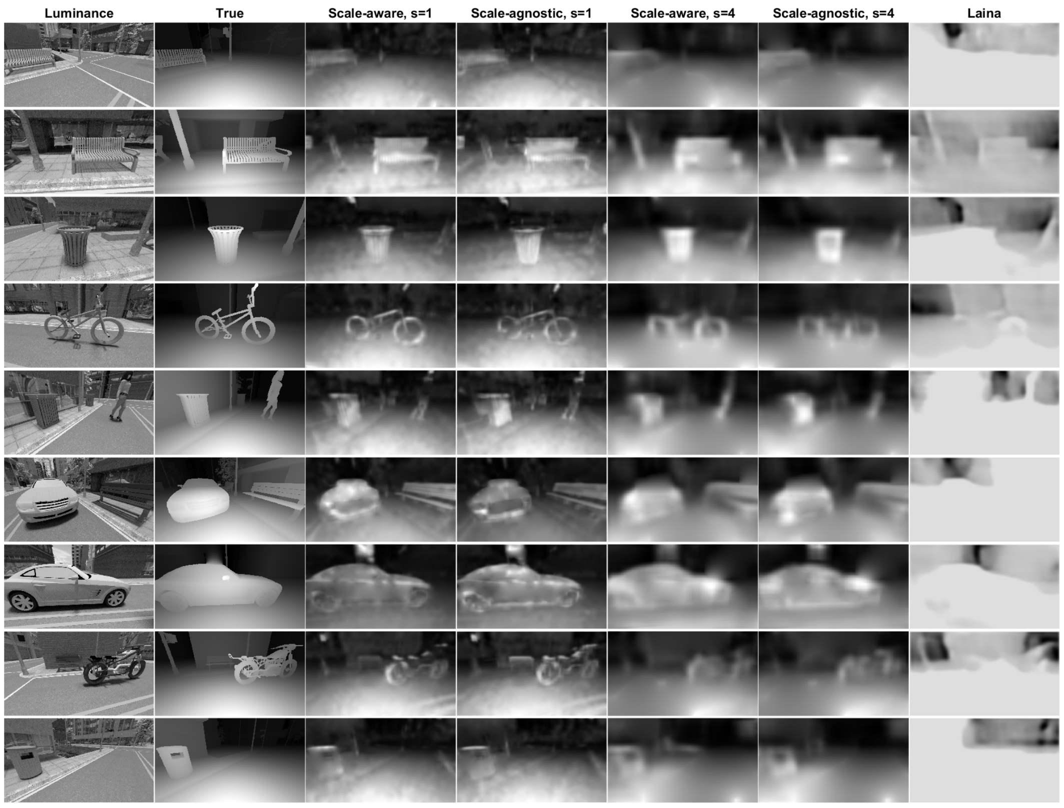


Figure 5.5: Inverse-range prediction results on the LIVE SV Database.

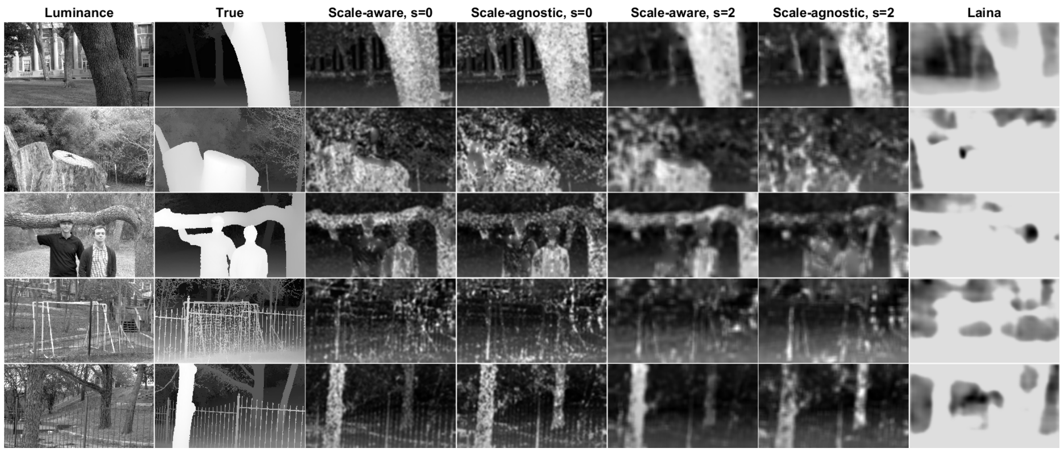


Figure 5.6: Inverse-range prediction results on the LIVE Color+3D Database.

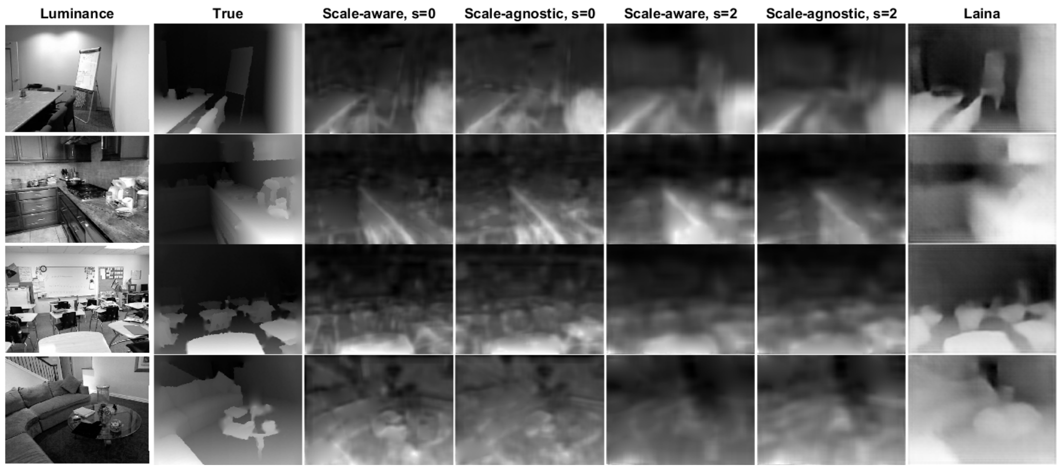


Figure 5.7: Inverse-range prediction results on the NYU Depth Database.

patch mean and variance. The latter two are represented in the network by the NS blocks, which predict scalar outputs. Though training begins with weakly biasing these two NS modules to predict the inverse-range patch mean and variance, this bias is removed to finish training. We are still, however, able to demonstrate that the values predicted by these intermediate network layers closely resemble the intended mean and variances scalars. In Figure 5.8, we show all three intermediate network outputs from the model trained on scale-2 data (360×640 image resolution) for all the LIVE SV images displayed in Figure 5.5. The predicted DNT, mean, and variance images all display similar characteristics to their ground-truth maps, which further supports the HVS motivation behind the design of our network.

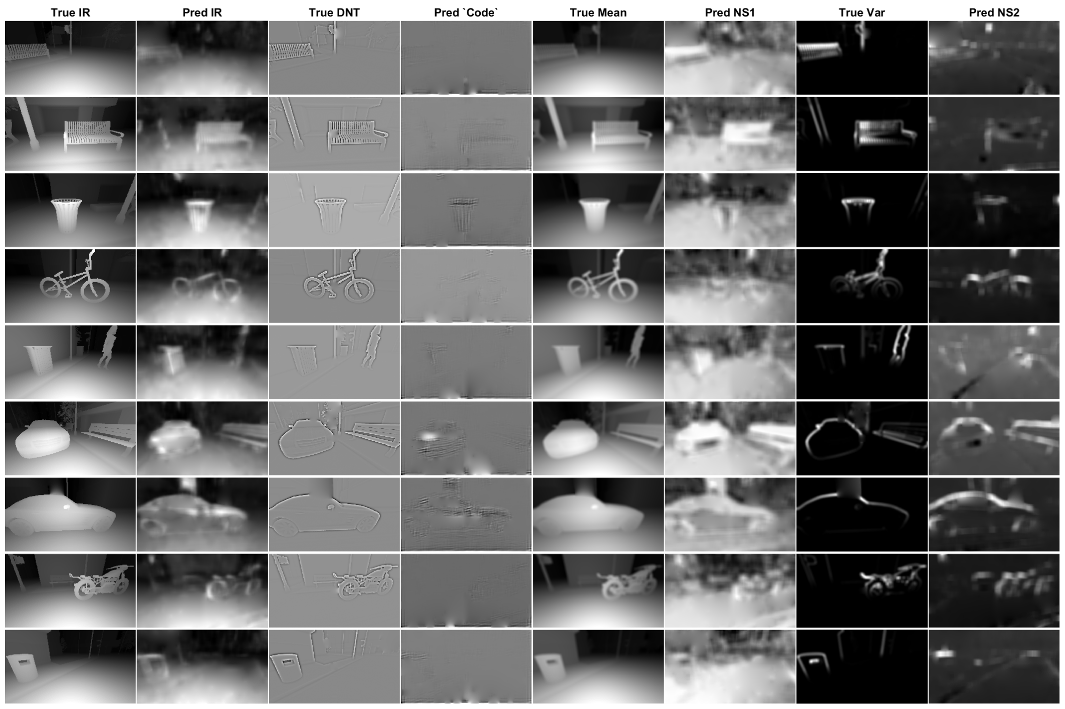


Figure 5.8: Intermediate model outputs. From left to right: the ground-truth inverse-range image; the predicted inverse-range image; the ground-truth divisively-normalized inverse-range image; the predicted divisively-normalized inverse-range image; the ground-truth inverse-range mean image; the predicted inverse-range mean image; the ground-truth variance image; and the predicted variance image.

5.3.2 The Value of NSS Features

As mentioned in Section 5.2.3, we also trained models without using the NSS features (F_s) as inputs to the model, as a way of probing the efficacy of these features. In these experiments, we excluded the NSS features and trained the models using only the luminance patches, which were fed into three separate CNN modules, each configured according to the luminance CNN described in Table 5.2. These luminance feature vectors were then separately fed into the same 'code' and 'NS' blocks shown in Figure 5.3a. For the SV examples shown in Figure 5.5, we also show, in Figure 5.9, the difference between the predictions when NSS features were included as input features (middle column) and excluded entirely from the network (last column). These examples clearly illustrate the enhanced performance afforded by including NSS features in the monocular depth estimation training process. A quantitative comparison is also provided in Table 5.6 for the scale-specific models (trained on 1/4-resolution SV data) that produced the results in Figure 5.9. Both the visual and numeric results strongly support the use of NSS features as inputs to the model as well as the idea that these kinds of fundamental, perceptually relevant statistical features can be used to significantly improve learned MDE models.

Table 5.6: Comparison with Previous Models

Database	**Model**	**MSE**		**MAE**		**nMAE**	
		Mean	**Median**	**Mean**	**Median**	**Mean**	**Median**
LIVE SV	w/ NSS	**0.4271**	**0.3322**	**0.3986**	**0.3739**	**0.3204**	**0.3174**
	w/o NSS	1.0158	0.8209	0.7089	0.6553	0.6310	0.6505

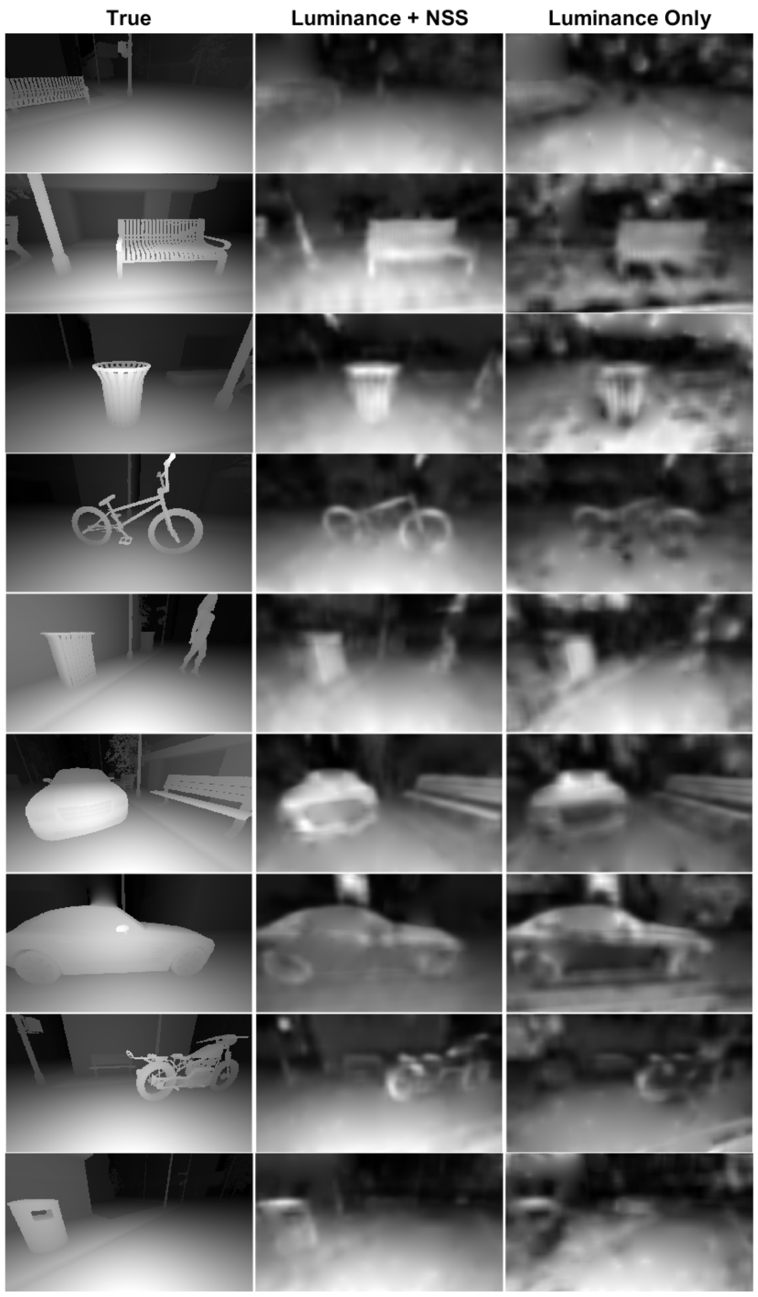


Figure 5.9: Comparison of results for models trained on single-scale (scale 2) data for the SV database, both with (middle column) and without (last column) using NSS features as inputs.

Chapter 6

Orthogonally-Divergent Fisheye Stereo[2]

We now shift our focus to estimating depths using fisheye images, specifically as it pertains to automotive surround-view (SV), which, as discussed in Chapter 3, uses four fisheye cameras placed on the front, right, rear, and left sides of a vehicle and generates a single birdseye-view image of the vehicle's surroundings by stitching together undistorted perspective-transformed versions of the four images captured by those fisheye cameras. Being able to estimate depths from fisheye images has potential applications not only in many general computer vision tasks, but within SV as well. For instance, one could use the knowledge of the scene structure that is gained from estimating depths to improve the stitched-together perspective-transformed birdseye image. If one knows where objects are located, common issues, such as ghosting or objects disappearing completely, which result from blindly stitching the four views together, may be avoided. Inherent in SV are four stereo camera pairs (Table 3.1), and because there exists a significant amount of overlap in FoVs between each pair, one only needs to exploit these overlapping views to use in

[2]Content presented in this chapter was first published in [76]. Co-authors were Martin Mueller, Tarek Lahlou, and Alan C. Bovik. Janice S. Pan did the work presented in [76] of formulating the fisheye stereo model, building and training the network, and designing and evaluating the pipeline while following the guidance and feedback provided by her co-authors.

triangulating depth.

Stereo systems are typically comprised of a pair of rectilinear cameras with parallel camera axes and narrow ($\sim$ 10 cm) baselines. The use of fisheye lenses enables stereo configurations to use wider baselines and non-parallel camera axes, since they capture larger FoVs that may still overlap under disparate camera positions and orientations. However, using fisheye lenses for stereo applications is inherently more challenging because wide-angle lenses produce spatial image distortions that get more severe closer to image edges, which are coincidentally where FoVs overlap and where one would want to exploit stereo vision.

We train a neural network to conduct correspondence prediction and use it along with the stereo camera parameters to predict a depth map, given a fisheye stereo pair. The proposed method does not require the removal of fisheye distortion, nor does it require stereo rectification. The trained network searches along epipolar curves to find correspondences, then triangulates and estimates depths under smoothness constraints.

We first detail the underlying fisheye stereo geometry and explain the model in Section 6.1 before discussing our pipeline for predicting depth and the neural network on which the pipeline depends (Section 6.2). The work presented in this chapter was previously published in [76].

6.1 Orthogonally-Divergent Fisheye Stereo Model

Because the stereo systems within SV use fisheye lenses instead of rectilinear lenses, each camera within any stereo pair can be rotated about its vertical axis away from the other and still maintain FoVs that overlap. Figure 6.1 depicts a diverging fisheye stereo system, where the left and right cameras are symmetrically rotated by angles $+\alpha$ and $-\alpha$, respectively, about their Y-axes (i.e., their vertical axes). In the context of SV, usually $\alpha \approx 45^o$, and each camera is angled slightly downward, i.e., rotated about their X-axes, to capture more of the ground plane. Again, the 'left' and 'right' cameras in a stereo pair may be any of the pairs listed in Table 3.1 (refer to Fig. 3.1).

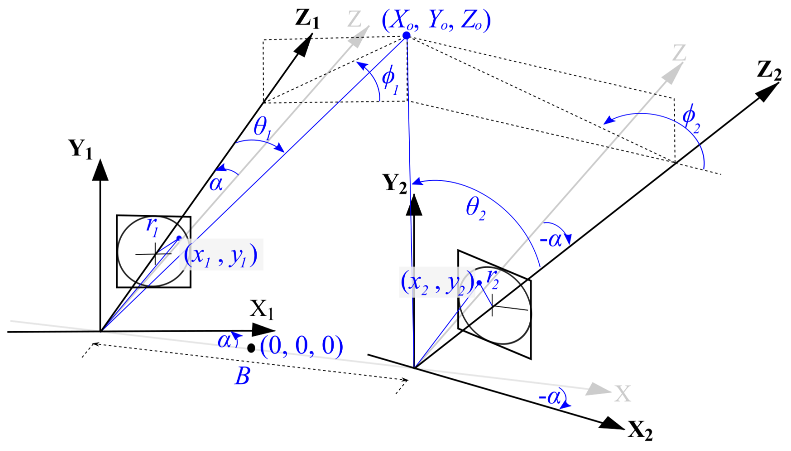


Figure 6.1: Fisheye stereo with diverging camera axes.

The camera lenses are assumed to follow an equisolid fisheye model, though the same method can be extended to stereo systems comprised of lenses with any known distortion model. Other camera projection models are shown

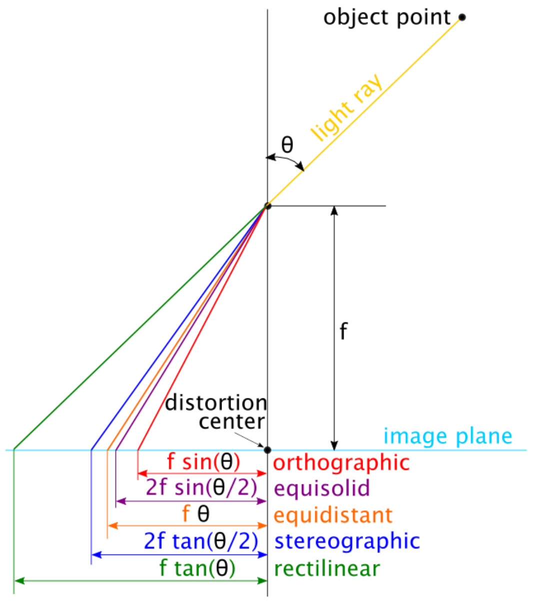


Figure 6.2: Comparison between camera models; adapted from [105].

in Figure 6.2. Under the equisolid fisheye model, the relationship between any image point (x_i, y_i) for camera i and its 3D point (X_{oi}, Y_{oi}, Z_{oi}) in the coordinate frame of camera i can be described as follows:

$$r_i = \sqrt{x_i^2 + y_i^2} = 2f \sin\left(\frac{\theta_i}{2}\right), \tag{6.1}$$

$$\phi_i = \arctan\left(\frac{y_i}{x_i}\right), \tag{6.2}$$

$$\tan(\theta_i) = \frac{\sqrt{X_{oi}^2 + Y_{oi}^2}}{Z_{oi}}, \tag{6.3}$$

where f is the focal length; r_i is the distance between (x_i, y_i) and the image center; θ_i is the angle between the Z-axis of camera i and (X_{oi}, Y_{oi}, Z_{oi}); and ϕ_i is the angle between the XZ-plane of camera i and (X_{oi}, Y_{oi}, Z_{oi}), which is the world point (X_o, Y_o, Z_o) in the coordinate frame of camera i. Thus, given a left image point (x_1, y_1), and a known or estimated depth Z_{o1} one can solve for X_{o1} and Y_{o1}:

$$\begin{aligned} X_{o1} &= \sqrt{\frac{(Z_{o1}\tan(\theta_1))^2}{1+\tan(\phi_1)^2}}, \\ Y_{o1} &= X_{o1}\tan(\phi_1). \end{aligned} \tag{6.4}$$

Both θ_1 and ϕ_1 depend only on the reference point (x_1, y_1) and can be computed from (6.1) and (6.2), respectively. After obtaining X_{o1} and Y_{o1} using (6.4), the world point (X_{o1}, Y_{o1}, Z_{o1}) can be represented in the coordinate frame of camera 2: (X_{o2}, Y_{o2}, Z_{o2}), from which (x_2, y_2), the matching point to (x_1, y_1), can be computed.

We only need to know how to transform between the left and right cameras' coordinate systems, which are defined by $(\mathbf{X}_1, \mathbf{Y}_1, \mathbf{Z}_1)$ and $(\mathbf{X}_2, \mathbf{Y}_2, \mathbf{Z}_2)$, respectively. The translation and rotation matrices used to transform points in either coordinate system into the world coordinate system are:

$$\mathrm{T}_L = \begin{bmatrix} 1 & 0 & 0 & -\frac{B}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{T}_R = \begin{bmatrix} 1 & 0 & 0 & \frac{B}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.5}$$

$$\mathrm{R}_{Ly} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathrm{R}_{Ry} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.6}$$

where baseline B is the distance between the camera centers, and the midpoint along B is the origin of the world coordinate system. In SV, B is typically on the order of a meter. In capturing the LIVE SV Database, $B \approx 1.4$m.

We also define a rotation matrix about the X-axis, because each camera is downtilted 20^o towards the ground to allow the fisheye cameras to capture enough of the ground to generate complete birdseye-view images:

$$\mathrm{R}_{Lx} = \mathrm{R}_{Rx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 20^o & -\sin 20^o & 0 \\ 0 & \sin 20^o & \cos 20^o & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.7}$$

The rigid transformation matrices can then be written as:

$$\mathrm{P}_{LW} = \mathrm{T}_L \mathrm{R}_{Ly} \mathrm{R}_{Lx} \quad \text{and} \quad \mathrm{P}_{RW} = \mathrm{T}_R \mathrm{R}_{Ry} \mathrm{R}_{Rx}, \tag{6.8}$$

where P_{LW} and P_{RW} are the transformation matrices to map points in the left and right camera coordinate frames, respectively, into the world coordinate frame. I.e.,

$$\begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix} = \mathrm{P}_{LW} \begin{bmatrix} X_{o1} \\ Y_{o1} \\ Z_{o1} \\ 1 \end{bmatrix} = \mathrm{P}_{RW} \begin{bmatrix} X_{o2} \\ Y_{o2} \\ Z_{o2} \\ 1 \end{bmatrix}, \tag{6.9}$$

or, more directly,

$$\begin{bmatrix} X_{o2} \\ Y_{o2} \\ Z_{o2} \\ 1 \end{bmatrix} = \mathrm{P}_{RW}^{-1} \mathrm{P}_{LW} \begin{bmatrix} X_{o1} \\ Y_{o1} \\ Z_{o1} \\ 1 \end{bmatrix}. \tag{6.10}$$

Thus, for any point (x_1, y_1) in the left image, and a known (or estimated) depth Z_{o1}, we can solve for X_{o1} and Y_{o1}. Therefore, a simple sweep over a range of depth values $\left\{Z_{o1}^k\right\}, k \in [0, K]$ can yield a set of candidate world points $\left\{(X_{o1}^k, Y_{o1}^k, Z_{o1}^k)\right\}$, which, when projected into the right image, comprise the epipolar line segment that serves as our correspondence search space. The objective is then to select the k associated with the best corresponding point. We use $K = 25$ meters (refer to Section 6.2.1).

6.1.1 Fisheye distortion removal and stereo rectification

We use the aforementioned analytical understanding of the fisheye stereo problem without attempting to reduce it to the more traditional rectilinear stereo problem, even though the latter approach is simpler and already has many robust proven approaches (see Section 2.3.1). The reason for working in the fisheye space is to retain the information provided by the wide fisheye FoV. In Figure 6.3, we show a pair of fisheye stereo images on the left. The middle and right-most columns show the images with their fisheye distortions removed at different resolutions. Much of the information at the edges of the fisheye FoVs is lost in going from the fisheye to the non-fisheye images, and capturing these regions, i.e., having such wide FoVs, is precisely why fisheye lenses are used over rectilinear lenses.

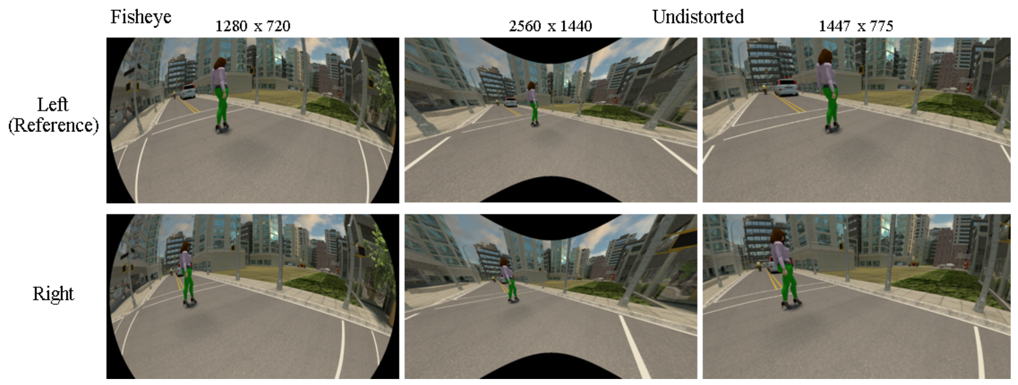


Figure 6.3: For a given fisheye stereo pair (1st column), different resolutions of the distortion-corrected images (2nd and 3rd columns). Resolutions are denoted at the top of each column. The angle between the camera axes that captured the fisheye images in the left column is 0^o, i.e., the stereo system has parallel camera axes, and the baseline between the cameras is about 1.4m.

6.2 Depth-prediction pipeline

Now that we understand how to map between the captured images and the world, we will present our approach to computing depth from fisheye stereo. Figure 6.4 shows the depth estimation pipeline given an input fisheye stereo pair. The search space in Step 2 for finding corresponding patches is limited by the patch size and by the spatial extent of the fisheye FoV projected on the image plane. Patches are extracted in Step 3 according to these constraints. Point correspondences in Step 4 are computed by taking the best match along each epipolar search curve, as predicted by a trained network, which we discuss next. The epipolar curves are determined by the camera and stereo parameters, so we can pre-compute them in a prior calibration step and then input them in a look-up table (LUT) to more efficiently compute point correspondences between any stereo pair. When triangulating depth (Step 5), the Euclidean distance between (X_o, Y_o, Z_o) and the left camera center is the estimated depth for (x_1, y_1). Hence, depth is a measurement on all three 3D space coordinates. To find point correspondences, we use a local correspondence prediction network, which we trained using patches extracted from image pairs in the LIVE SV-F Database [76], which contains $18,648$ unique fisheye stereo pairs. We use the LIVE SV-F database, which was created for exploring SV problems, because it is the only openly available database containing well-calibrated wide-baseline orthogonally-divergent fisheye stereo image pairs with ground-truth depth maps.

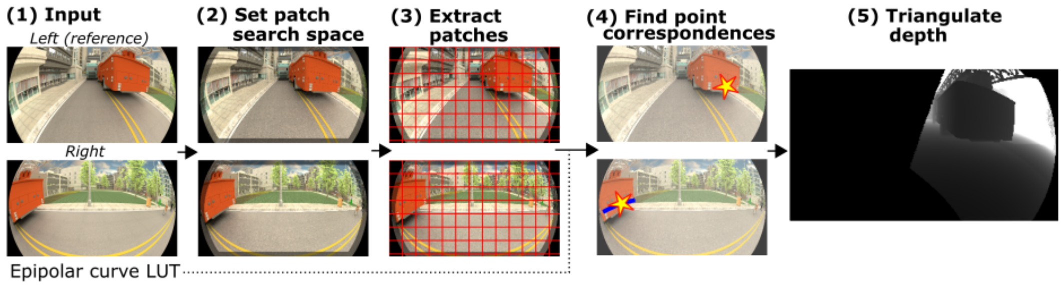


Figure 6.4: Algorithm pipeline for computing depth image.

6.2.1 Neural network for correspondence prediction

The correspondence prediction network we trained was based on the 2-channel architecture in [119], as shown in Figure 6.5 and we followed similar methods for training. We used 200 of the 222 scenes in the database yielding $16,800$ stereo pairs (200 scenes $\times$ 21 augmentations per scene $\times$ 4 pairs per augmentation). From these pairs, we drew 250, 056 positive (correct) matching patches from feature points and an equal number of negative (incorrect) matches. We used 80% of the data for training, 10% for validation, and 10% for testing. The inputs to the network are 25×25 RGB patches, and we used a regularized hinge loss function, similar to the work in [119], using a manually designed matching cost for patch pairs.

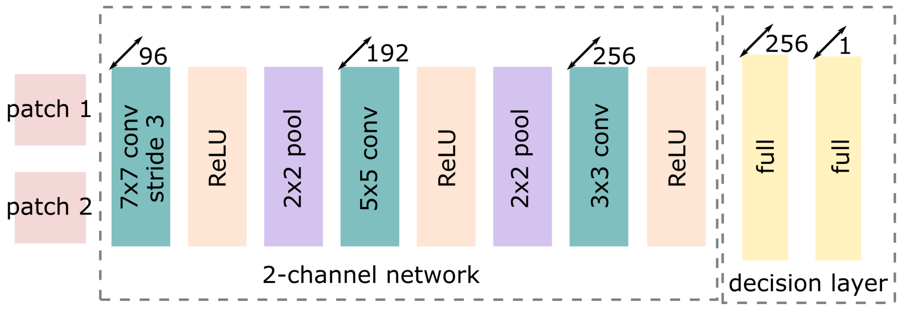


Figure 6.5: 2-channel architecture [119].

To extract positive matches from the dataset, we used the ground-truth depth maps produced by Blender to compute the 3D location of every pixel in each fisheye image. By projecting all the image points into world coordinates, we matched points projected from fisheye image pairs using a nearest-neighbor search to select the point in the right image closest in 3D space to each point in any given left image. The *cost* of the match was represented by the distance between the matched points. A threshold of 0.1 meters was chosen to be the upper limit on classifying a match as true *positive*. In other words, if the *best* match, i.e., the 3D point from the right image that was closest to the left reference pixel, was more than 0.1 meters away, the reference pixel was considered to have *no* stereo match. Thus, the collection of positively matching patches had costs in the range $[0, 0.1]$.

To extract negative matches, for each positive match, we computed the epipolar curve segment in the right image corresponding to the reference point in the left image. We used a depth threshold of 25 meters when computing the epipolar curve segment. In other words, the maximum triangulated depth between the reference point (in the left image) and any point on the epipolar curve segment (in the right image) is 25 meters. We then randomly selected a patch from this epipolar curve that was not the true match to be the negative match. To compute the *cost* of the match, we computed the difference between (1) the triangulated depth using the reference point and the positive matching point and (2) the triangulated depth using the reference point and the negative matching point. Recall that we established in Section 3.2 that depth in this work is computed as a 3D Euclidean distance. Therefore, by computing cost in this way, we still maintain the cost as representing a 3D distance for both positive and negative matches.

In order to achieve balance between the positive and negative matching scores, we flipped and scaled all positive and negative matching *costs* to obtain *scores* for the positive matches. Specifically, let us represent the distances between positive matches as d_p and the distances between negative patches as d_n:

$$d_p \in [0, 0.1], \qquad d_n \in (0, 25). \tag{6.11}$$

Note that the endpoints for negative depth distances d_n are non-inclusive, because all depths are non-zero, and the maximum triangulated depth using any point on any epipolar curve segment is 25 meters. Therefore, the difference between the true depth and the depth of a negative matching point will be in the range $(0, 25)$. To compute a more balanced set of target match scores, s_p and s_n for the positive and negative matches, respectively:

$$s_p = \frac{0.1 - d_p}{0.1}, \qquad s_n = \frac{-d_n}{25}. \tag{6.12}$$

By processing the costs for the positive and negative matches in this way, we obtain a set of match scores with the following properties:

- The strongest positive match score is $s_p = 1$, and the strongest negative match score is $s_n = -1$, so all scores lie within $[-1, 1]$.
- All positive matches have positive scores, and all negative matches have negative scores.
- The triangulated 3D distance between matches monotonically decreases as match score increases, i.e., as strength of match increases.

The hinge loss function we use is:

$$\min_w \frac{\lambda}{2}||w||_2 + \sum_{i=1}^{N} \max(0, 1 - s_i o_i^{net}), \tag{6.13}$$

where w are network weights, N is the mini-batch size, i is the training sample index, s_i is the matching score for patch pair i, and o_i^{net} is the network output for patch pair i. We thus train the network to primarily predict patch matches with scores that are indicative of the degree to which patches match, so selecting the patch candidate along an epipolar curve with the *largest predicted magnitude* can then reasonably serve as an effective method of choosing the best match.

We show examples of positive and negative matches in Figure 6.6. The first two columns show the captured left and right fisheye images, respectively. In the left fisheye image, the reference patch is indicated with a red box. In the right fisheye image, the blue and yellow boxes correspond to patches that positively and negatively match, respectively, to the red box. The right three columns show these patches extracted from their respective images, so we can better visualize features between both true matches and false matches and understand the difficulty sometimes in visually determining match strength.

6.2.1.1 Training details

We trained the 2-channel network using mini-batch sizes of 500, a total of 100 epochs, and Adam optimization with learning rate initialized at 0.001. Weights were initialized randomly, and we trained the model from scratch. All pixel values were scaled and centered to fall in the range [-1,1], i.e, if the original pixel value is v, the processed value is $2\left(\frac{v}{255} - 0.5\right)$, because the images are of type `uint8`). All training examples are shuffled prior to learning.

We monitor the validation loss while training, and every 100 steps, we compute the predicted classification accuracy, among other performance metrics, across the entire validation dataset. If the counts of true positives, true negatives, false positives, and false negatives are denoted by TP, TN, FP, and FN, respectively, the metrics are defined as follows:

$$\begin{aligned}
\text{Accuracy} &= \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}, && (6.14)\\
\text{TPR} &= \frac{\text{TP}}{\text{TP}+\text{FN}}, && (6.15)\\
\text{TNR} &= \frac{\text{TN}}{\text{TN}+\text{FP}}, && (6.16)\\
\text{PPV} &= \frac{\text{TP}}{\text{TP}+\text{FP}}, && (6.17)\\
\text{NPV} &= \frac{\text{TN}}{\text{TN}+\text{FN}}, && (6.18)\\
\text{F1} &= \frac{2\times\text{TPR}\times\text{PPV}}{\text{TPR}+\text{PPV}}, && (6.19)\\
& && (6.20)
\end{aligned}$$

where TPR, TNR, PPV, and NPV stand for true positive rate (i.e., sensitivity, recall), true negative rate (i.e., sensitivity), positive predictive value (i.e., precision), and negative predictive value, respectively, and the F1 score is the harmonic average between precision and recall. We report the quantitative results from training our network in Table 6.1.

Table 6.1: Network Performance on Test Data

Acc	TPR	TNR	PPV	NPV	F1
0.860	0.851	0.869	0.872	0.848	0.861

We show in Figure 6.7 pairs of positively and negatively matching patch pairs drawn from the test data. The first two columns titled ‘True Positives’ and 'True Negatives' show examples that were correctly classified. The third column shows examples of false positives and negatives. Above each left (reference) patch, we show the ground-truth match score, and over each right patch, we show the predicted matching score. Our network is fairly consistent in predicting positive matches to have match scores very close to 1, meaning our network may not have learned the strength-of-match concept for positive matches that we designed into our training data. However, with our use of hinge loss, erroneous predictions of match class (positive or negative) most significantly impact weight updates in network training, and strong correct predictions of positively-matching patches have minimal impact on network parameter updates. Additionally, more important than the magnitudes of the predicted match scores are the signs of the predictions, i.e., the predicted classes, and the quantitative results we presented in Table 6.1.

6.2.2 Results

The results presented in Table 6.1 only pertain to the trained network. We have yet to see how well the entire pipeline (Figure 6.4) performs when we use our trained network to predict point correspondences (Step 4 in Figure 6.4). To test how accurately the depth maps are predicted, we use the 22 scenes that were not previously used for training the correspondence prediction network. In other words, we used 1848 stereo pairs (22 scenes $\times$ 21 augmentations per scene $\times$ 4 pairs per augmentation), from which no patches were extracted for training, validating, or testing the network for predicting match scores. We used normalized absolute depth error to evaluate the accuracy of predicted depth maps:

$$\text{Error} \quad = \quad \frac{|G - P|}{|G|}, \tag{6.21}$$

where P is predicted depth, and G is ground-truth depth. We computed the mean normalized absolute depth error over all pixels in each stereo pair and then averaged these values over all $1,848$ pairs.

Additionally, because of the highly localized nature of correspondence prediction, with the 'best' match for any reference patch being that with the highest matching score, we also impose a smoothness constraint that considers neighboring correspondences when selecting the most appropriate match. Given a set of pixels in a window W_p around any reference image point p, denote the median 'best' depth in W_p as d_m, i.e., the depth which was triangulated from the corresponding patch with the highest matching score. For pixel p, there is also a set of candidate matching pixels C_p in the corresponding image along the epipolar curve, and each candidate pixel $c \in C_p$ has an associated depth d_c, which would result from triangulating depth between p and c. Thus, we penalize match scores for candidate matching pixels based on how far their associated depth is from the median neighboring depth:

$$o'_c \quad = \quad o^{net}_c - \lambda |d_c - d_m|, \tag{6.22}$$

where o'_c denotes the new matching score, and o^{net}_c is the raw network output for the patch pair comprised of the patch at reference pixel p and candidate matching pixel c. The weight λ determines how heavily to penalize a candidate depth's distance from its neighbors. The neighborhood window was fixed to be 3×3 when computing d_m.

To establish a benchmark for performance, we first predicted correspondences without the smoothness constraint and relied only on the highest match scores. We report the errors when using census similarity, sum of absolute differences (SAD), and sum of squared differences (SSD) to estimate correspondences in Table 6.2. We compare the results with the errors obtained when using our trained network and with the additional smoothing step (6.22). Our trained network already performed better than the benchmark similarity metrics, and the error was further reduced when we imposed the smoothness constraint. We show results from our depth-map prediction pipeline and the effectiveness of the additional smoothing step in Figure 6.8.

Table 6.2: Network Performance on Test Data

Similarity Metric	Error
Census	0.93
SAD	1.08
SSD	1.05
Network	0.75
Network, smoothed	0.51

We compute depths for pixels for which the estimated correspondence has a positive predicted match score. If even the highest match score along an epipolar curve is negative, then the reference point in the left image is considered to not have a predicted matching point in the right image. Additionally, our epipolar search curves were computed only for depths within 25 meters (as we discussed in Section 6.2.1), so when we evaluate depth prediction performance, we only consider pixels for which we have a valid depth estimate and can *expect* a valid depth estimate. Therefore, our evaluation results are based on pixels with positive predicted matches and ground-truth depths less than 25 meters.

6.3 Concluding Remarks

We have demonstrated the use of the new LIVE SV-F Database [76] in a new depth-map prediction pipeline for fisheye stereo inputs. We introduced the difficult stereo configuration involving a wide baseline and fisheye cameras, which, while it provides a significant amount of flexibility in computer vision tasks, also possesses inherent challenges the more this flexibility is exploited. In developing our pipeline for densely estimating depth values, we trained a neural network to conduct correspondence prediction and, by using the stereo system parameters, showed how it can be used to predict a depth map given a fisheye stereo pair without requiring the removal of fisheye distortions followed by stereo rectification, thus avoiding steps which are both common and necessary when using traditional stereo-matching methods. We also introduced a novel smoothness constraint for refining predicted depth maps that significantly improves the results. Despite the difficult conditions, the proposed method is able to produce depth estimates close to the ground-truth.

The proposed approach, however, requires a calibrated system, known camera parameters, and does not run in real time as implemented. In a SV system, if depth estimation using the inherent fisheye stereo is meant to improve rendering of the top-view image, robustness and speed are both paramount to the successful application of the algorithm. Requiring a prior calibration step indicates robustness may be an issue, because the system would need to be re-calibrated any time the system physically changes. We next show how we adapt our perceptual monocular depth estimation method to be able to predict depth maps from fisheye images. Our MDE method has fewer constraints, runs faster, and performs better, making it a more desirable solution for estimating scene structures with a SV camera configuration.

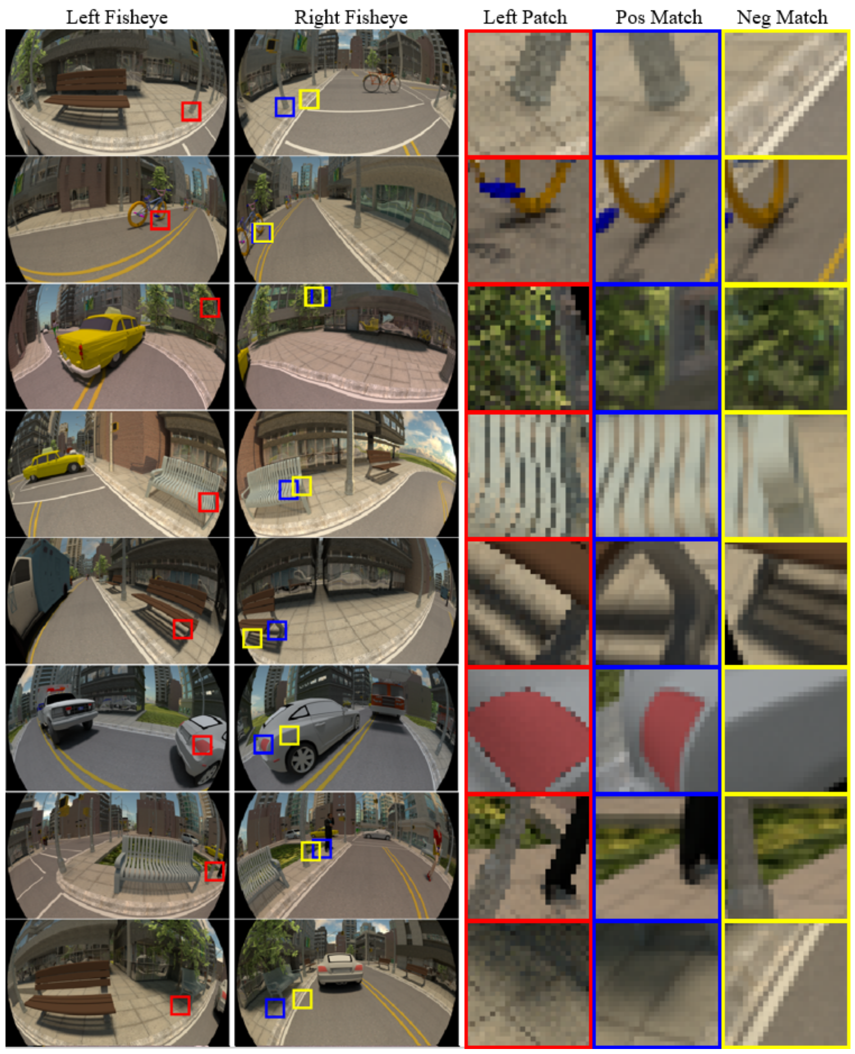


Figure 6.6: Examples of positive and negative patch matches. Columns from left to right: the left (reference) fisheye image, in which the reference patch is indicated with a red box; the right fisheye image, in which the blue and yellow boxes correspond to patches that positively and negatively match to the red box; the isolated red boxes from the left fisheye images; the blue boxes from the right fisheye images; and the yellow boxes from the right fisheye images.

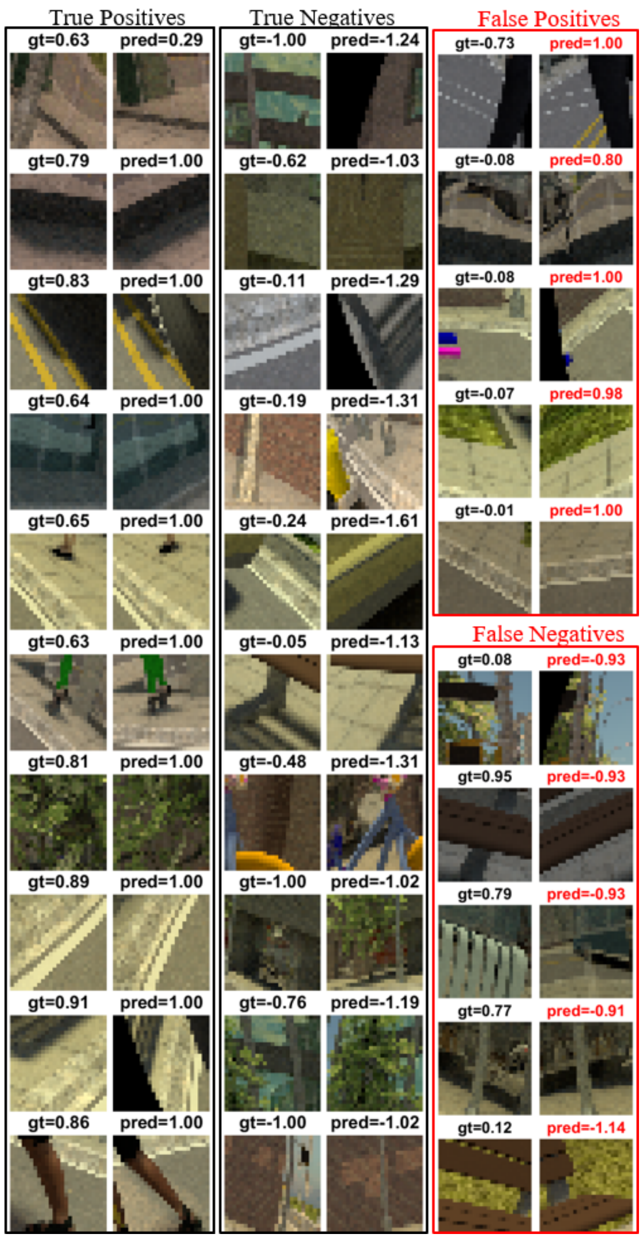


Figure 6.7: Predictions on example test patch pairs.

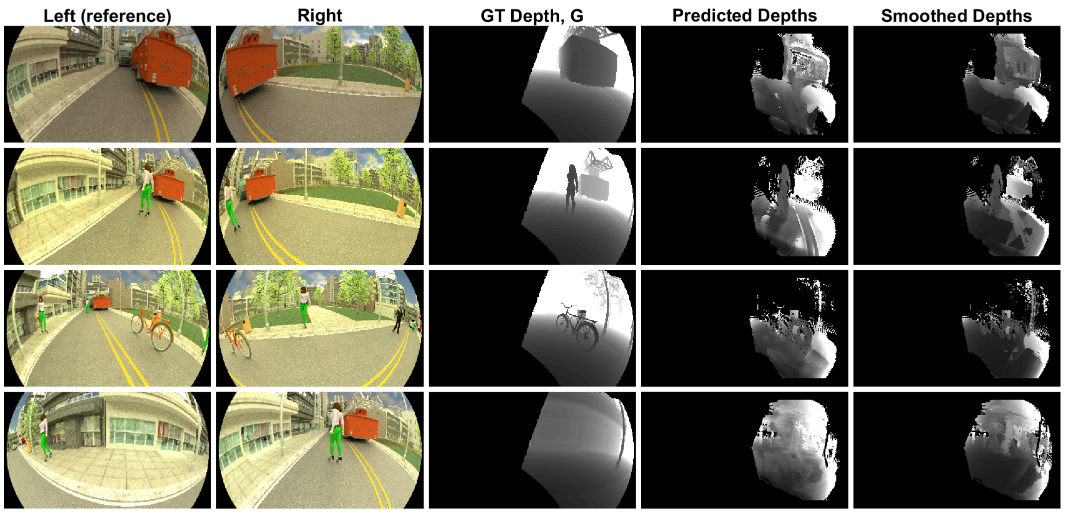


Figure 6.8: Example stereo pairs from the $1,848$ pairs reserved for testing the depth-map prediction pipeline (Fig. 6.4). First two columns: left and right fisheye captures; third column: ground-truth depths for pixels visible to both cameras and in the overlapping FoVs; fourth column: predicted depths triangulated using only the network-scored best-match pixel; fifth column: depth maps that result when the smoothness constraint is imposed.

Chapter 7

Fisheye Monocular Depth Estimation

In Chapter 6, we discussed a method for depth estimation using stereo-vision, i.e., using two cameras to triangulate the depth of any point that falls within the overlapping field-of-view (FoV) between the cameras. That method operates in the fisheye domain, meaning that the images are not transformed, via spatial distortion removal and stereo rectification, into a rectilinear stereo pair on which traditional stereo matching methods may be used. A method requiring such a transformation would fail to benefit from the main advantage provided by using a fisheye camera model, namely its near-180^o FoV.

Because fisheye cameras can capture such wide FoVs, their application in stereo configurations provides a lot of flexibility, particularly in the baseline separation between the cameras and in the relative orientation between the cameras. Even when the cameras are separated by a wide baseline or if the angle between the camera axes begins to diverge, there remains an overlapping FoV region between the captured images, so we still have stereo vision, and therefore, stereo methods may still be applied to predict the depth maps in these regions. However, the amount of overlap between the captured FoVs decreases as the baseline or divergence angle increases, which directly restricts the amount of depth information we are able to predict. As seen in [76], with orthogonally-divergent fisheye stereo, the resulting depth map is less than half of the reference image. Thus, relying on stereo methods in such challenging configurations can be limiting, and a monocular approach could provide much more flexibility in application. We adapt our NSS-based monocular depth estimation (MDE) method for fisheye images, which has the following important advantages over our stereo approach:

- Does not require a second view for triangulating depth;
- Does not require calibration of the camera system to compute camera extrinsic parameters;
- Provides depth estimates for the entire fisheye image, not just for an overlapping FoV.

In this chapter, we first explore fisheye NSS (Section 7.1) before explaining our process for adapting our MDE method to be able to handle fisheye-distorted images (Section 7.2). We then present the quantitative and visual results and compare the predictions made by our MDE model with those obtained using the stereo method presented in Chapter 6 (Section 7.4). As we will demonstrate, our monocular approach is less complex, has fewer constraints, and performs better at predicting depths than our stereo approach.

7.1 Fisheye NSS

Because the features used in our MDE model to predict depth are extracted from NSS models, we must first understand whether the fisheye distortion model changes the underlying NSS of images in a way that might prevent the meaningful extraction of NSS model parameters. We use the naturalistic fisheye images in the LIVE SV-F database, because we have already demonstrated the naturalness of the non-fisheye portion of the dataset in Section 4.1.

We observe that the naturalistic fisheye images in the LIVE SV database display similar NSS to those of pristine images captured using real-world rectilinear cameras. Additionally, because fisheye images are radially distorted, we partitioned the fisheye images into nine sub-images to analyze the effect of radial distortions on univariate NSS. Specifically, we partitioned the fisheye images into a 3×3 grid of equally-sized sub-images, and fit a GGD (4.4) to the divisively-normalized luminance coefficients of valid patches (i.e., we exclude the black bordering pixels not corresponding to any image content). We show how we partitioned the fisheye images in Figure 7.1, and it is apparent that the spatial distortions are strongest in sub-images around the perimeter and minimal in the center sub-image (fisheye_{22}). Figure 7.2 shows that these DNT luminance values have empirical distributions that are Gaussian and very similar in shape and scale to those drawn from undistorted, i.e., rectilinear, image data. Knowning that the fisheye-distorted images contain statistical regularities similar to those of undistorted images gives us confidence that the

NSS parameters extracted from fisheye images may be used in a method similar to that used for non-fisheye images to predict inverse-range and that our MDE method is appropriate for such a domain.

7.2 Adapting the NSS-based MDE Method for Fisheye

Fisheye images follow a known radial distortion based on the lens model (equisolid, in our case), so the normalized y-coordinate on its own is no longer sufficient in capturing the vertical placement of points in the image, which is a perceptual depth cue [102]. Therefore, we add a normalized x-coordinate feature to provide the network with sufficient spatial information. Specifically, the x-coordinate feature is computed using:

$$x = \frac{|w/2 - j|}{w/2}, \tag{7.1}$$

where j is the pixel index along the x-axis, and w is the width of the image in pixels. We normalize by half the image width starting from the middle of the image, where the spatial distortion is minimal.

As we previously did for each database, we also learn a new 64-element dictionary from the set of fisheye images within the LIVE SV-F database to train a MDE model specifically for fisheye images. Instead of using all seven of the scales we used with the LIVE SV dataset, for the LIVE SV-F data, we use only three resolutions: 720×1280 (the original resolution), 360×640, and 180×320, because we anticipate severe spatial distortions to make it more

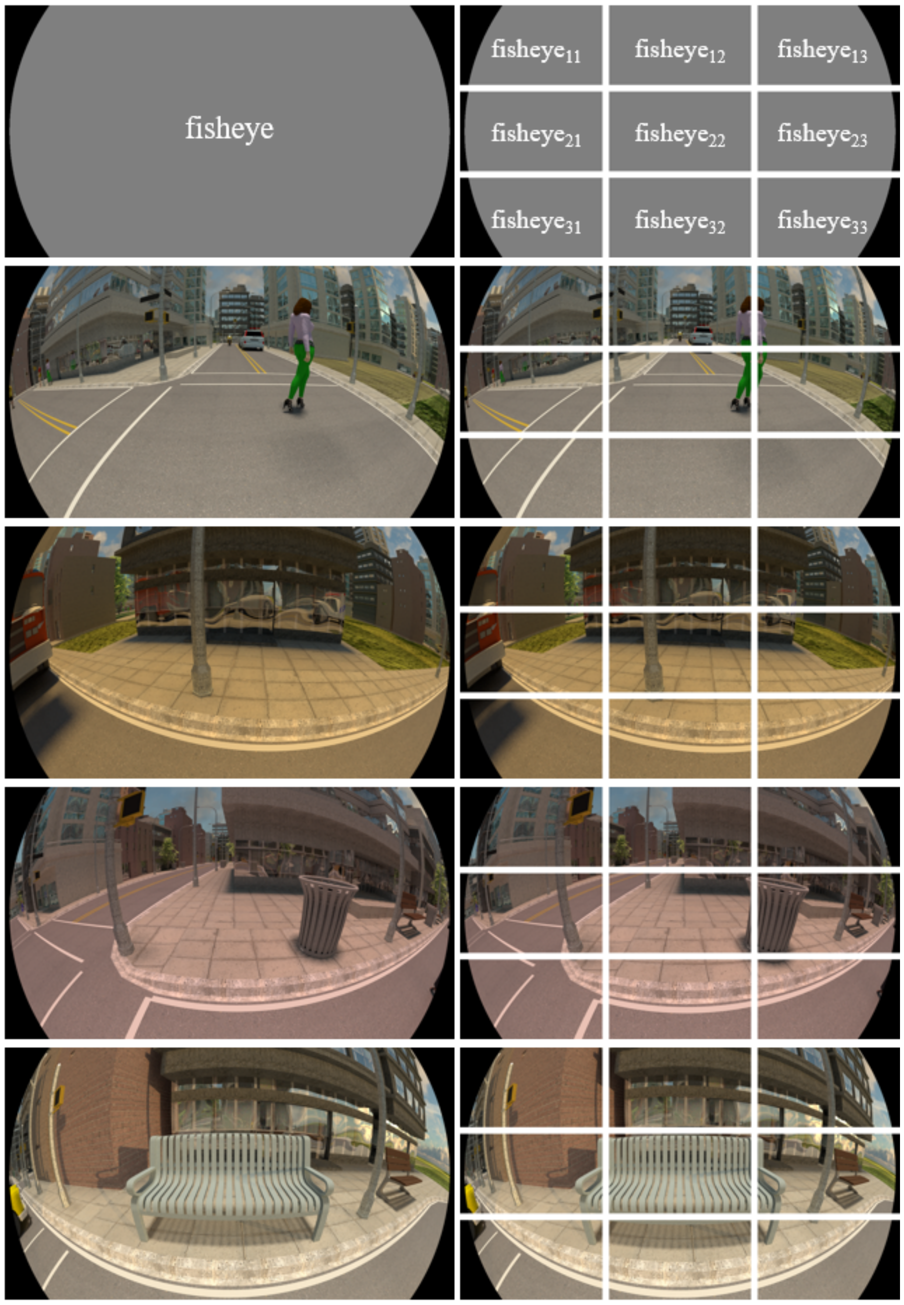


Figure 7.1: Top row: labeled regions for NSS analysis; Rows 2-5: Example fisheye images (left) with their partitions (right).

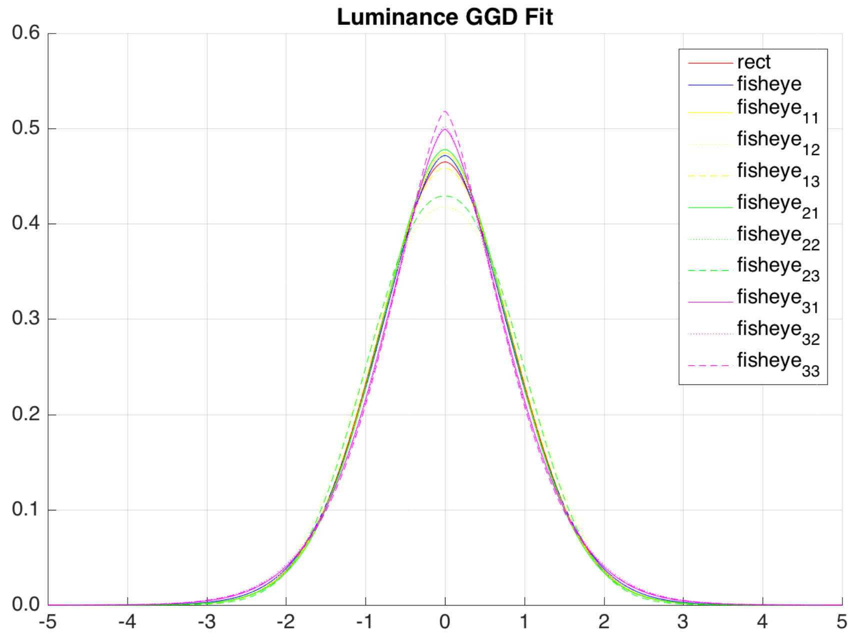


Figure 7.2: Comparison of the GGD fits to rectilinear, fisheye, and partitioned fisheye divisively-normalized luminance histograms.

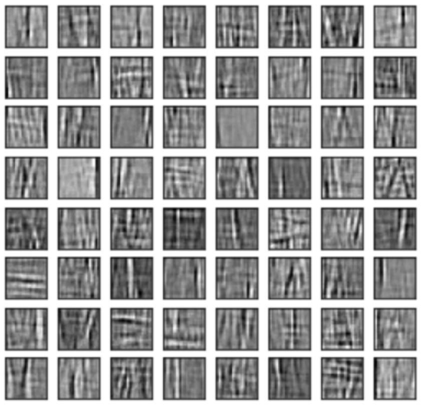


Figure 7.3: Dictionary elements of the learned sparse inverse-range patches computed on the LIVE SV-F Database.

difficult to train the same multi-scale model. In the DNT process (4.1), as was used for the non-fisheye data, $C_{ir} = 0.1$. Figure 7.3 shows the learned dictionary elements, and Figure 7.4 shows examples of DNT fisheye images reconstructed from this dictionary at two scales to demonstrate the generalizability of the learned dictionary patches.

7.3 Training Details

The same network shown in Figure 5.3 was trained with LIVE SV-F data, but with the additional x-coordinate feature, F_s becomes a 22-length feature vector, and the dimensions of the inputs to the 'code' and two NS blocks (listed in Table 5.2) are (983,) and (22,), respectively. Everything else with respect to training the network remains the same: the amount of training/validation/testing data, the two-tiered training procedure, the loss function, the patience threshold, the method for augmenting the training data,

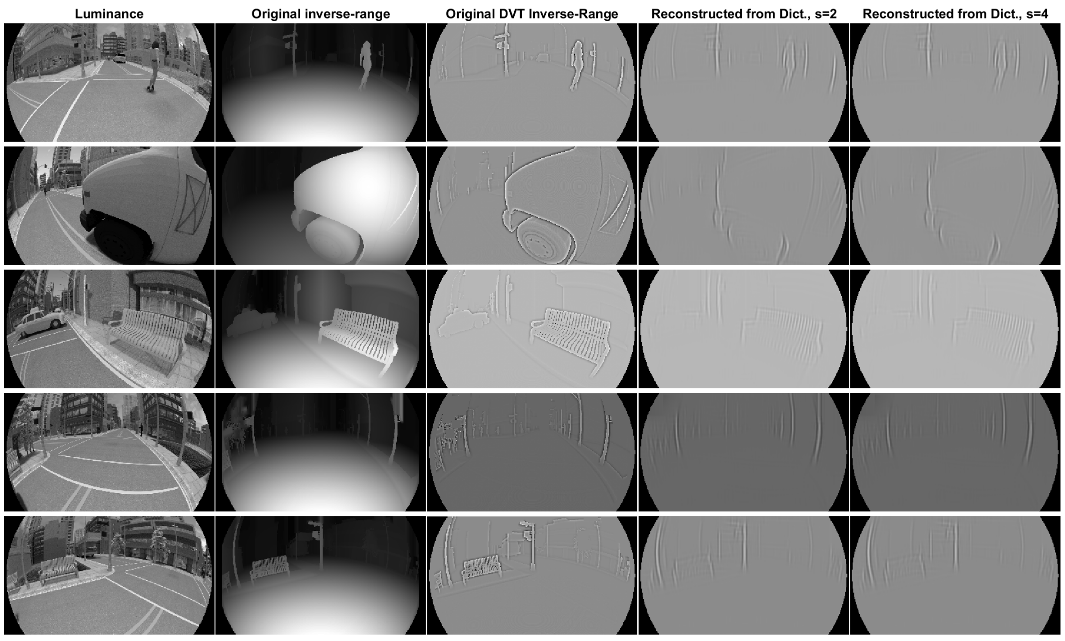

Luminance
Original inverse-range
Original DVT Inverse-Range
Reconstructed from Dict., s=2
Reconstructed from Dict., s=4

Figure 7.4: Examples showing the DNT inverse-range images reconstructed from the dictionary patches. From left to right in each row: original luminance image; ground-truth inverse-range map; divisively normalized bandpass inverse-range; DNT inverse-range map of resolution 360×640 (scale 2) reconstructed from the dictionary patches; DNT inverse-range map of resolution 180×320 (scale 4) reconstructed from the dictionary patches.

and so on. Please refer to Section 5.2.4 for the details.

7.4 Results

We tested the MDE fisheye model in two ways: first, following the same testing procedure we used to test the model trained on rectilinear data, and then on the 'test' images in the LIVE SV-F dataset. As introduced in [76], these test images capture 22 new scenes within the synthetic cityscape and, like the 20 LIVE SV images previously set aside to test the trained model, also contain content unseen by the training pipeline. From these 22 scenes, we only selected the $1,848$ images captured using the orthogonally-divergent camera configuration, so as to appropriately compare results to those computed using the stereo pipeline (Chapter 6).

In Table 7.1, we show the average and median MAE and nMAE (defined in Section 5.3) when the model was trained on patches extracted from images of the same resolution (scale-aware) and when the model was trained on patches extracted from images across all three tested resolutions (scale-agnostic). Therefore, the image resolutions listed in Table 7.1, for the *scale-aware* results, correspond to the data on which the models were both trained and tested, whereas for the *scale-agnostic* results, the listed resolutions correspond only to the data on which the models were tested, because the scale-agnostic model was trained using data across all three scales. Figure 7.5 shows the fisheye versions of the examples shown in Figure 5.5, when scale-agnostic and scale-aware (for scale 0: 720×1280; and scale 2: 360×640) models were run on the fisheye versions of the test images. As we observe, the same model with the same architecture and training parameters can perform quite similarly at predicting inverse-range values in fisheye images with only a single addition to the input features.

As is the case in Tables 5.3 and 5.4, smaller numbers indicate better performance. If we consider normalized MAE, we still observe that the scale-agnostic (multi-scale) models perform slightly worse than the scale-aware (single-scale) models, but again, the multi-scale results support the efficacy of using the scale-invariant NSS to learn scale-agnostic models.

Table 7.1: Results from Multi-Scale and Single-Scale Models on the LIVE SV-F Database

Model	**Image Resolution**	**Avg MAE**	**Median MAE**	**Avg nMAE**	**Median nMAE**
Scale-Agnostic	720×1280	0.4422	0.5040	0.5385	0.5818
	360×640	0.4388	0.4534	0.5591	0.5255
	180×320	0.5085	0.5230	0.5751	0.6348
Scale-Aware	720×1280	0.4250	0.3993	0.5040	0.4763
	360×640	0.3972	0.4045	0.4801	0.4615
	180×320	0.4937	0.4552	0.5669	0.5543

7.4.1 Comparison with Stereo

We also compute the performance on the 22 ‘test’ scenes in the LIVE SV-F database that were generated specifically to serve as a disjoint subset of scenes containing no overlapping content with the rest of the database. As we did in Section 6.2.2, we used $1,848$ stereo pairs, inputting the left (reference) image of each pair into our MDE model. We also downscaled the test images to a resolution of 180×320, because the stereo pipeline was developed only for this resolution. We computed performance using both inverse-range and depth

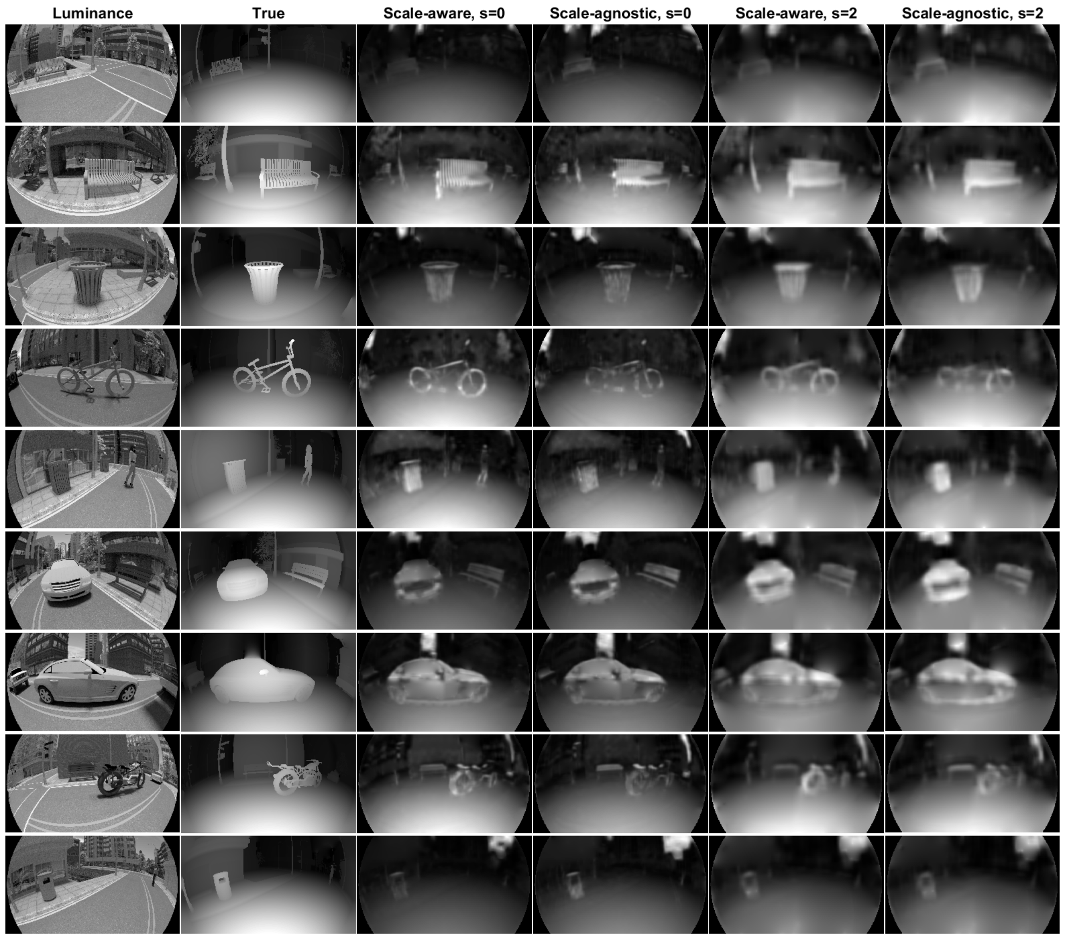


Figure 7.5: Inverse-range prediction results on fisheye images from LIVE SV Database.

on the entire image as well as on the 'overlapping FoV' region by assuming the wide-baseline orthogonally-divergent stereo configuration (Section 6.1). Our stereo method was developed to predict depths on points falling within the overlapping FoV, so the monocular and stereo results can only be compared in this domain.

Table 7.2 shows the quantitative results when comparing inverse-range and depth predictions when considering both entire images and overlapping FoV regions. To compute the MDE models' predictions of depth, we simply took the reciprocal of the predicted inverse-range values, and to compute the stereo pipeline's predictions of inverse-range, we similarly took the reciprocal of the predicted depth values. As we can see, the scale-aware (i.e., single-scale or SS) MDE model consistently outperforms the scale-agnostic (i.e., multi-scale or MS) model and the stereo method (in the overlapping FoV regions, where the comparison is appropriate to make).

To clarify, the scale-aware model that was used to generate the results in Table 7.2 was trained on patches extracted from images that were scaled down to 180×320; as was the stereo model. The multi-scale model, on the other hand, was trained on patches extracted from images across three scales, and *all* of the results in Table 7.2 are based on the disjoint 180×320-resolution test set.

We also show visual results in Figure 7.6 on the same examples used to demonstrate the performance of the stereo pipeline (Figure 6.8). We show the inverse-range predictions on the entire 'reference' image along with the depths

Table 7.2: Results on the LIVE SV Fisheye Test Database, Resolution 180×320

Comparison Domain	**Model**	**Avg MAE**	**Median MAE**	**Avg nMAE**	**Median nMAE**
Inv-Range	MDE-MS	0.2782	0.2748	0.4781	0.4490
	MDE-SS	0.2281	0.2156	**0.4229**	**0.3380**
Inv-Range, Overlapping FoV	MDE-MS	0.2673	0.2622	0.4388	0.4397
	MDE-SS	0.2019	0.1857	**0.3592**	**0.2900**
	Stereo	0.2460	0.2463	0.4233	0.4218
Depth	MDE-MS	0.5264	0.2713	0.2222	0.1922
	MDE-SS	0.5588	0.2739	**0.1794**	**0.1660**
Depth, Overlapping FoV	MDE-MS	0.1985	0.1785	0.2211	0.1732
	MDE-SS	0.1974	0.1746	**0.1713**	**0.1545**
	Stereo	1.7090	1.6668	0.5136	0.4988

(computed as the reciprocal of the inverse-range values) on the overlapping FoV if the reference is considered to be the left image captured in the wide-baseline orthogonally-divergent stereo configuration described in Section 6.1. Our MDE method produces much smoother depth maps than the stereo pipeline produces. The low resolution in combination with the constant patch size (31×31) yields predicted depth maps that have very blurred edges and seem to lack fine detail, however the depths of significant objects are still estimated quite well, and the scenes predicted by our monocular models are much closer to the ground-truth scenes than those predicted using our stereo pipeline, and these visual results are supported by the quantitative results presented in Table 7.2.

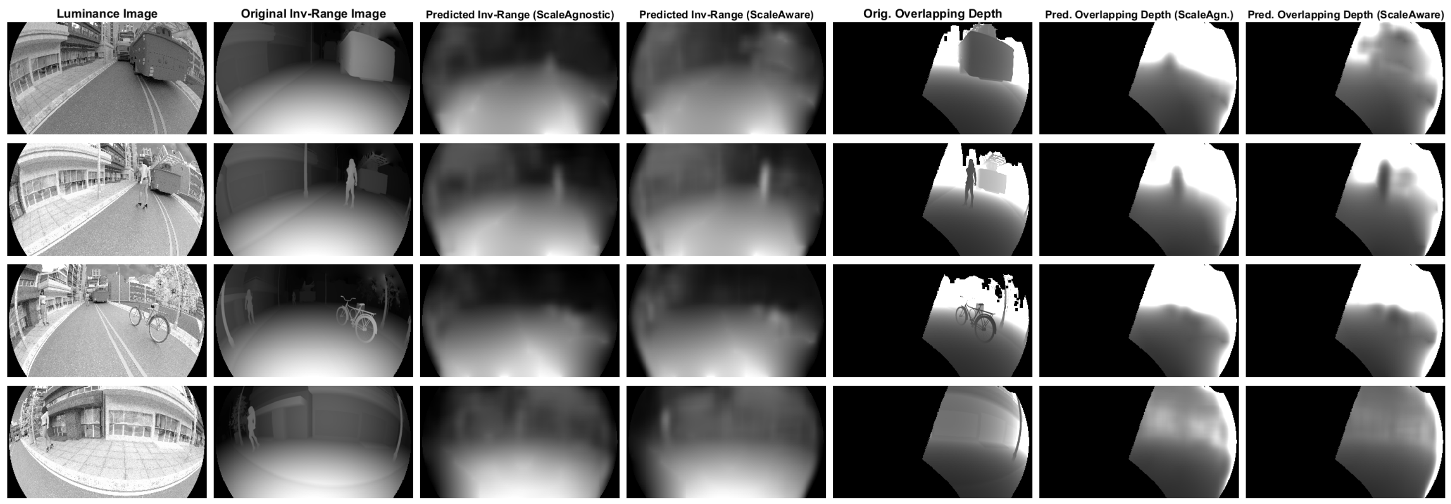

Luminance Image
Original Inv-Range Image
Predicted Inv-Range (ScaleAgnostic)
Predicted Inv-Range (ScaleAware)
Orig. Overlapping Depth
Pred. Overlapping Depth (ScaleAgn.)
Pred. Overlapping Depth (ScaleAware)

Figure 7.6: Results from running our MDE model on the same four examples shown in Figure 6.8. The first column shows the reference image, which is the left image in each of the stereo pairs in Figure 6.8. The next three columns show inverse-range results (L-R): the ground-truth inverse-range map; the predictions using the scale-agnostic model; and the predictions using the scale-aware model. The last three columns show the depth results (reciprocal of inverse-range) on the overlapping region between the left and right views (L-R): the ground-truth depth map; the predictions using the scale-agnostic model; and the predictions using the scale-aware model. The stereo method in Chapter 6 predicts depths for only the overlapping FoV between the cameras, so the the last two columns displayed in this Figure can be compared to the results in Figure 6.8.

Chapter 8

Conclusion

We now briefly summarize the four main contributions we introduced in this work:

- the LIVE SV Database;
- a bivariate NSS correlation model;
- a wide-baseline orthogonally-divergent fisheye stereo network and pipeline;
- and a perceptual MDE model.

We first introduced a new large-scale synthetic database containing fisheye images and co-registered depth maps initially for the purpose of exploring approaches to the surround-view (SV) feature in Advanced Driver Assistance Systems (ADAS). The database, called the LIVE SV Database, though it was generated with fisheye cameras, can easily be converted to a non-fisheye database, which we have also done for this work. The scenes that were captured are photorealistic; a wide variety of camera configurations were used; and the database contains large amounts of data for exploring a variety of computer vision problems. Thus, this database has substantial potential applicability, and can be useful in developing many different computer vision models.

We then presented a new bivariate correlation model for image luminance coefficients in Section 4.2. The model is a simple constrained difference-of-exponentials, and we demonstrated how well it models the empirical correlations of the real-world images in the LIVE Color+3D Database Release 2 [102] and of the synthetic images non-fisheye images in the LIVE SV Database [76]. We then used features extracted from this model, along with other univariate and bivariate NSS features, in a modular patch-based deep neural network that is trained to predict inverse-range patches from a single luminance patch. The model architecture was inspired by the DNT processing theorized to be performed by the HVS. We also utilized a learned dictionary to enable the representation of DNT inverse-range patches by sparse codes, which served as intermediary features within the predictive model. Our use of DNT inverse-range patches (as well as DNT luminance patches) was motivated by the low-level visual processing in the HVS, and representing inverse-range patches in such a way significantly reduced the required complexity of our network.

We trained our network on patches extracted from images of the same scale as well as on patches extracted from images across multiple scales to see how well our models could generalize. We found that our scale-agnostic models performed comparably to the scale-specific models, likely because the features that are input to the model are based on NSS models, which are scale-invariant. By incorporating NSS features and a DNT-based network architecture, we presented a novel perceptually-motivated deep-learning approach for densely estimating inverse-range maps, and we showed that it can perform well on complicated scenes, as well as on databases lacking a sufficient number co-registered image/depth pairs to train deep full-image models.

In Chapter 6, we focused on the task of obtaining depth from fisheye stereo and detailed the fisheye models, system parameters, and relevant mapping equations in Section 6.1 before presenting our depth-prediction pipeline and the correspondence-prediction network within it in Section 6.2. We adapted a network that was previously developed to predict matching patches to our problem and our unique database. We specifically had to design the matching scores to represent match strength, and we trained our network from scratch, providing all training details in Section 6.2.1.1. To evaluate the isolated network, we computed classification metrics, and to evaluate its function in the entire pipeline, we looked at normalized absolute depth error, or normalized MAE. By comparing our network to classical similarity metrics, such as the census similarity metric, sum of absolute differences (SAD), and sum of squared differences (SSD), we showed that our network's ability to predict matching patches propagates through the pipeline to the final predicted depth map.

Finally, in Chapter 7, we adapted our MDE model to use on fisheye images by adding an x-coordinate feature and learning a new sparse dictionary for patches drawn from DNT inverse-range fisheye images. We first demonstrated that even though fisheye images suffer from severe spatial distortions, their DNT luminance coefficients still follow similar NSS models as those developed for pristine rectilinear images, thus supporting the application of our MDE model in the fisheye domain. We compared results from the scale-agnostic, scale-aware, and stereo models, and showed that our NSS-based MDE models perform better than the stereo pipeline, and the scale-aware MDE model performs better than the scale-agnostic model, though the results are close and still support the efficacy of using scale-invariant NSS features to learn scale-agnostic networks.

Bibliography

[1] Archive3D. Archive3d. `http://archive3d.net/`.

[2] Fred Attneave. Some informational aspects of visual perception. *Psychological Review*, 61(3):183, 1954.

[3] Horace B Barlow. Single units and sensation: A neuron doctrine for perceptual psychology? *Perception*, 1(4):371–394, 1972.

[4] Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory Communication*, 1:217–234, 1961.

[5] Anthony J. Bell and Terrence J. Sejnowski. The âĂIJindependent componentsâĂİ of natural scenes are edge filters. *Vis. Research*, 37(23):3327–3338, 1997.

[6] Blender. Blender 2.77 Release. `http://download.blender.org/release/Blender2.77/`, 2016.

[7] A. C. Bovik. Automatic prediction of perceptual image and video quality. *Proc. IEEE*, 101(9):2008–2024, Sep. 2013.

[8] Y. Cao, Z. Wu, and C. Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE*

Trans. Circuits and Systems for Video Technology, 28(11):3174–3182, Nov 2018.

[9] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[10] Yi-Yuan Chen, Yuan-Yao Tu, Cheng-Hsiang Chiu, and Yong-Sheng Chen. An embedded system for vehicle surrounding monitoring. *International Conference on Power Electronics and Intelligent Transportation System*, 2:92–95, 2009.

[11] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[12] A. Couturier. Scene City. `http://cgchan.com/store/scenecity`, Nov 2016.

[13] S. Dabral, S. Kamath, V. Appia, M. Mody, B. Zhang, and U. Batur. Trends in camera based Automotive Driver Assistance Systems (ADAS). *IEEE International Midwest Symposium on Circuits and Systems*, pages 1110–1115, Aug 2014.

[14] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and

Thomas Brox. Flownet: Learning optical flow with convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[15] M. Drulea, I. Szakats, A. Vatavu, and S. Nedevschi. Omnidirectional stereo vision using fisheye lenses. *IEEE International Conference on Intelligent Computer Communication and Processing*, pages 251–258, Sept 2014.

[16] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[17] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Adv. in Neural Inf. Process. Systems*, pages 2366–2374, 2014.

[18] M. J. Bertin et al. *Pisot and Salem Numbers.* user Verlag, Berlin, 1992.

[19] David J Field. Relations between the statistics of natural images and the response properties of cortical cells. *J. Optical Soc. of America*, 4(12):2379–2394, 1987.

[20] David J Field. Wavelets, vision and the statistics of natural scenes. *Philosophical Trans. Royal Society of London. Series A: Mathematical, Physical, Engineering Sciences*, 357(1760):2527–2542, 1999.

[21] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth

estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, June 2018.

[22] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *Proc. Eur. Conf. Comput. Vis.*, pages 740–756, 2016.

[23] S. Gehrig, C. Rabe, and L. Krueger. 6D vision goes fisheye for intersection assistance. *Canadian Conference on Computer and Robot Vision*, pages 34–41, May 2008.

[24] S.K. Gehrig. Large-field-of-view stereo for automotive applications. *Proceedings of the Workship on Omnidirectional Vision, Camera Networks and Nonclassical Cameras*, 1, 2005.

[25] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[26] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[27] Clement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc.*

IEEE Conf. Comput. Vis. Pattern Recogn., July 2017.

[28] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation. *arXiv preprint arXiv:1806.01260*, 2018.

[29] E Gómez, MA Gomez-Viilegas, and JM Marin. A multivariate generalization of the power exponential family of distributions. *Comm. in Statistics-Theory and Methods*, 27(3):589–600, 1998.

[30] Google. Google maps. `https://www.google.com/maps`.

[31] Hyowon Ha, Sunghoon Im, Jaesik Park, Hae-Gon Jeon, and In So Kweon. High-quality depth from uncalibrated small motion clip. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, June 2016.

[32] Guy Hadash, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Alon Jacovi. Estimate and replace: A novel approach to integrating deep neural networks with existing applications. *arXiv preprint arXiv:1804.09028*, 2018.

[33] K. Hamada, Z. Hu, M. Fan, and H. Chen. Surround view based parking lot detection and tracking. *IEEE Intelligent Vehicles Symposium*, pages 1106–1111, June 2015.

[34] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys. Real-time direct dense matching on fisheye images using plane-sweeping stereo. *International Conference on 3D Vision*, 1:57–64, Dec 2014.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 770–778, 2016.

[36] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. *Computer Society Conference on Computer Vision and Pattern Recognition*, 2:807–814, June 2005.

[37] Aapo Hyvärinen and Patrik Hoyer. Emergence of phase-and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, 12(7):1705–1720, 2000.

[38] Jinn-Yeu Jou and Alan C Bovik. Improved initial approximation and intensity-guided discontinuity detection in visible-surface reconstruction. *Comput. Vis., Graphics, Image Process.*, 47(3):292–326, 1989.

[39] Biliana Kaneva, Antonio Torralba, and William T Freeman. Evaluation of image features using a photorealistic virtual world. *International Conference on Computer Vision*, pages 2282–2289, 2011.

[40] K. Karsch, C. Liu, and S. B. Kang. Depth transfer: Depth extraction from video using non-parametric sampling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(11):2144–2158, Nov 2014.

[41] Keras-contrib. DSSIM Loss Function. `https://github.com/keras-team/keras-contrib/blob/master/keras_contrib/losses/dssim.py`, Dec. 2018.

[42] J. Kim, V. Kolmogorov, and R. Zabih. Visual correspondence using energy minimization and mutual information. *IEEE International Conference on Computer Vision*, pages 1033–1040, 2003.

[43] Seungryong Kim, Kihong Park, Kwanghoon Sohn, and Stephen Lin. Unified depth prediction and intrinsic image decomposition from a single image via joint convolutional neural fields. In *Proc. Eur. Conf. Comput. Vis.*, pages 143–159, 2016.

[44] N. Kita. Dense 3D measurement of the near surroundings by fisheye stereo. *IAPR International Conference on Machine Vision Applications*, 2011.

[45] N. Kita, F. Kanehiro, M. Morisawa, and K. Kaneko. Obstacle detection for a bipedal walking robot by a fisheye stereo. *Proceedings of the IEEE/SICE International Symposium on System Integration*, pages 119–125, Dec 2013.

[46] Donald K. Knuth. *The TEXbook*. Addison-Wesley, 1984.

[47] Naejin Kong and Michael J. Black. Intrinsic depth: Improving depth transfer with intrinsic images. In *IEEE Int'l Conf. Comput. Vis.*, December 2015.

[48] George Kour and Raid Saabne. Fast classification of handwritten on-line arabic characters. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th Int'l Conf. of*, pages 312–318, 2014.

[49] George Kour and Raid Saabne. Real-time segmentation of on-line handwritten arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th Int'l Conf. on*, pages 417–422, 2014.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[51] Yevhen Kuznietsov, Jorg Stuckler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, July 2017.

[52] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM J. on Optimization*, 9(1):112–147, 1998.

[53] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *Int'l Conf. 3D Vision*, pages 239–248, 2016.

[54] Leslie Lamport. *LaTeX: A document preparation system.* Addison-Wesley, 2nd edition, 1994.

[55] Tai-sing Lee and Brian R Potetz. Scaling laws in natural scenes and the inference of 3D shape. In *Adv. in Neural Inf. Process. Systems*, pages 1089–1096, 2006.

[56] Bo Li, Chunhua Shen, Yuchao Dai, Anton Van Den Hengel, and Mingyi He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 1119–1127, 2015.

[57] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(10):2024–2039, 2016.

[58] Y. Liu, L. K. Cormack, and A. C. Bovik. Statistical modeling of 3D natural scenes with application to bayesian stereopsis. *IEEE Trans. Image Processing*, 20(9):2515–2530, Sep. 2011.

[59] Yang Liu, Lawrence K Cormack, and Alan C Bovik. Luminance, disparity, and range statistics in 3D natural scenes. In *Proc. SPIE, Human Vis. and Electronic Imaging*, volume 7240, page 72401G, 2009.

[60] Yu-Chih Liu, Kai-Ying Lin, and Yong-Sheng Chen. Bird's-eye view vision system for vehicle surrounding monitoring. *International Workshop on Robot Vision*, pages 207–218, 2008.

[61] W. Luo, A.G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016.

[62] F Mittelbach M Goosens and A Samarin. *The LaTeXCompanion.* Addison-Wesley, 1994.

[63] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proc. Int'l Conf. Mach. Learning*, pages 689–696, 2009.

[64] 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. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[65] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015.

[66] A. Mittal, A. K. Moorthy, and A. C. Bovik. No-reference image quality assessment in the spatial domain. *IEEE Trans. Image Processing*, 21(12):4695–4708, Dec 2012.

[67] A. Mittal, R. Soundararajan, and A. C. Bovik. Making a âĂIJcompletely blindâĂİ image quality analyzer. *IEEE Signal Process. Letters*, 20(3):209–212, March 2013.

[68] J. Moreau, S. Ambellouis, and Y. Ruichek. 3D reconstruction of urban environments based on fisheye stereovision. *International Conference on Signal Image Technology and Internet Based Systems*, pages 36–41, Nov 2012.

[69] J. Moreau, S. Ambellouis, and Y. Ruichek. Equisolid fisheye stereovision calibration and point cloud computation. *Conference on Serving Society with Geoinformatics*, Nov 2013.

[70] T. Nishimoto and J. Yamaguchi. Three dimensional measurement using fisheye stereo vision. *SICE Annual Conference*, pages 2008–2012, Sept 2007.

[71] R. Okuda, Y. Kajiwara, and K. Terashima. A survey of technical trend of ADAS and autonomous driving. *IEEE International Symposium on VLSI Technology, Systems and Application*, pages 1–4, 2014.

[72] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vis. Research*, 37(23):3311–3325, 1997.

[73] Bruno A Olshausen and David J Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14(4):481–487, 2004.

[74] Janice Pan, Vikram Appia, and Alan C Bovik. Virtual top-view camera calibration for accurate object representation. *IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 21–24, 2016.

[75] Janice Pan and Alan C Bovik. Perceptual monocular depth estimation. *Neural Proc. Letters*, 2019. Submitted.

[76] Janice Pan, Martin Mueller, Tarek Lahlou, and Alan C Bovik. Orthogonally-divergent fisheye stereo. In *Int'l Conf. Advanced Concepts for Intell. Vis. Systems*, pages 112–124, 2018.

[77] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *J. Mach. Learning Research*, 12:2825–2830, 2011.

[78] Javier Portilla, Vasily Strela, Martin J Wainwright, and Eero P Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Trans. Image Process.*, 12(11), 2003.

[79] Brian Potetz and Tai Sing Lee. Statistical correlations between two-dimensional images and three-dimensional structures in natural scenes. *J. Optical Soc. of America*, 20(7):1292–1303, 2003.

[80] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1(2):4, 2017.

[81] A. Rajagopalan, S. Chaudhuri, and U. Mudenagudi. Depth estimation and image restoration using defocused stereo pairs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(11):1521–1525, Nov. 2004.

[82] Rajesh PN Rao, Bruno A Olshausen, and Michael S Lewicki. *Probabilistic Models of the Brain: Perception and Neural Function.* MIT press, 2002.

[83] Anirban Roy and Sinisa Todorovic. Monocular depth estimation using neural regression forest. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, June 2016.

[84] Daniel L Ruderman. The statistics of natural images. *Network: Computation in Neural Systems*, 5(4):517–548, 1994.

[85] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. *German Conference on Pattern Recognition*, pages 31–42, 2014.

[86] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[87] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1, 2003.

[88] J. Schneider, C. Stachniss, and W. FÃČÂűrstner. On the accuracy of dense fisheye stereo. *IEEE Robotics and Automation Letters*, 1(1):227–234, Jan 2016.

[89] Karnran Sharifi and Alberto Leon-Garcia. Estimation of shape parameter for generalized gaussian distributions in subband decompositions of video. *IEEE Trans. Circuits and Systems for Video Technology*, 5(1):52–56, 1995.

[90] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *Eur. Conf. Comput. Vis.*, pages 746–760, 2012.

[91] Eero P Simoncelli and Bruno A Olshausen. Natural image statistics and neural representation. *Annual Review of Neuroscience*, 24(1):1193–1216, 2001.

[92] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[93] Z. Sinno, C. Caramanis, and A. C. Bovik. Towards a closed form second-order natural scene statistics model. *IEEE Trans. Image Process.*, 27(7):3194–3209, July 2018.

[94] Michael Spivak. *The joy of TEX*. American Mathematical Society, Providence, R.I., 2nd edition, 1990.

[95] Che-Chun Su, A. C. Bovik, and L. K. Cormack. Natural scene statistics of color and range. In *IEEE Int'l Conf. Image Process.*, pages 257–260, Sep. 2011.

[96] Che-Chun Su, A. C. Bovik, and L. K. Cormack. Statistical model of color and disparity with application to bayesian stereopsis. In *IEEE Southwest Symp. Image Anal. and Interpretation*, pages 169–172, April 2012.

[97] Che-Chun Su, L. K. Cormack, and A. C. Bovik. Color and depth priors in natural images. *IEEE Trans. Image Process.*, 22(6):2259–2274, June 2013.

[98] Che-Chun Su, L. K. Cormack, and A. C. Bovik. Depth estimation from monocular color images using natural scene statistics models. In *IEEE Image, Video, Multidimensional Signal Process.*, pages 1–4, June 2013.

[99] Che-Chun Su, L. K. Cormack, and A. C. Bovik. New bivariate statistical model of natural image correlations. In *IEEE Int'l Conf. Acoustics, Speech, Signal Process.*, pages 5362–5366, May 2014.

[100] Che-Chun Su, L. K. Cormack, and A. C. Bovik. Closed-form correlation model of oriented bandpass natural images. *IEEE Signal Process. Letters*, 22(1):21–25, Jan 2015.

[101] Che-Chun Su, L. K. Cormack, and A. C. Bovik. Oriented correlation models of distorted natural images with application to natural stereopair quality evaluation. *IEEE Trans. Image Process.*, 24(5):1685–1699, May 2015.

[102] Che-Chun Su, L. K. Cormack, and A. C. Bovik. Bayesian depth estimation from monocular natural images. *J. of Vis.*, 17, May 2017.

[103] Che-Chun Su, Lawrence K. Cormack, and Alan C. Bovik. Bivariate statistical modeling of color and range in natural scenes. In *Proc. SPIE, Human Vis. and Electronic Imaging*, volume 9014, 2014.

[104] H. Tang, N. Joshi, and A. Kapoor. Learning a blind measure of perceptual image quality. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 305–312, June 2011.

[105] Michel Thoby. `http://michel.thoby.free.fr/Fisheye_history_short/Projections/Models_of_classical_projections.html`, Nov 2012.

[106] B. Thomas, R. Chithambaran, Y. Picard, and C. Cougnard. Development of a cost effective bird's eye view parking assistance system. *IEEE Recent Advances in Intelligent Computational Systems*, pages 461–466, 2011.

[107] Alf J. van der Poorten. Some problems of recurrent interest. Technical Report 81-0037, School of Mathematics and Physics, Macquarie University, North Ryde, Australia 2113, August 1981.

[108] J Hans van Hateren and Dan L Ruderman. Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proc. Royal Society of London. Series B: Biological Sciences*, 265(1412):2315–2320, 1998.

[109] J. Hans Van Hateren and Arjen van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proc. Royal Society of London. Series B: Biological Sciences*, 265(1394):359–366, 1998.

[110] M. J. Wainwright, O. Schwartz, and E. P. Simoncelli. Natural image statistics and divisive normalization: Modeling nonlinearities and adaptation in cortical neurons. *Probabilistic Models of the Brain: Perception and Neural Function*, 2001.

[111] Martin J Wainwright and Odelia Schwartz. Natural image statistics and divisive. *Probabilistic Models of the Brain: Perception and Neural Function*, page 203, 2002.

[112] Z. Wang and A. C. Bovik. Reduced- and no-reference image quality assessment. *IEEE Signal Process. Magazine*, 28(6):29–40, Nov 2011.

[113] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004.

[114] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 5354–5362, 2017.

[115] Kai Xu, Vladimir G Kim, Qixing Huang, Niloy Mitra, and Evangelos Kalogerakis. Data-driven shape analysis and processing. *SIGGRAPH ASIA 2016 Courses*, page 4, 2016.

[116] J. Yamaguchi. Three Dimensional Measurement Using Fisheye Stereo Vision. *Advances in Theory and Applications of Stereo Vision*, 2011.

[117] M. Yu and G. Ma. 360 surround view system with parking guidance. *SAE International Journal of Commercial Vehicles*, 7:19–24, 2014.

[118] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. *European Conference on Computer Vision*, pages 151–158, 1994.

[119] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2015.

[120] Amir Roshan Zamir and Mubarak Shah. Image geo-localization based on multiplenearest neighbor feature matching usinggeneralized graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1546–1558, 2014.

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

[122] B. Zhang, V. Appia, I. Pekkucuksen, Y. Liu, A. U. Batur, P. Shastry, S. Liu, S. Sivasankaran, and K. Chitnis. A surround view camera solution for embedded systems. *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pages 662–667, 2014.

[123] H. Zhao and J. K. Aggarwal. 3D reconstruction of an urban scene from synthetic fish-eye images. *IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 219–223, 2000.

[124] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, et al. Making Bertha driveÃćÂĂÂŤ–An autonomous journey on a historic route. *IEEE Intelligent Transportation Systems Magazine*, 6(2):8–20, 2014.

Vita

Janice S. Pan received her B.S. in Electrical Engineering from The University of Texas at Austin in 2013. She joined the Laboratory for Image and Video Engineering (LIVE) in the Fall of 2013 under the supervision of Dr. Alan C. Bovik and earned her M.S. degree in 2016.

Her research interests include computer vision, machine learning, and data science. She was a recipient of the Virginia & Ernest Cockrell, Jr. Fellowship in Engineering (2013-2016), the Dr. Brooks Carlton Fowler Endowed Presidential Graduate Fellowship in Electrical and Computer Engineering (2018), and the Charles W. and Margaret A. Tolbert Endowed Fellowship in Electrical and Computer Engineering (2019).

Contact: janicespan@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.