Monocular Depth Estimation with Edge-Based Constraints and Active Learning

by

Anshul Rai

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2019 by the
Graduate Supervisory Committee:

Yezhou Yang, Chair
Wenlong Zhang
Jianming Liang

ARIZONA STATE UNIVERSITY

August 2019

ABSTRACT

The ubiquity of single camera systems in society has made improving monocular depth estimation a topic of increasing interest in the broader computer vision community. Inspired by recent work in sparse-to-dense depth estimation, this thesis focuses on sparse patterns generated from feature detection based algorithms as opposed to regular grid sparse patterns used by previous work. This work focuses on using these feature-based sparse patterns to generate additional depth information by interpolating regions between clusters of samples that are in close proximity to each other. These interpolated sparse depths are used to enforce additional constraints on the network's predictions. In addition to the improved depth prediction performance observed from incorporating the sparse sample information in the network compared to pure RGB-based methods, the experiments show that actively retraining a network on a small number of samples that deviate most from the interpolated sparse depths leads to better depth prediction overall.

This thesis also introduces a new metric, titled *Edge*, to quantify model performance in regions of an image that show the highest change in ground truth depth values along either the x-axis or the y-axis. Existing metrics in depth estimation like Root Mean Square Error(RMSE) and Mean Absolute Error(MAE) quantify model performance across the entire image and don't focus on specific regions of an image that are hard to predict. To this end, the proposed *Edge* metric focuses specifically on these hard to classify regions. The experiments also show that using the *Edge* metric as a small addition to existing loss functions like $L1$ loss in current state-of-the-art methods leads to vastly improved performance in these hard to classify regions, while also improving performance across the board in every other metric.

DEDICATION

To my parents, Dr. Rathika Rai and Dr. B. Umesh Rai, for their unwavering confidence in me and for giving me the freedom to shape my life.

ACKNOWLEDGMENTS

I want to thank my advisor Dr. Yezhou Yang for being a great mentor and always being ready to help. I deeply appreciate the advice and for providing me a work environment that was very conducive for research.

I want to thank my committee members, Dr. Wenlong Zhang and Dr. Jianming Liang for their advice and feedback on my research work and this thesis.

I want to extend a special thanks to (soon to be Dr.) Duo Lu. I am very grateful for the time I got to spend learning first hand about how to approach engineering and research problems, and the discussions that always left me motivated to push myself a bit more.

I want to thank all the members of the ASU Active Perception Group for insightful discussions that introduced me to a wide variety of fields and for inspiring me with your research work. I would specially like to thank my closest collaborators –Shibin, Gowtham, and Varun– for making our shared work environment hassle-free.

Finally, I would like to thank my parents, without whom this wouldn't be possible.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

As human beings, we may not fully realize how big a role depth estimation plays in our lives. Ever since we are born, our ability to gauge the distance of objects from ourselves enables us to interact with the world around us. This ability plays a key role in our daily mundane decision-making processes; Do I need to physically move my legs to walk so that I am able to reach an object? How far should I move my finger to touch a key on the keyboard? How quickly do I need to react to dodge an incoming object? All of these tasks, and more, are dependent on our ability to quickly and accurately gauge the depth of objects in the world around us. This makes the problem of making machines learn the relative distance of objects around them extremely important. If we are to develop functioning Artificial Intelligence, depth estimation is an extremely important prerequisite for the ability to reason about the world. Therefore, this has been a topic of active research in the computer vision community.

The widespread impact of deep learning in the computer vision community has been well documented. Depth estimation is a subfield where its impact can be especially felt due to the rapid improvements that have been seen over the past few years. Traditionally, most applications that needed depth information would not consider using a camera-based system over depth sensors that used structured light or LiDARs; single camera systems didn't provide high enough performance, while stereo camera systems were computationally expensive to calibrate and could only be used in feature-rich scenes as triangulation would not work otherwise. However, the improvements in Graphics Processing Unit(GPU) technology and improved Convolutional Neural Net(CNN) architectures, coupled with the drawbacks of traditional depth sensors like LiDARs (much more expensive and have a limited range up to which they can provide reliable prediction) have made camera-based systems a viable option for accurate and accessible depth estimation. As single camera systems possess multiple advantages over their stereo counterparts (cheaper, no calibration required so easier to handle, ubiquitous in society due to smartphones and other small electronics), single camera depth estimation is a topic of high importance.

Monocular depth estimation is usually posed as a regression problem; a deep CNN based architecture takes a single three channel RGB image as input and outputs an image of the same size with depth predictions for every pixel. To address the limitations of methods that use only RGB images, recent work(Ma and Karaman 2018) adds an additional channel that contains easily obtained sparse depth information with the RGB image to extract additional performance from the network. This sparse depth information can be acquired either from an additional low-resolution depth sensor or from a visual odometry algorithm, but the pattern obtained from the former (regular grid) is very different from the latter(clustered around detected features). Existing works in this field have focused on regular grid-shaped sparse patterns for their benchmarks on the popular KITTI(Geiger, Lenz, and Urtasun 2012) and NYU Depth Dataset V2(Nathan Silberman and Fergus 2012) datasets. But since pure camera-based systems can only obtain patterns that are clustered around features, and not uniformly spaced across the image, we focus our work on benchmarking results with feature-based sparse depth patterns. Therefore, in this thesis, we focus on the supervised learning of monocular depths using sparse patterns generated from feature detection algorithms like ORB(Rublee et al. 2011).

Using feature-based sparse patterns provide a more realistic baseline for pure camera-based systems while still providing the benefits of incorporating sparse depth information in a model, compared to models that work purely with RGB images. We can enforce additional constraints on the model by interpolating depths for regions between the feature based sparse depths that are in close proximity to each other. As features obtained from feature detectors like ORB are usually clustered, this thesis argues that the interpolated depths are modeling distinct object surfaces in the RGB image. As most points on a surface have roughly identical depths, these interpolated depths can be used as a proxy for ground truth depths to update the deep neural net model's parameters. Finally, an active learning framework is used to selectively retrain network on pixels in the model's depth predictions that deviate most from the interpolated depths, which leads to noticeable better depth predictions overall.

As feature based sparse patterns do not provide uniformly spread out sparse depth information like regular grid-based sparse patterns, we focus on improving depth predictions of the most challenging parts of the image, namely boundaries of objects and regions of transitions between objects. To quantify and improve the performance of the model on such regions this thesis introduces a new metric called "EDGE". This thesis shows that while the current state-of-the-art methods perform well in traditional metrics like RMSE and MAE, it doesn't perform well on the challenging regions of the image. Using the EDGE metric as an additional component to the loss function not only leads to a huge improvement of the state-of-the-art models in the above mentioned challenging regions, it also leads to an improvement on nearly all the other metrics as well.

1.1 Contributions of This Thesis

- In Chapter 2, we consider the monocular depth estimation problem and explores the different variants. We then explore existing monocular depth estimation research and provide an explanation of traditional metrics used to benchmark performance across datasets.
- In Chapter 3, we propose a sparse depth sampling framework that sets a more realistic baseline for pure camera-based systems compared to existing work.
- In Chapter 4, we propose an interpolation framework that utilizes feature-based sparse depth samples that can later be used to enforce additional constraints on model training.
- In Chapter 5, we propose a new metric to quantify prediction performance on challenging regions in an RGB image. We also show that this metric can be used as a component to existing loss functions to show improved results on the baseline NYU-D V2 dataset.
- In Chapter 6, we explain the experimental setting and present our results for the proposed frameworks and metrics against existing work. We also show why previous work isn't well suited for camera only systems through quantitative results.
- Finally, Chapter 7 concludes the thesis and provides a brief exploration of future work.

Chapter 2

BACKGROUND AND METRIC EXPLANATION

2.1 Related Work

Traditional depth estimation work focused on using only RGB images to predict depths by relying on probabilistic models and domain knowledge based feature engineering. One such approach(Saxena, Chung, and Ng 2006) tried to learn depths with a Markov Random Field model by trying to estimate the absolute scale of patches in an RGB image, while others(Konrad, Wang, and Ishwar 2012; Karsch, Liu, and Kang 2014; Liu, Salzmann, and He 2014) used non-parametric approaches involving querying of images with similar photometric content from a database.

However, more recent work has been increasingly focused on using deep learning to solve the depth estimation problem. Pioneering work that utilized deep learning(Eigen, Puhrsch, and Fergus 2014) used a stacked convolutional neural net(CNN) architecture to get better predictions where one CNN network predicted the global coarse scale, while the other refined network predictions. (Laina et al. 2016) introduced a ResNet(He et al. 2016) based deep residual network to improve performance obtained from using a deep CNN architecture with a conditional random field(Liu, Shen, and Lin 2015). Most existing works(Badrinarayanan, Kendall, and Cipolla 2017; Long, Shelhamer, and Darrell 2015) make use of some variant of the encoder-decoder architecture. This is because standard encoder-decoder architectures can produce full-resolution prediction maps with pixel-wise predictions.

Unfortunately, standard methods can't compete with existing alternative sensors like LiDARs. Therefore, current state-of-the-art work has tried to combine the benefits of using deep neural net architectures with different sensors that can easily produce sparse depth predictions, which are then combined to produce overall better depth predictions. (Ma and Karaman 2018) uses uniform sparse depth input concatenated as an additional channel with RGB images to vastly outperform previous work. (Chen et al. 2018) proposed an invertible method of parameterizing uniform sparse depth inputs to a deep neural net model that achieved results comparable to conventional depth sensors.

2.2 Dataset

2.2.1 NYU-Depth V2 Dataset

The NYU-Depth V2 Dataset(Nathan Silberman and Fergus 2012) is comprised of RGB and depth image pairs of 464 unique indoor scenes collected with a Microsoft Kinect V1 sensor with a range of 10 meters. The official train/validation split of 249/215 scenes is used, with the original labelled test of 654 images for benchmarking model performance. This is the primary dataset used for benchmarking results in this thesis.

While the original images are of size 640x480, the dimensions used in this thesis are 304x228 following the methods of (Laina et al. 2016; Eigen, Puhrsch, and Fergus 2014).

2.3 Metrics

Let the ground truth depths for an image with N pixels be represented with y and the depth predictions of a model be represented with $\hat{y}$. To check the performance improvements of one technique compared to another, the depth estimation community uses a fixed set of easily explainable metrics that quantify performance across the entire image, with each metric being explained in detail later in this section. Each of these metrics uses a unique formulation combining y and $\hat{y}$ to explain a unique performance aspect of the model in question. The standard list of metrics used are:

- Root Mean Square Error
- Mean Absolute Relative Error
- δ_1, δ_2, δ_3

2.3.1 Root Mean Square Error

$$\sqrt{\frac{1}{N}\sum[\hat{y}-y]^2} \tag{2.1}$$

The Root Mean Square Error(RMSE) is an always non-negative metric used to check the fit of the prediction with the ground truth data where a **lower** value indicates a better model fit. A value of 0, which is almost never seen in practice, indicates a perfect fit with the value you want your model to achieve.

Since RMSE is the square root of the square of the difference between the predicted value and the ground truth value, this metric is **very sensitive to outliers** as values that tend to be close to each other have an extremely small effect on the overall metric value. Therefore, this metric is a great way to explain the model's performance with outliers and hard to classify regions in the image.

When used as a loss for a model, this metric ensures that a model heavily penalizes outlier predictions, but it doesn't ensure closeness of fit as most of the predictions that are relatively close to each other will tend to have similar values after taking the square root of the squared difference. RMSE is a good choice for a loss when it is critical to get good performance on outliers.

2.3.2 Mean Absolute Relative Error

$$\frac{1}{N}\sum\left(\frac{|\hat{y}-y|}{y}\right) \tag{2.2}$$

The Mean Absolute Relative Error(REL) is an always non-negative metric used to check the fit of the prediction to the ground truth data where a **lower** value indicates a better model fit. A value of 0, which is almost never seen in practice, indicates a perfect fit with the value you want your model to achieve.

Since REL is the absolute difference between the predicted value and the ground truth value divided by the ground truth value, this metric provides the **average performance across all pixels** as the metric doesn't treat outliers differently from inliers. Therefore, this metric is great way to explain the model's overall performance across the image.

When used as a loss for a model, this metric ensures that a model works towards getting a **good fit**. This allows model predictions to come closer to the ground truth value compared to RMSE as small differences are not suppressed further(Willmott and

Matsuura 2005). REL is a good choice for a loss when it is critical to get predictions that are extremely close to the ground truth values.

2.3.3 Relative Distance Based Metrics (δ_1, δ_2, δ_3)

$$\delta_i = \frac{\mathrm{card}\left(\left\{\hat{y}_i : \max\left\{\frac{\hat{y}_i}{y_i}, \frac{y_i}{\hat{y}_i}\right\} < 1.25^i\right\}\right)}{\mathrm{card}\left(\{y_i\}\right)} \tag{2.3}$$

The δ_i metrics are use to compare the **ratio of pixels correctly labelled within a maximum allowed relative error** 125^i. A **higher** value is better. Using the three δ metrics simultaneously allows to compare model predictions within different scalar ranges. Therefore, these metrics are helpful to understand if our model predictions are close to the ground truth values, without putting on emphasis on how close the values are.

Chapter 3

FEATURE BASED SPARSE DEPTH SAMPLING

Existing state-of-the-art methods that utilize sparse depths have shown exciting results far surpassing previous benchmarks on the NYU-Depth V2 and KITTI(Geiger, Lenz, and Urtasun 2012) datasets. While these results are very promising, one issue that can be raised is that such methods can't be effectively used in a camera-only system as the sparse depths used by these methods tend to be uniformly spread out throughout the image. Such sparse patterns are definitely possible to be obtained from a sensor like a LiDAR but it is unrealistic to expect the same from a camera. This is because the sparse patterns from camera-only systems tend to be clustered around distinct features like edges in an image, and are seldom uniformly spread throughout the image as assumed by these works. Therefore, this thesis focuses on extracting performance from more realistic feature-based sparse patterns that can be obtained from cameras by leveraging methods similar to those employed in computer graphics research and concepts from active learning(Cohn, Atlas, and Ladner 1994).

3.1 Feature Detection Background

Feature detection is a low-level image processing operation that is usually the first step of more complex computer vision algorithms like Visual Odometry algorithms used for Simultaneous Localization And Mapping(SLAM). Therefore, having a good feature detector is usually a prerequisite to good performance for algorithms that contain a feature detection module as this module usually guides an algorithm only towards regions in an image where a feature is detected.

Since the definition of a “feature” is ambiguous, the kind of feature detected by a detector varies between applications. Traditionally, features detected by camera only systems that perform sparse depth estimation through SLAM algorithms use feature detectors that classify edges and descriptive shapes as a feature. Consequently, in this thesis, we focus on generating sparse depth patterns similar to those generated by camera SLAM algorithms, where sparse depth information can be obtained from a scene by locating edges and descriptive features.

To mimic the feature detection operation performed by SLAM algorithms, we explore two feature detection algorithms:

- Scale Invariant Feature Transform(Lowe et al. 1999)
- Oriented FAST and Rotated BRIEF(Rublee et al. 2011)

3.1.1 Scale Invariant Feature Transform

The Scale Invariant Feature Transform(SIFT) is a pioneering feature detection algorithm to describe local features detected in an image. It is widely used in a diverse variety of applications, not limited to mapping and navigation, image stitching, tracking, and object recognition.

SIFT works by providing *feature descriptions* to any interesting features detected in an object in an image. These extracted feature descriptions, called *descriptors*, are stored in a database so that they can then be used to identify the object in other images. These descriptors are local and are solely based on the appearance of the object. The highly distinctive nature of the detected features make it easy to extract and reduce chances of mismatch with other features in a database.

Since it is imperative that these features be detected independent of the orientation of the object in the new image, the detected features tend to be along the edges of the object as they are high contrast/frequency regions. This allows SIFT to provide strong feature detection even when objects of interest are occluded or surrounded by other objects as SIFT descriptors are invariant to uniform scaling and orientation.

3.1.2 Oriented FAST and Rotated BRIEF

While SIFT provides a solid and well tested feature detection offering, the algorithm as-is isn't frequently used compared to other alternatives as it was patented upon its inception in 1992. Therefore, one popular fast and efficient alternative to SIFT that is widely used in most modern computer vision applications is Oriented FAST and rotated BRIEF(ORB).

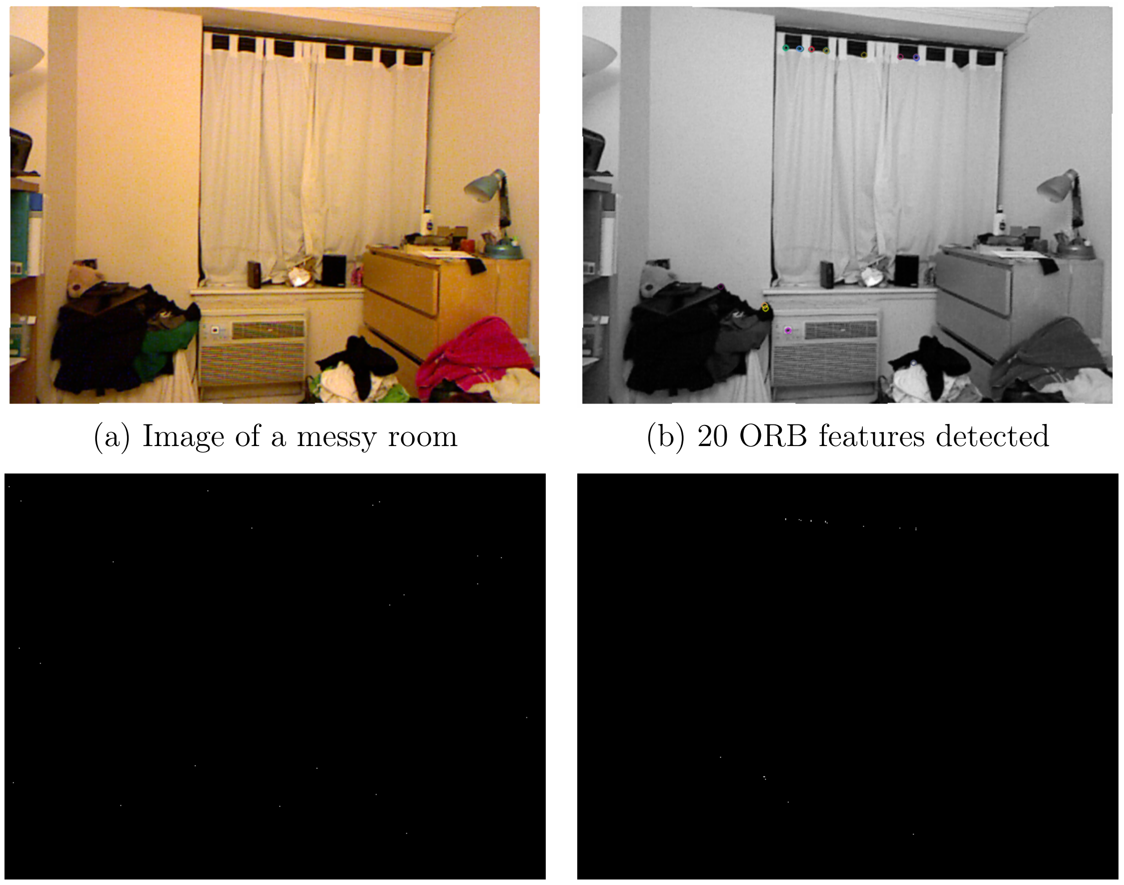
ORB uses a very fast binary descriptor that is rotation invariant and resistant to noise to deliver comparable performance to SIFT at a speed increase of around two orders of magnitude in most cases. This allows it to be used a viable alternative to SIFT for real time algorithms like visual odometry by removing the large computational burden that comes with SIFT.

3.2 Sparse Sampling Methods Comparison

The depth sampling of sparse depths in existing state-of-the-art work is performed by assigning a Bernoulli probability of sampling to every pixel in the ground truth depth. If we want to sample m points from the ground truth depth, the Bernoulli sampling probability of each valid depth pixel is $p = \frac{m}{n}$, where n is the number of valid depth pixels in the ground truth depth. This sampling strategy results in a quite uniform, grid-shaped distribution of sparse samples across the image and the selection of a number of pixels that is around the expectation m. This sampling strategy can also be used as a data augmentation technique as different sparse patterns can be obtained for a unique RGB image as **this sparse depth sampling technique is independent of the RGB image**.

While the above-mentioned sampling strategy is flexible with the number of sparse samples and is robust to targeted noise as the pattern is spread throughout the image, it isn't a realistic approximation of camera-only systems that extract depth information from detected feature points as described above. To that end, the sparse sampling strategy used in this thesis is feature-based and not probabilistic in nature. This results in **a sparse sampling technique that is dependent on the RGB image**. The sampling strategy is straightforward; if we want to sample m points from the ground truth depth, an ORB feature detector is run to detect m features in the RGB image after which the depth information corresponding to the features selected are sampled. This results in sparse patterns that correspond to the edges of objects and around other descriptive features. While the previous sampling strategy resulted in a uniformly spread out, grid-shaped pattern of sparse samples, this method results in sparse samples that are in isolated clusters. The differences between these sparse depth patterns become more pronounced as the number of sparse depth points sampled, m, increases. This can be visualized in the figures of the next subsection where the probabilistic sparse sampling strategy is compared against the feature-based sampling strategy for the **same number of sparse samples** across three random RGB images taken from the NYU-D V2 dataset. **The clustered nature of feature-based sparse sampling becomes increasingly obvious as the number of samples increases.**

3.2.1 Visual Comparison With Increasing Number of Sparse Samples



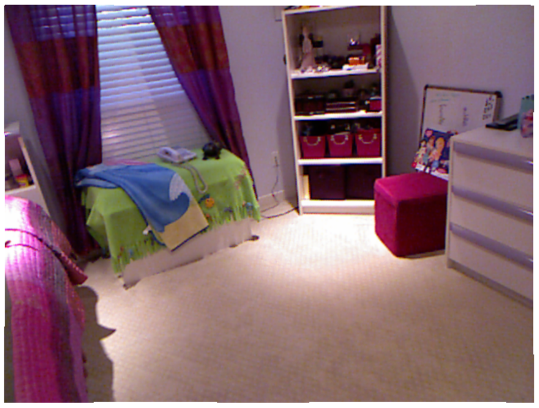
(a) Image of a messy room

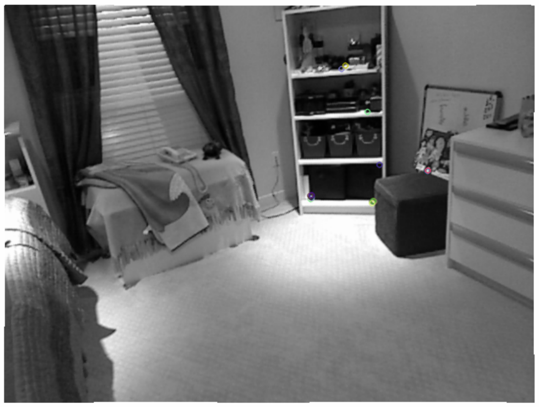
(b) 20 ORB features detected

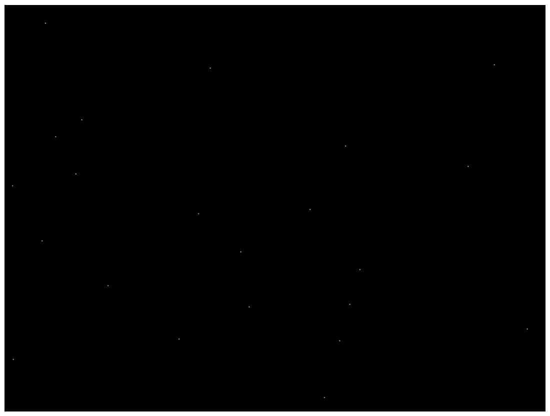
(c) Probabilistic Sparse Sampling

(d) Feature-based Sparse Sampling

Figure 1: Sparse Sampling comparison for 20 samples in a messy room

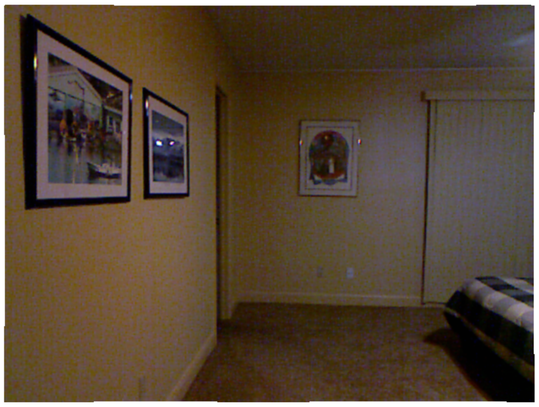

(a) Image of a clean room

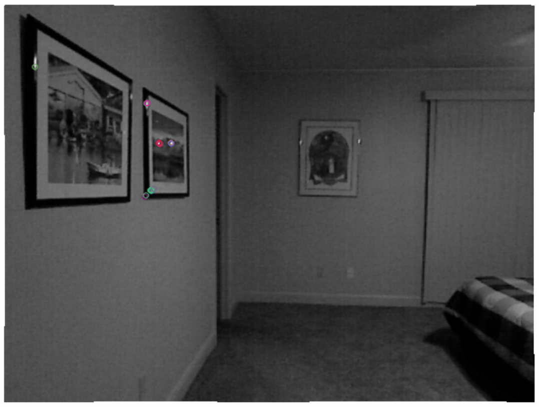

(b) 20 ORB features detected


(c) Probabilistic Sparse Sampling

(d) Feature-based Sparse Sampling

Figure 2: Sparse Sampling comparison for 20 samples in a clean room


(a) Image of a room with paintings


(b) 20 ORB features detected

(c) Probabilistic Sparse Sampling

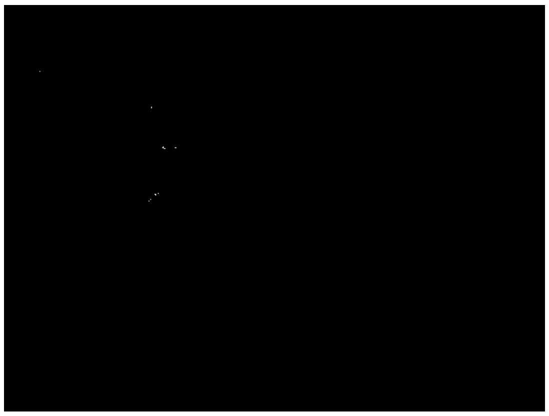
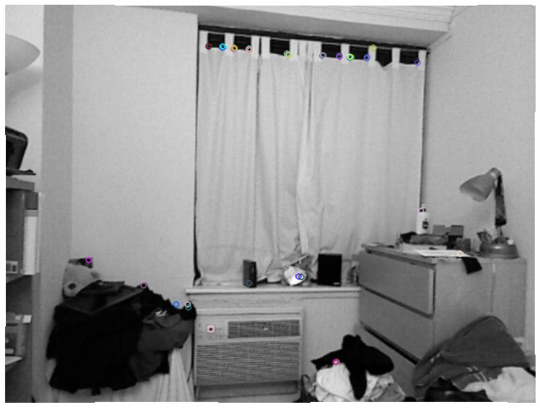
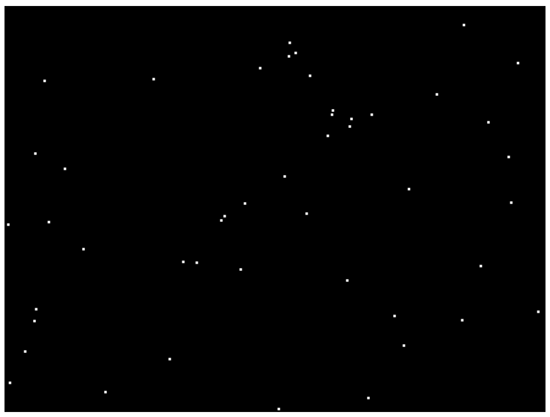

(d) Feature-based Sparse Sampling

Figure 3: Sparse Sampling comparison for 20 samples in a room with paintings

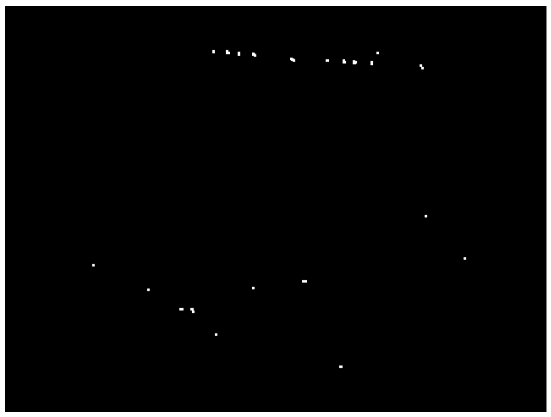
(a) Image of a messy room



(b) 50 ORB features detected

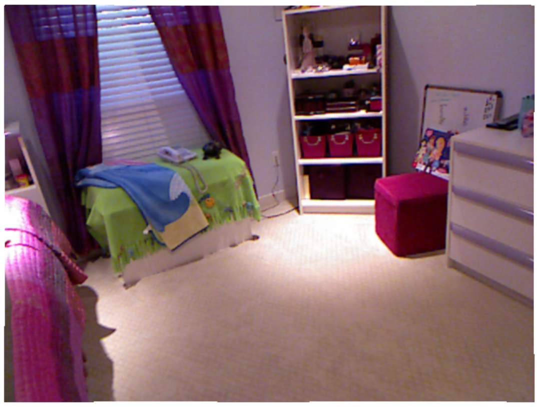


(c) Probabilistic Sparse Sampling

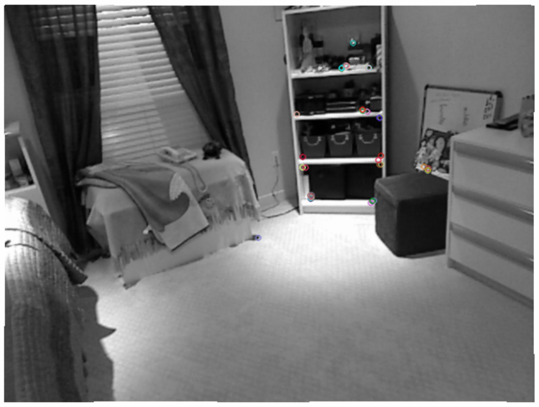


(d) Feature-based Sparse Sampling

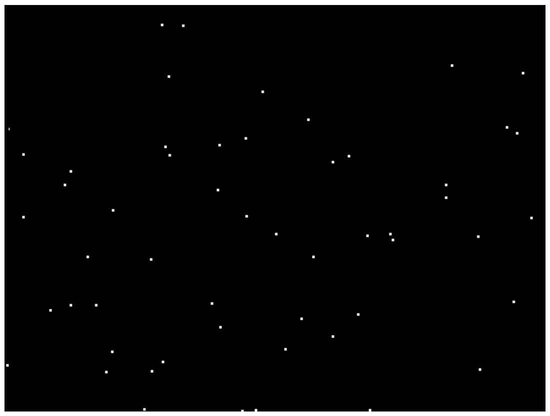
Figure 4: Sparse Sampling comparison for 50 samples in a messy room



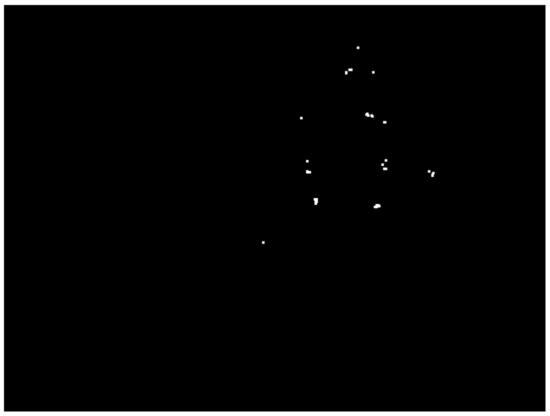
(a) Image of a clean room



(b) 50 ORB features detected

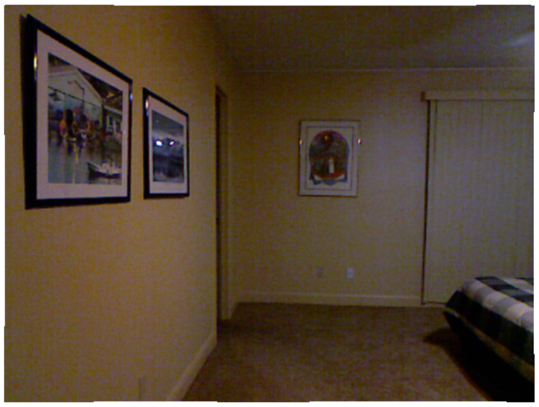


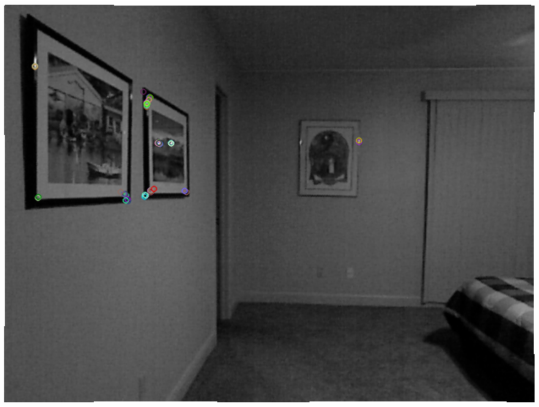
(c) Probabilistic Sparse Sampling

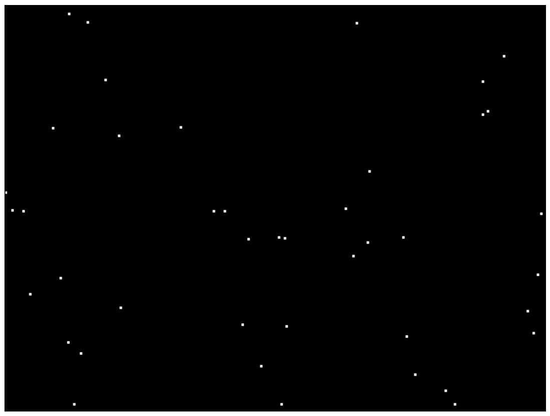


(d) Feature-based Sparse Sampling

Figure 5: Sparse Sampling comparison for 50 samples in a clean room


(a) Image of a room with paintings


(b) 50 ORB features detected


(c) Probabilistic Sparse Sampling

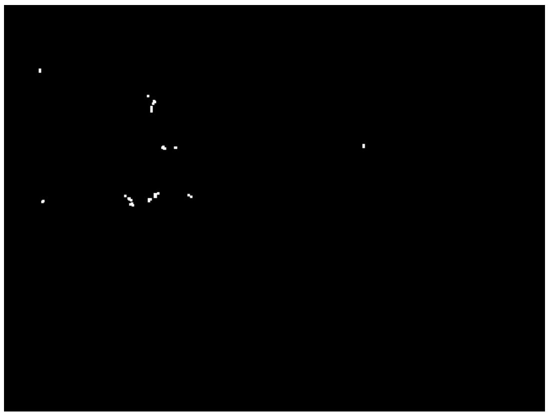
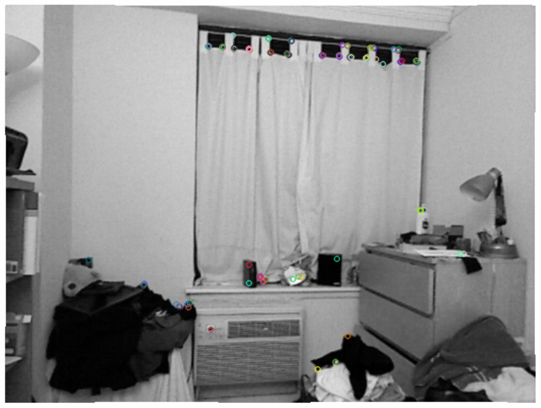
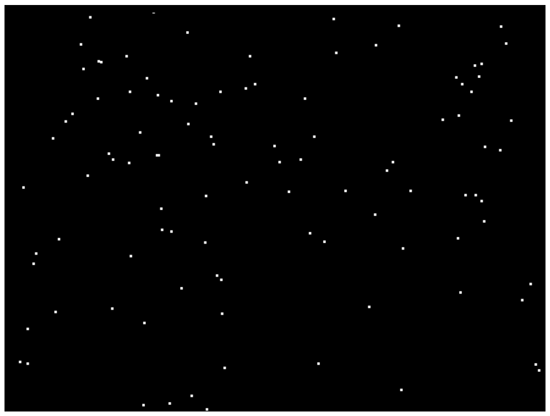

(d) Feature-based Sparse Sampling

Figure 6: Sparse Sampling comparison for 50 samples in a room with paintings

(a) Image of a messy room


(b) 100 ORB features detected


(c) Probabilistic Sparse Sampling

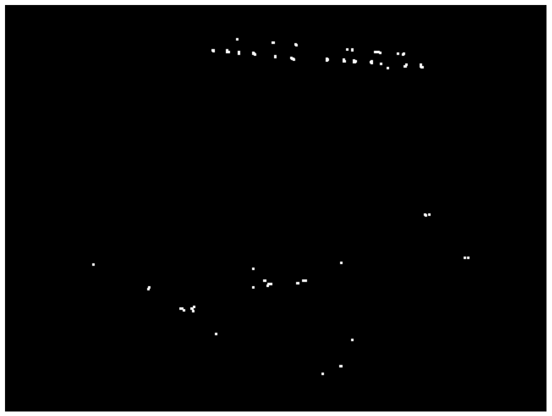
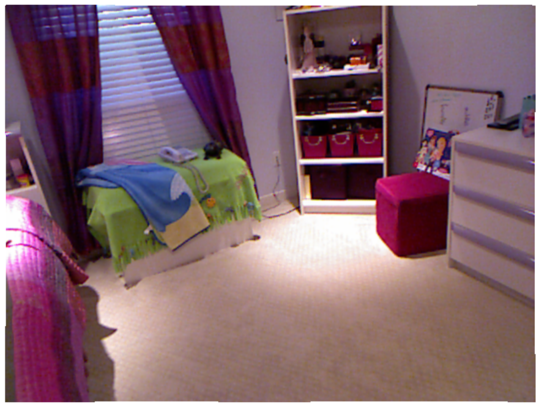
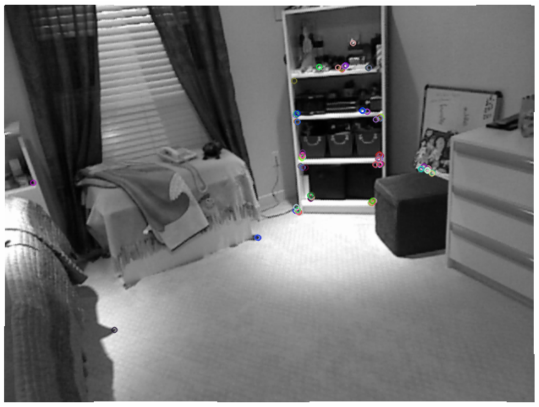
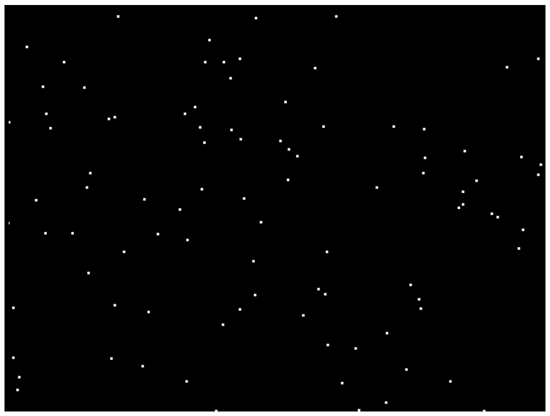

(d) Feature-based Sparse Sampling

Figure 7: Sparse Sampling comparison for 100 samples in a messy room


(a) Image of a clean room


(b) 100 ORB features detected


(c) Probabilistic Sparse Sampling

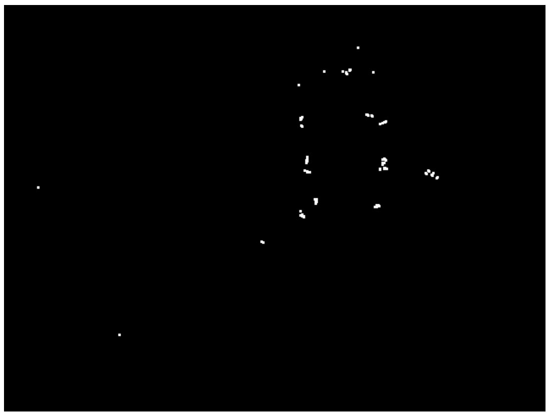
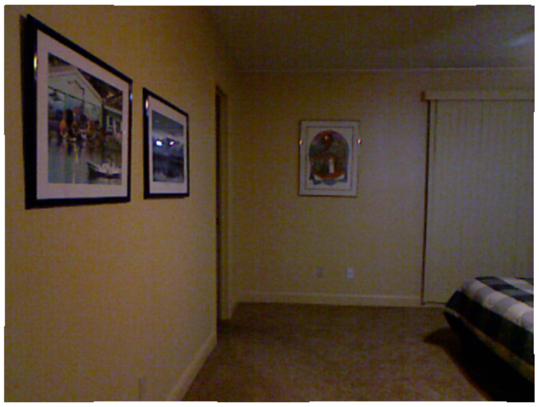
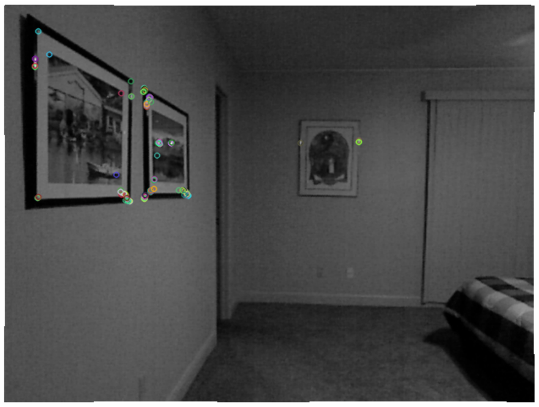
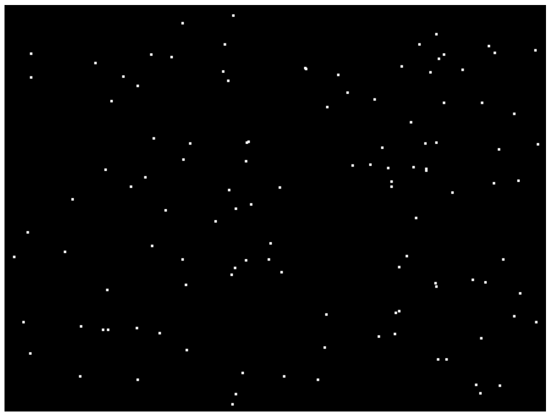

(d) Feature-based Sparse Sampling

Figure 8: Sparse Sampling comparison for 100 samples in a clean room


(a) Image of a room with paintings


(b) 100 ORB features detected


(c) Probabilistic Sparse Sampling

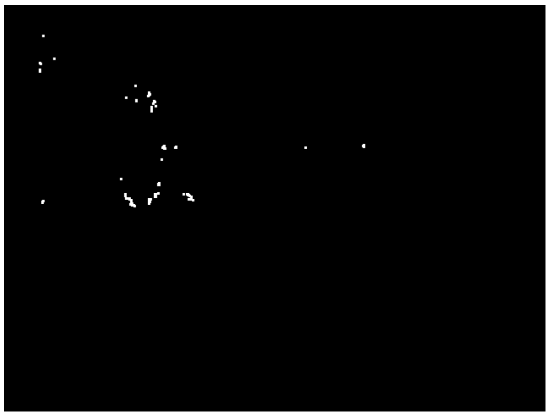
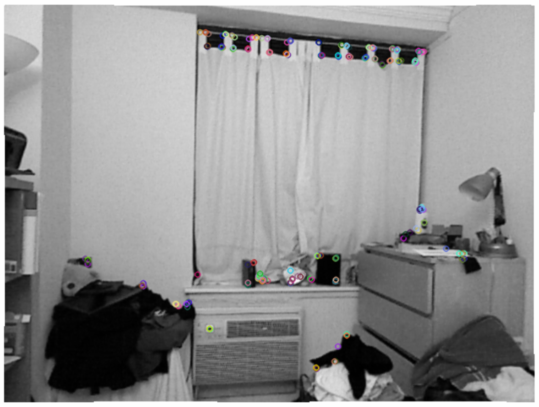
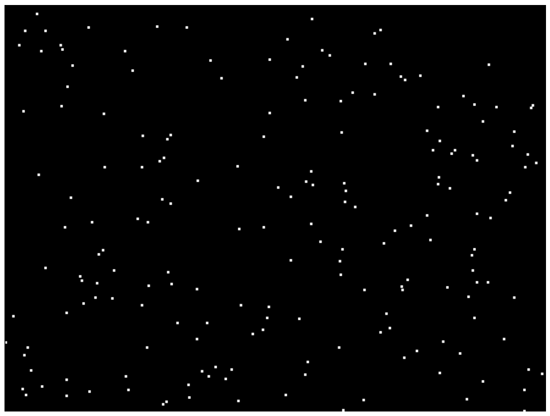

(d) Feature-based Sparse Sampling

Figure 9: Sparse Sampling comparison for 100 samples in a room with paintings

(a) Image of a messy room


(b) 200 ORB features detected


(c) Probabilistic Sparse Sampling

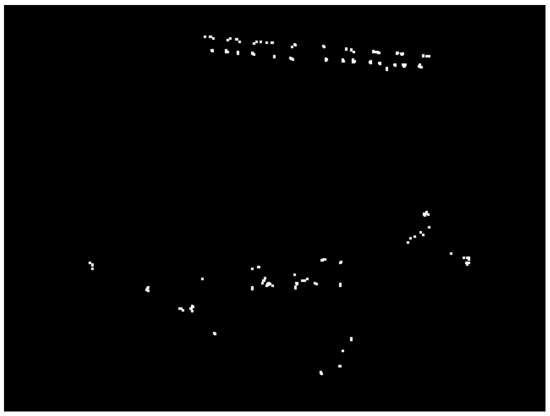
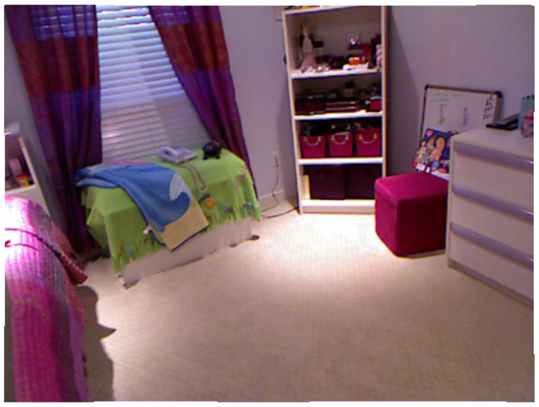
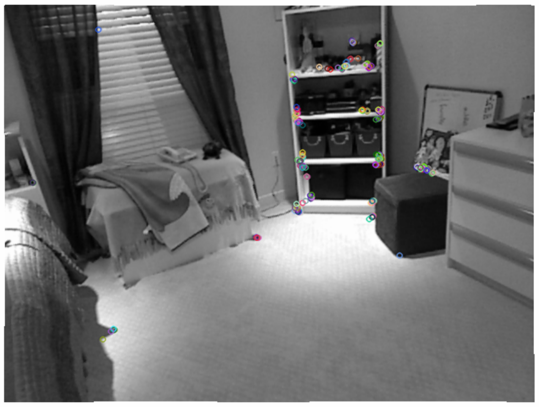
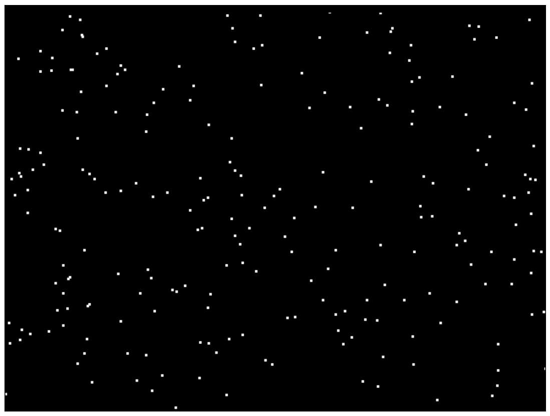

(d) Feature-based Sparse Sampling

Figure 10: Sparse Sampling comparison for 200 samples in a messy room


(a) Image of a clean room


(b) 200 ORB features detected


(c) Probabilistic Sparse Sampling

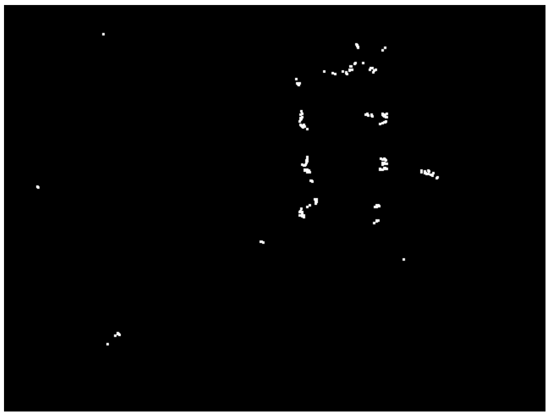
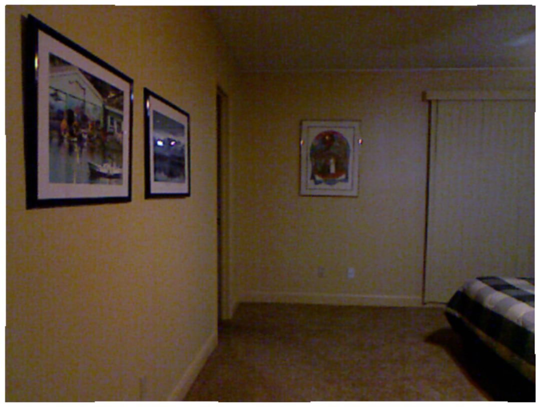
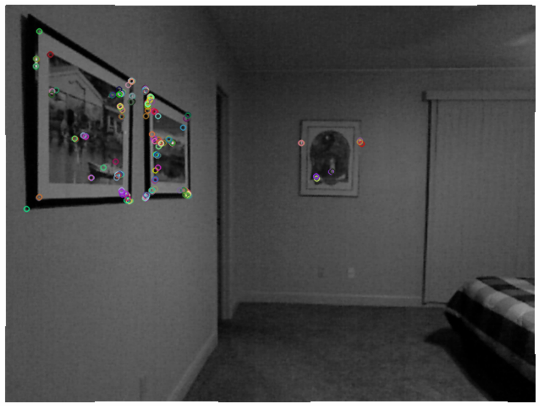
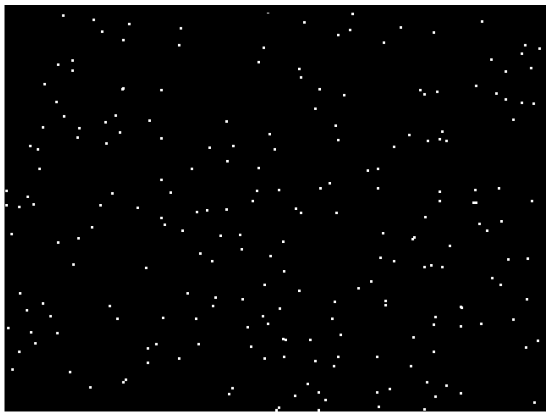

(d) Feature-based Sparse Sampling

Figure 11: Sparse Sampling comparison for 200 samples in a clean room


(a) Image of a room with paintings


(b) 200 ORB features detected


(c) Probabilistic Sparse Sampling

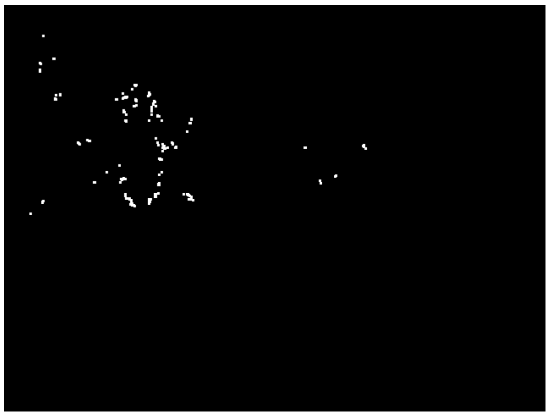

(d) Feature-based Sparse Sampling

Figure 12: Sparse Sampling comparison for 200 samples in a room with paintings

Chapter 4

SPARSE DEPTH INTERPOLATION AND ACTIVE LEARNING FRAMEWORK

Once the feature-based sparse depth map is obtained, it can be concatenated as an additional channel along with the three channels of an RGB image into the encoder-decoder architecture that was used in the state-of-the-art work. While doing this allows us to set up a baseline performance threshold, it is not interesting enough in its novelty to be counted as an important contribution to the field of depth estimation. To this end, this thesis focuses on exploiting the clustered nature of the feature-based sampled sparse depths. While the clustered nature of sampled sparse depths of a feature-based approach does not provide the robustness of a probabilistic, grid-shaped sparse sampling strategy where the uniform distribution of the sampled depths provides some prior knowledge of every region in an image, it provides us the ability to utilize estimation techniques to increase the number of sparse depth points for which we have prior information. This is done by exploiting the proximity of sampled sparse depth points in the feature-based sampling strategy by interpolating sparse depth information after using an estimation technique similar to ones used in the computer graphics community. These interpolated sparse depths not only provide more information about pixels in distinct regions of an image, they also enable us to incorporate an *active learning* framework to further improve performance of the network by focusing on pixel predictions that deviate the most from the estimated interpolated depth, thereby acting as a constraint without using leaking ground truth depth information to the network.

4.1 Interpolation Methodology

To interpolate sparse depth information of unknown pixels that are enclosed by distinct clusters of the sampled feature-based sparse depths, we use an estimation technique for interpolation that is similar to those used in computer graphics for mesh-based triangulation. This interpolation is done by drawing unique triangles between uniques sets of sampled sparse depth points that are in the same cluster. To get the depth information of unknown pixels that lie on the same triangular plane we solve the equation of the plane that connects the three vertices of the triangle. Finally, we obtain the estimated depth information for all unknown pixels that lie within the same cluster by solving the equation of the plane on which the unknown pixel lies on. Interpolation of the sparse depths to get additional estimated depth information provides multiple benefits: Firstly, it vastly increases the count of pixels for which we have some kind of depth information, and this leads to a more robust depth estimation prediction. Secondly, it enables the application of the active learning paradigm; the estimated sparse depth information obtained from solving the plane equations can be used as an additional constraint during neural network training without explicitly using ground truth depths.

4.1.1 Triangulation Methodology And Visual Interpolation Results

The process of drawing a mesh composed of distinct triangles between unique sampled sparse depth sets is the key prerequisite to performing the interpolation. Prior to this, a method to divide the original sampled sparse depths into distinct clusters must be established. To do this, a **hyperparameter** called *cluster distance* is created, where *cluster distance* is the maximum distance a point can be from another point from the same cluster. Point-wise cluster association is done for every sampled sparse depth point which results in the formation of distinct clusters. Finally, triangulation is performed independently for each cluster.

While there are several triangulation algorithms that can be used to draw unique triangles between three unique sampled sparse depth points in a cluster, in this thesis we use *Delaunay Triangulation*(Delaunay et al. 1934) as it maximizes the minimum angle of all triangles in the triangulation. This property is especially useful for interpolation as it helps to avoid generating triangles with one or two extremely acute angles which leads to thin triangles. Therefore, *Delaunay Triangulation* can break the planar surface connecting points contained within distinct clusters in the sampled sparse depth into a mesh composed of well-shaped triangles.

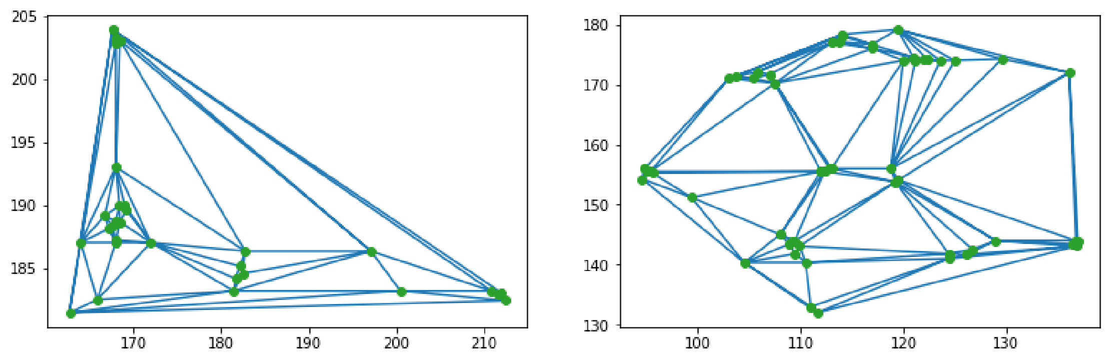
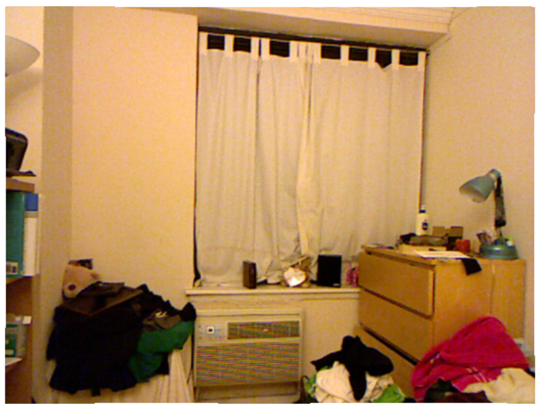
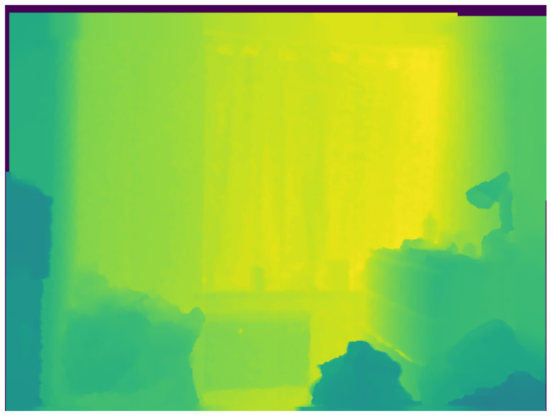
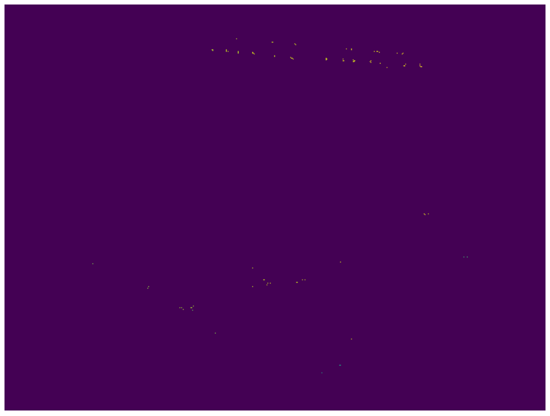


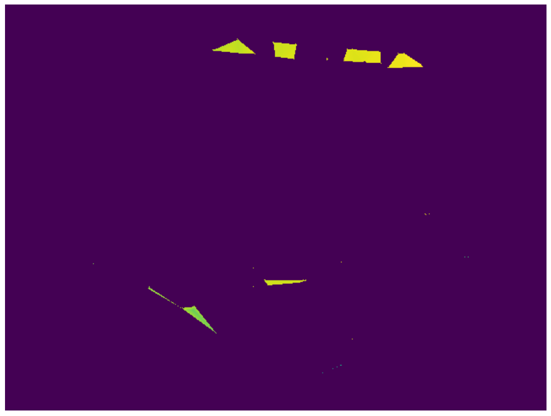
Figure 13: Delaunay Triangulations Obtained From Different Clusters In The Same Feature-based Sparse Sampling

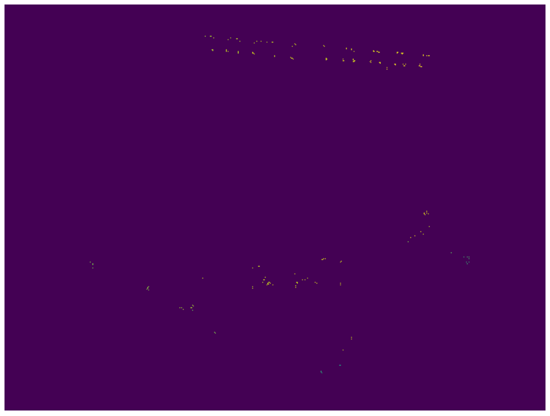
After the triangular mesh of the planar surface of a cluster has been obtained, the estimated depth of unknown points in the cluster is obtained by solving the plane equation of the triangle on which a point lies. This leads to an **extremely significant increase in the number of pixels for which the model has some kind of depth information**.


(a) Image of a messy room


(b) Ground Truth Depths


(c) Sparse Depths From 100 Features


(d) Interpolated Sparse Depths


(e) Sparse Depths From 200 Features

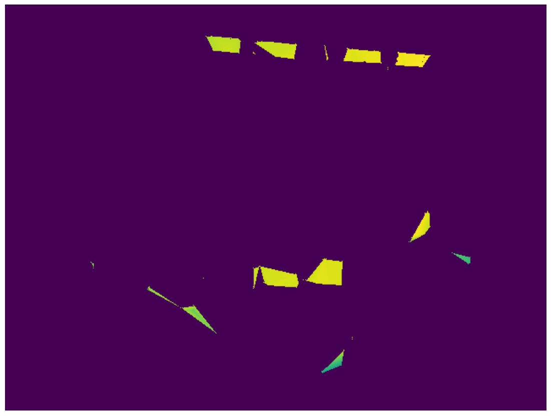
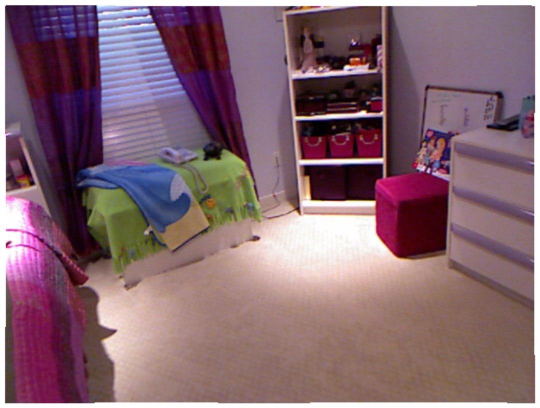
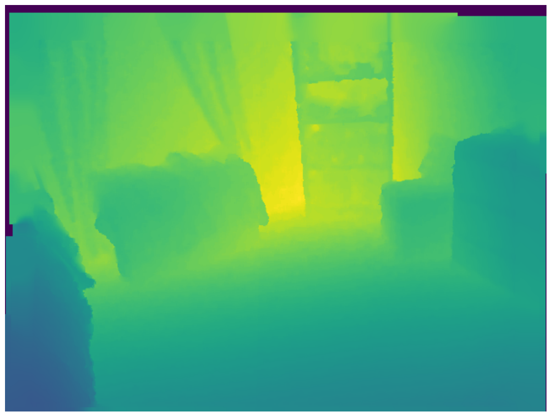
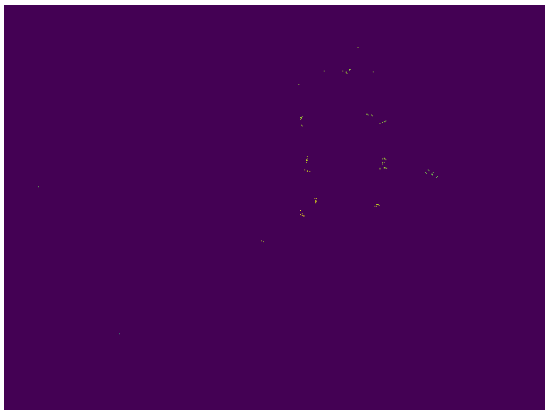
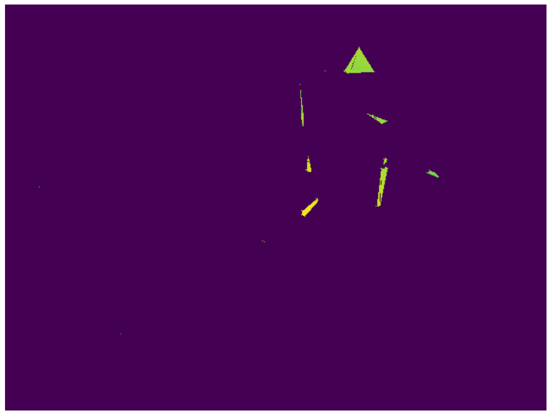

(f) Interpolated Sparse Depths

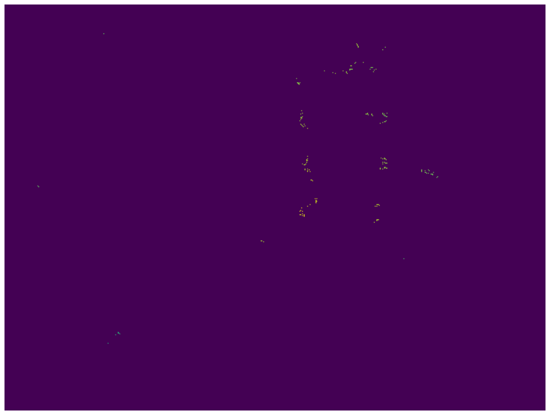
Figure 14: Interpolated Sparse Depths for features detected in a messy room


(a) Image of a clean room


(b) Ground Truth Depths


(c) Sparse Depths From 100 Features


(d) Interpolated Sparse Depths


(e) Sparse Depths From 200 Features

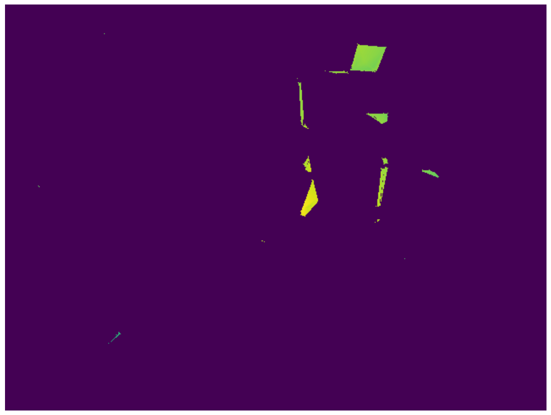
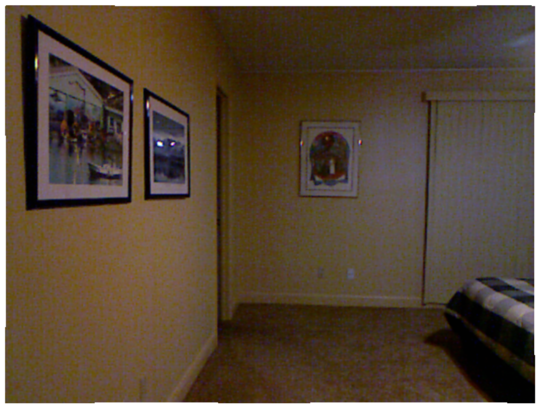
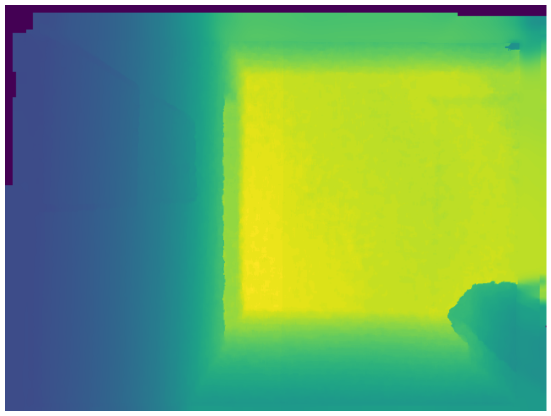
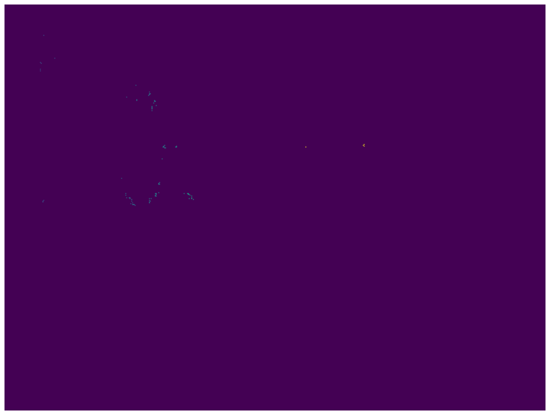
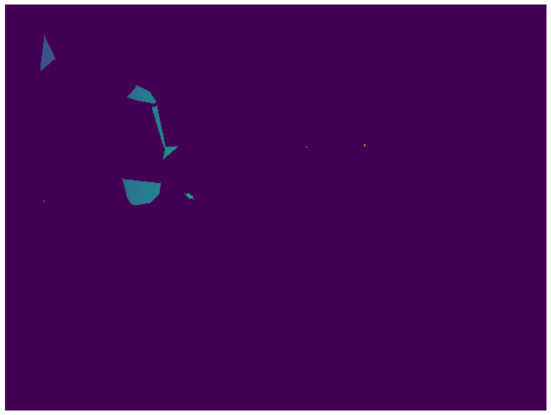

(f) Interpolated Sparse Depths

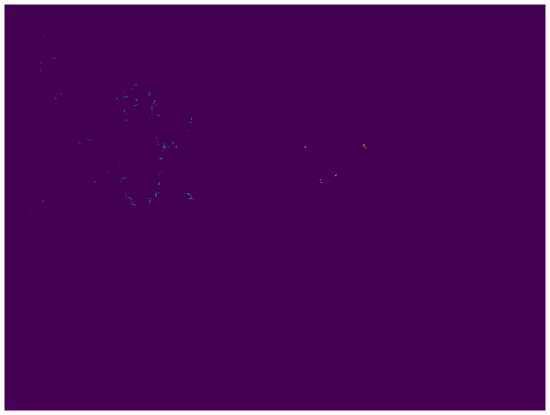
Figure 15: Interpolated Sparse Depths for features detected in a clean room


(a) Image of a room with paintings


(b) Ground Truth Depths


(c) Sparse Depths From 100 Features


(d) Interpolated Sparse Depths


(e) Sparse Depths From 200 Features

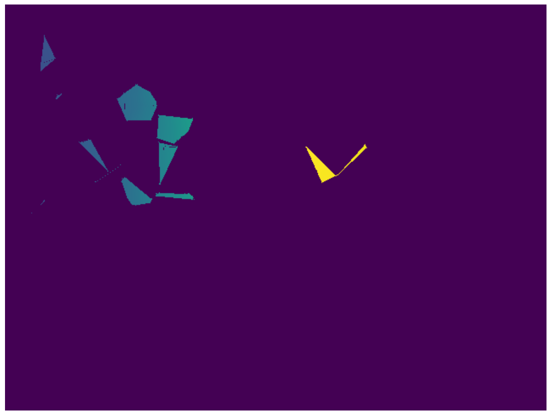

(f) Interpolated Sparse Depths

Figure 16: Interpolated Sparse Depths for features detected in a room with paintings

4.2 Active Learning

Active Learning is a supervised machine learning paradigm where the learning algorithm is provided samples at each iteration to selectively train on, separate from the original training data set. This kind of learning can potentially teach a learning algorithm a concept with a lesser number of examples than the case where the algorithm has to spend multiple additional epochs to learn the same concept. This is possible because some unique samples can provide more valuable information when isolated than when provided in a batch alongside typical data that the model performs well on.

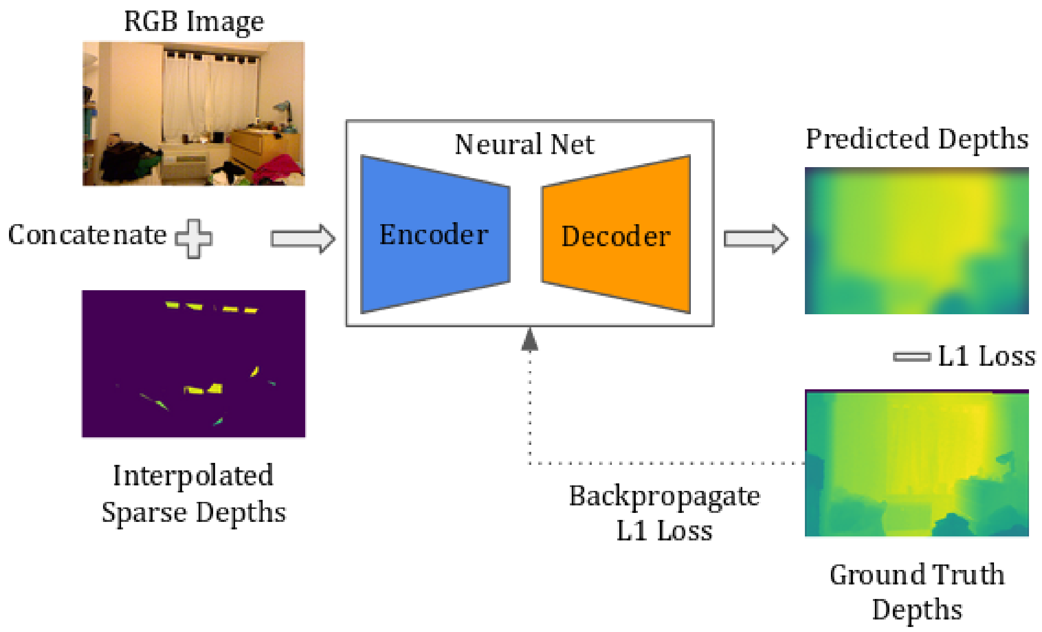
In our setting, we utilize the interpolated depths generated from the original sampled sparse feature-based depths along with the active learning paradigm to selectively train the network on samples that are the hardest to learn. This is done by creating an additional training loop for the model in addition to the one where it trains on the entire training data set. In the new training loop, the generated interpolated sparse depths are utilized as the output that the model must optimize towards. **During every iteration of this training loop, the model is selectively trained on samples that deviate most from the interpolated sparse depths, and not the ground truth depths**.

Here, the estimated interpolated sparse depths are treated as proxy target predictions for the actual target predictions that are the ground truth depths. This assumption is possible because of the feature-based sampling strategy; since the sampled sparse depths correspond to features detected in the RGB image by the ORB feature detector, there is an assurance that each independent cluster of features corresponds to points that lie on roughly the same object surface in the RGB image.

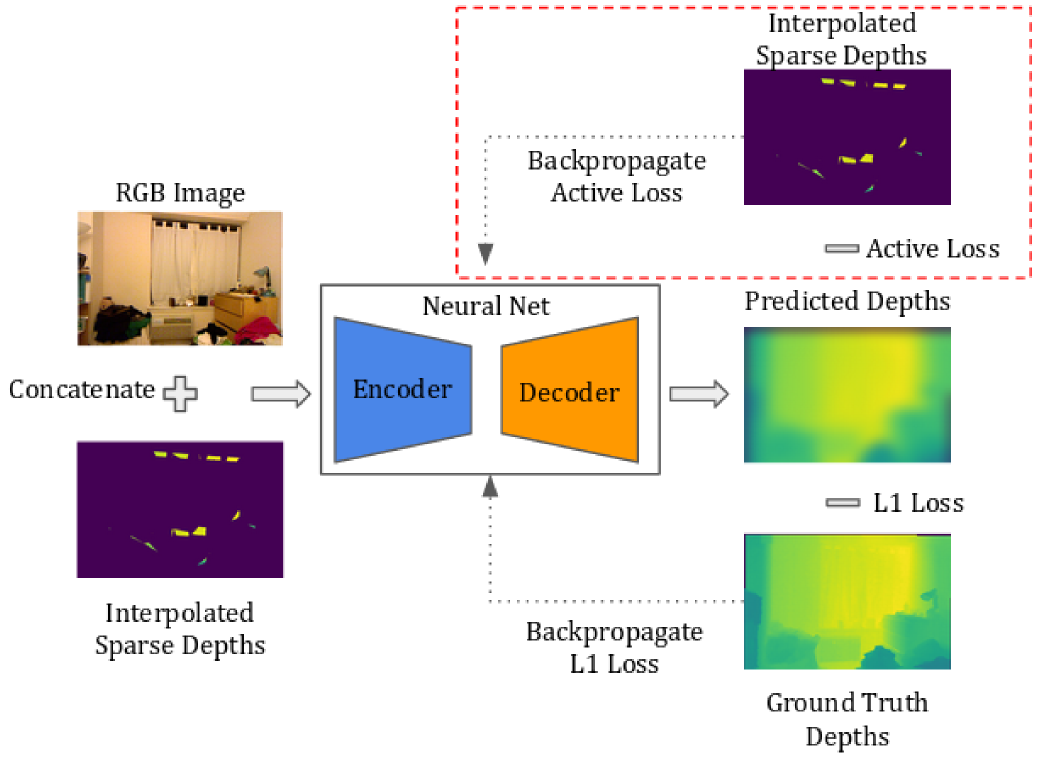
Therefore, it follows that the new interpolated sparse depths obtained from solving the plane equations connecting the original sampled sparse depths for each independent cluster are good approximations for the ground truth depths since the original sampled sparse depths for a cluster lie on roughly the same object surface.

This adds an additional constraint during model training; not only do overall model predictions need to be comparable to the ground truth target depths, but the model must also focus on predictions that were previously outliers. In our setting, outliers tend to be at the object boundaries separating distinct objects in an RGB image. This is because all points that lie on an object are roughly at the same depth, and moving from one object to another leads to a sharp change in depth value. Since these image regions where there is a transition between objects are minuscule in terms of number of pixels compared to the rest of the image, our experiments have shown that existing models focus on improving performance on the other regions of the image at the cost of performance in these regions of object transitions, therefore they can be classified as outliers for the model.

Utilizing the active learning paradigm, therefore, allows the model to selectively train on outliers, with respect to the estimated interpolated depth, during the training loop in addition to the original training process with the ground truth data. This acts as additional training data that the model previously did not have access to as the samples that the model gets to train on during the active learning training loop will change per iteration depending on the predictions that deviate the most from the estimated interpolated sparse depths. Our experiments show that utilizing this paradigm leads to noticeably better model performance across all metrics.



(a) Without Active Learning



(b) With Active Learning

Figure 17: Comparison of Frameworks With and Without Active Learning Framework

Chapter 5

THE EDGE METRIC

Motivated by the results obtained from using the active learning paradigm, a major contribution of this thesis is the formulation of a novel metric, titled the Edge metric, to track the performance of the model on the regions where there are sudden drastic changes in depth values, which tend to be the hardest regions to classify in an image. Such regions tend to be at the boundaries and edges of objects in a scene, as described in Chapter 4. In addition to the metric effectively being able to track performance on object transition regions, we show that existing state-of-the-art work can be improved dramatically by incorporating this new Edge metric as a small component of a model's loss function.

5.1 Drawbacks of Existing Metrics

While existing metrics that are used to track performance in depth estimation do a good job of reporting performance across the entire image as described in 2, they do not specifically track performance at a region of interest in an image. Therefore, while they are great at understanding how well a model is performing over the entire image, they do not give a reader any information about how specific parts of an image are affecting the overall metric.

For example, while the value of the Root Mean Square Error(RMSE) can give the reader an indication of the performance of a model on all outliers in an image, it doesn't explicitly focus on the performance of the model on the highest contributing outliers. This is where the Edge metric truly shines, as it is explicitly modeled to track the performance of the model on regions of an image that intuitively should be the biggest contributors to all the other primary metrics. Therefore, optimizing towards a better Edge metric score in most cases will also lead to better performance in other metrics as outliers are usually the biggest contributors to every metric.

5.2 Methodology

We make use of the *image gradient* values of every pixel in the ground truth depth of an image, which can be defined as the difference in depth value of a pixel and its preceding pixel. The process to compute the Edge metric value of an image can be broken down into the following steps:

- Regions in the ground truth depth image that show non-negligible changes in the *image gradient* along either the x-axis or the y-axis are identified.
- These *image gradient* values are modeled as a normal distribution to compute the mean and standard deviation.
- Special focus is placed on pixels that exhibit an *image gradient* value that is present in the tails of the modeled normal distribution. In this case, only the *image gradient* values along **either** the x-axis or the y-axis whose absolute value is greater than the mean *image gradient* value of the image plus twice the standard deviation of the normal *image gradient* value distribution of the image are considered. All the other pixels are ignored.
- Finally, the Mean Absolute Error between the ground truth depth and the predicted depth is computed, but only at these pixels in the ground truth depth image that have the highest *image gradient*s.

5.3 Incorporating With Traditional Loss Functions

The experiments in this thesis show that incorporating the Edge metric as a loss to existing loss function like $L1$ loss leads to improved performance of the state-of-the-art model on the Edge metric and most of the other primary metrics as well. The new loss is formulated as

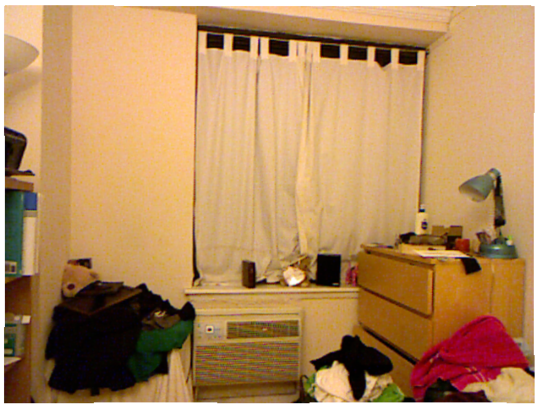
$$\theta * l1_{loss} + (1 - \theta) * edge_{loss} \tag{5.1}$$

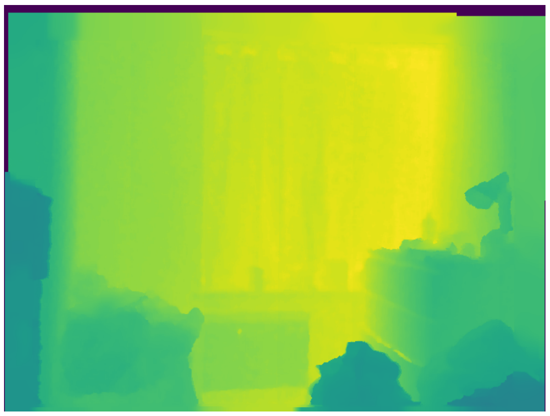
where $l1_{loss}$ is the $L1$ loss, $edge_{loss}$ is the loss from the Edge metric, and θ is the hyperparameter controlling the contribution of each individual loss.

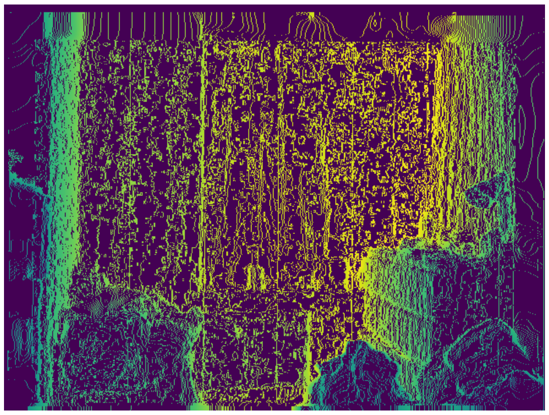
PyTorch(Paszke et al. 2017) code snippet implementing the forward pass of the new proposed loss:

```
def forward(self, pred, target, theta=0.90):
        valid_mask = (target >0).detach()
        diff = target - pred
        diff = diff[valid_mask]
        l1_loss = diff.abs().mean()

        def image_gradients(image):
            image_shape = image.shape
            batch_size, depth, height, width = image_shape
            dy = image[:, :, 1:, :] - image[:, :, :-1, :]
            dx = image[:, :, :, 1:] - image[:, :, :, :-1]
            shape = [batch_size, depth, 1, width]
            dy = torch.cat([dy,
                torch.zeros(shape,
                dtype=torch.float).cuda()],
                dim=2)
            dy = torch.reshape(dy, image_shape)
            shape = [batch_size, depth, height, 1]
            dx = torch.cat([dx,
                torch.zeros(shape,
                dtype=torch.float).cuda()],
                dim=3)
            dx = torch.reshape(dx, image_shape)
            return dy, dx

        def edge_mask(dy, dx):
            dy_mask_greater = dy > dy.mean() + 2*dy.std()
            dy_mask_lesser = dy < dy.mean() - 2*dy.std()
            dy_mask = dy_mask_greater | dy_mask_lesser
            dx_mask_greater = dx > dx.mean() + 2*dx.std()
            dx_mask_lesser = dx < dx.mean() - 2*dx.std()
            dx_mask = dx_mask_greater | dx_mask_lesser
            mask = dy_mask | dx_mask
            return mask

        dy_true, dx_true = image_gradients(target)
        mask = edge_mask(dy_true, dx_true)
        target_edges = target[mask]
        pred_edges = pred[mask]
```

```
edge_abs_diff = (target_edges - pred_edges).abs()
self.loss = theta*l1_loss
    + (1-theta)*edge_abs_diff.mean()
return self.loss
```


(a) Image of a messy room


(b) Ground Truth Depths


(c) Image Gradients along x-axis

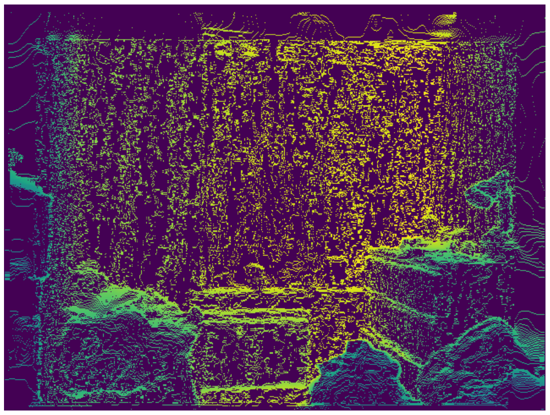
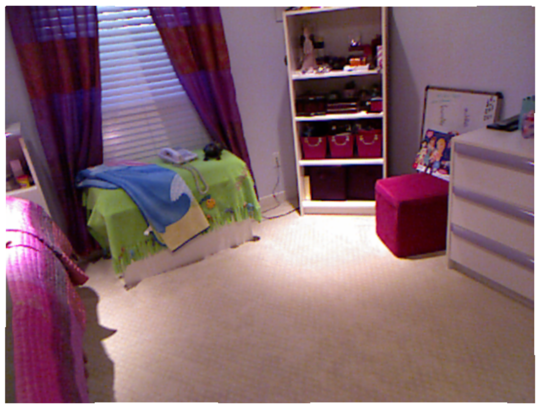
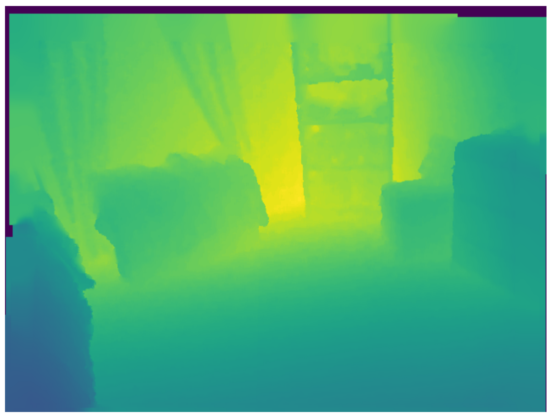
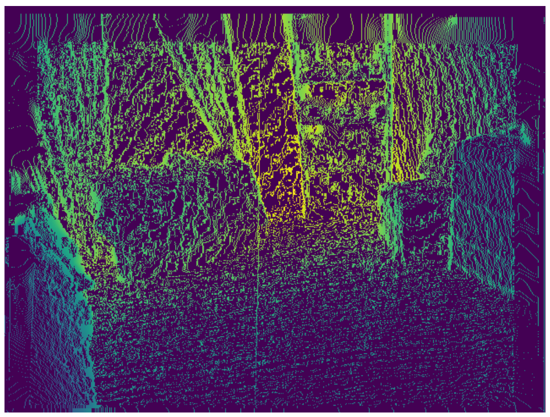

(d) Image Gradients along y-axis

Figure 18: Image Gradients of a messy room


(a) Image of a clean room


(b) Ground Truth Depths


(c) Image Gradients along x-axis

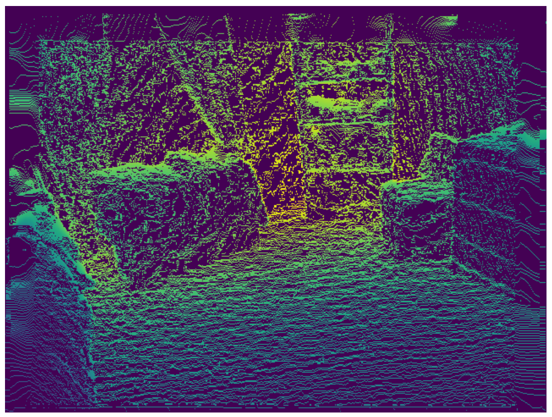
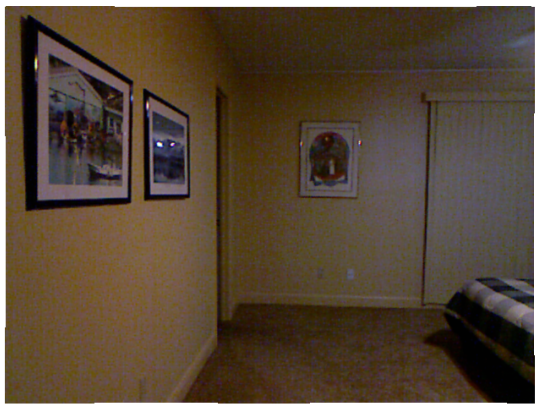
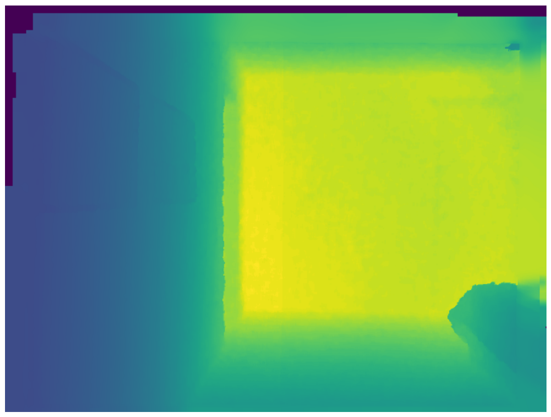
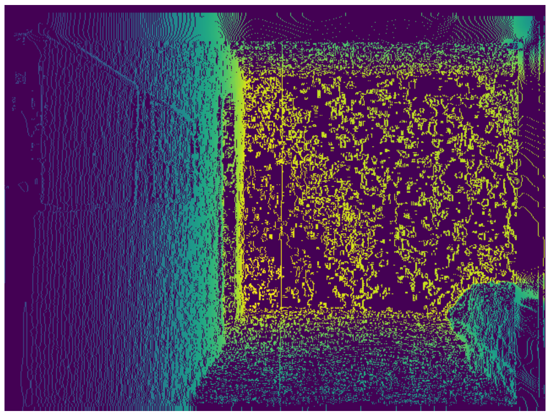

(d) Image Gradients along y-axis

Figure 19: Image Gradients of a clean room


(a) Image of a room with paintings


(b) Ground Truth Depths


(c) Image Gradients along x-axis

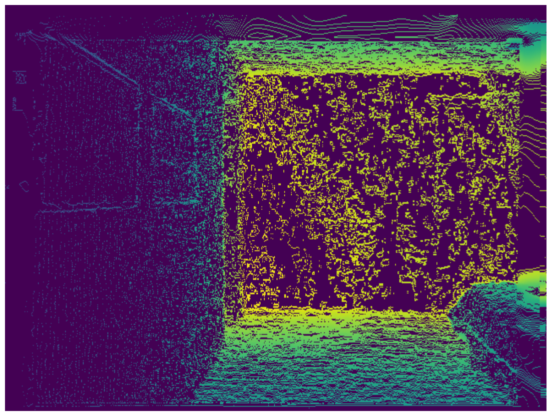
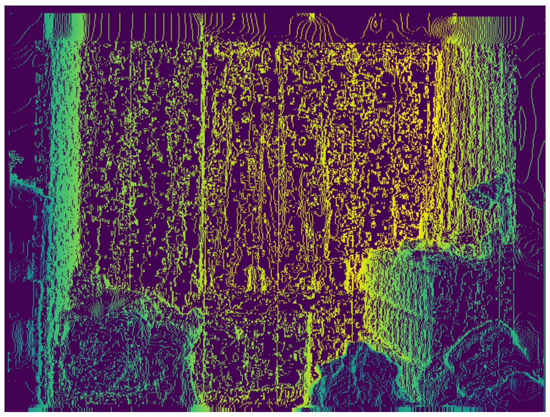
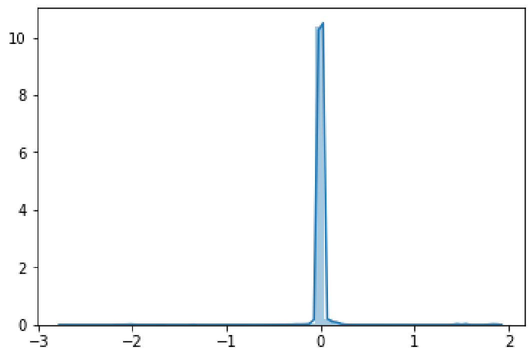
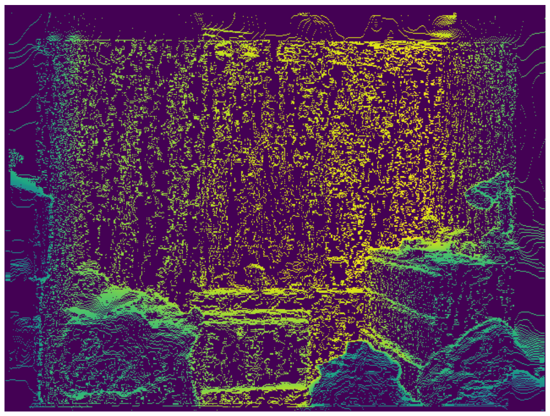

(d) Image Gradients along y-axis

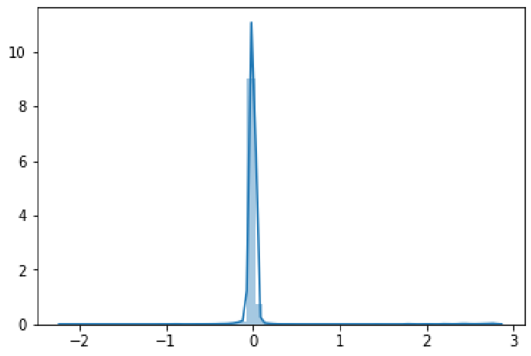
Figure 20: Image Gradients of a room with paintings


(a) Image Gradients along x-axis



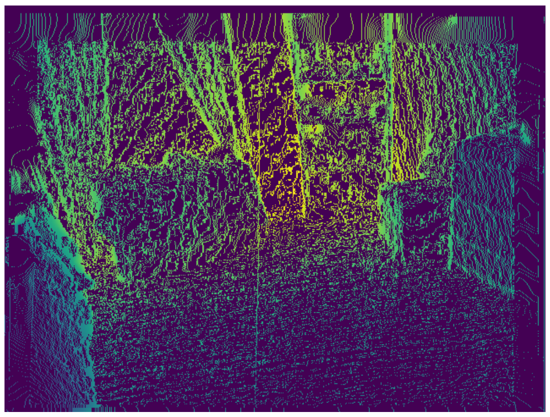
(b) Distribution along x-axis

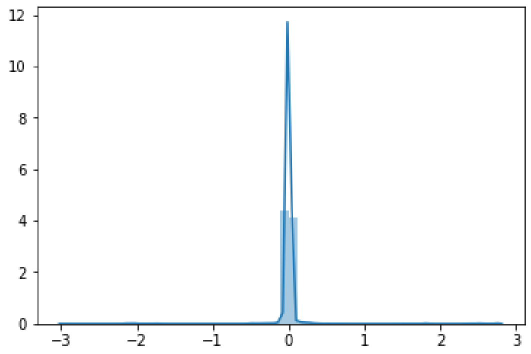

(c) Image Gradients along y-axis



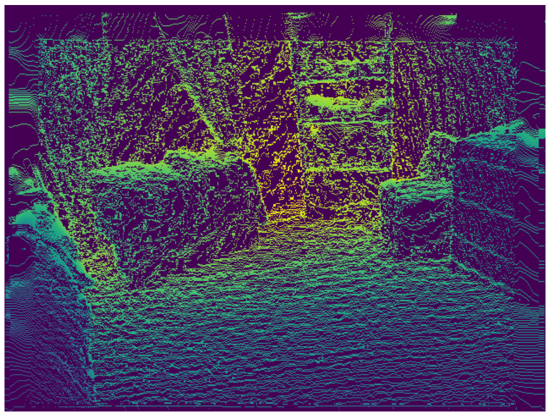
(d) Distribution along y-axis

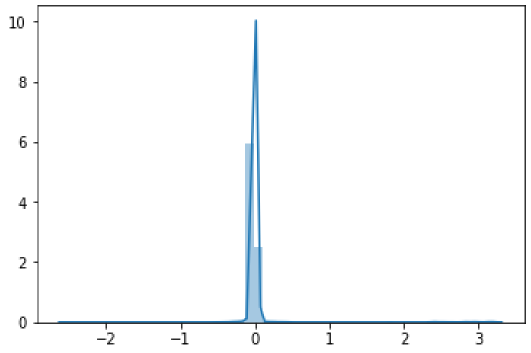
Figure 21: Distribution of Image Gradients of a messy room


(a) Image Gradients along x-axis



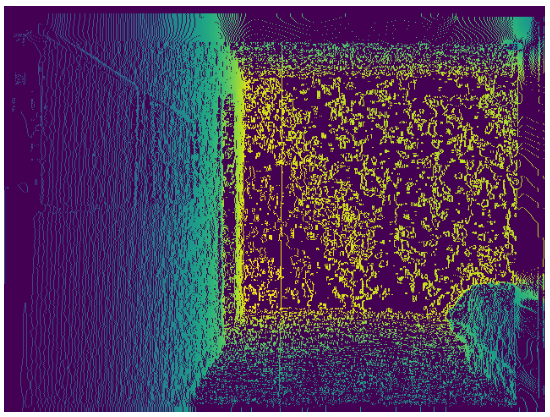
(b) Distribution along x-axis

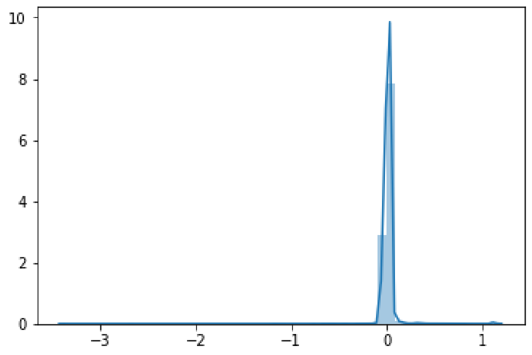

(c) Image Gradients along y-axis



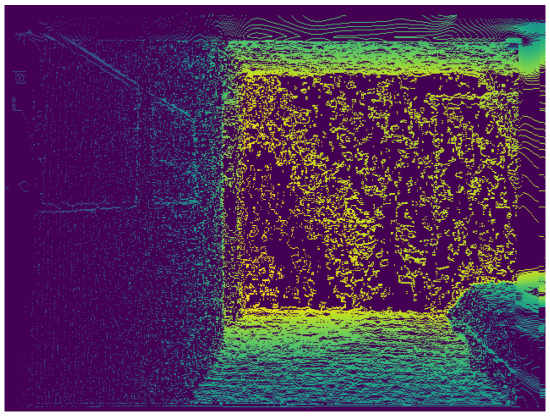
(d) Distribution along y-axis

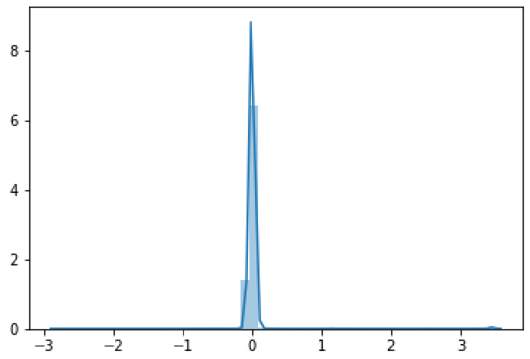
Figure 22: Distribution of Image Gradients of a clean room


(a) Image Gradients along x-axis



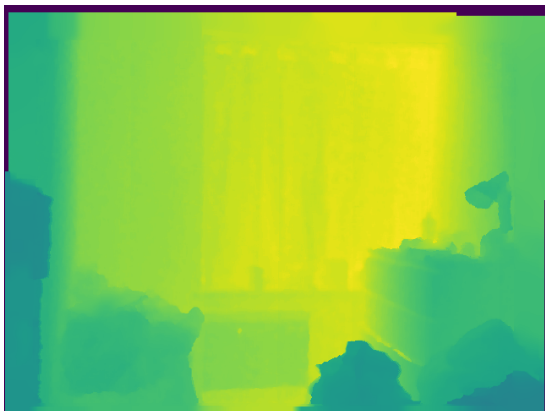
(b) Distribution along x-axis

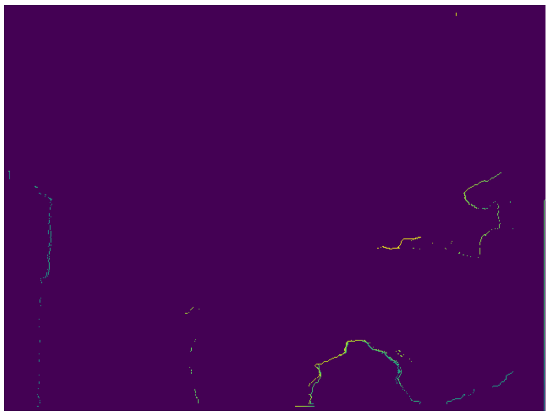

(c) Image Gradients along y-axis

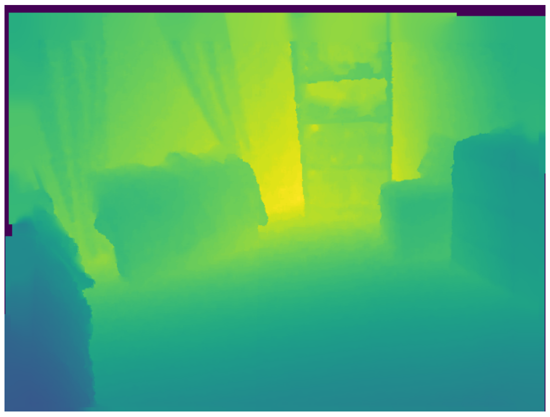


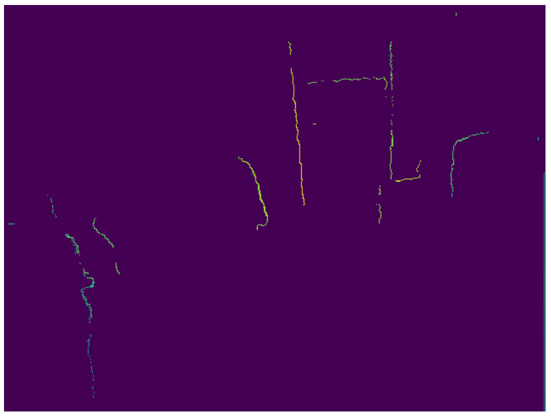
(d) Distribution along y-axis

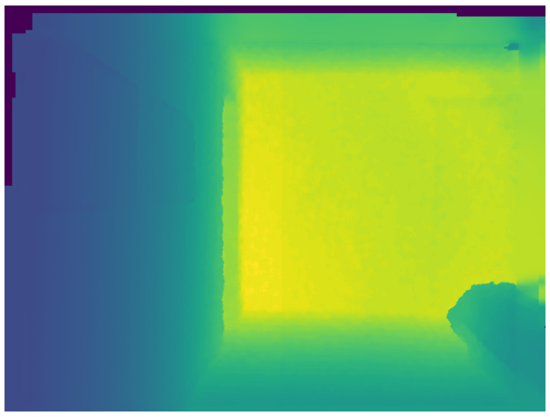
Figure 23: Distribution of Image Gradients of a room with paintings


(a) Depths of messy room


(b) Pixels considered for Edge


(c) Depths of clean room


(d) Pixels considered for Edge


(e) Depths of room with paintings

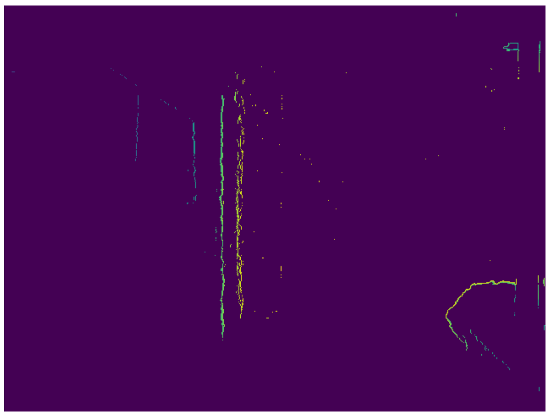

(f) Pixels considered for Edge

Figure 24: Pixels considered for Edge metric from Ground Truth Depths

Chapter 6

EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Network Architecture

To validate the efficacy of our proposals, we use a deep learning approach as the learning algorithm of choice. We use the encoder-decoder Convolutional Neural Network(CNN) architecture popularized by (Laina et al. 2016) that achieved state-of-the-art performance in purely RGB based depth estimation. To benchmark performance, we use the same network as (Ma and Karaman 2018) shown in the below Figure. The network takes in an input with four channels; three channels for RGB and one channel for sampled sparse depth. Since our experiments are conducted specifically on the NYU-D V2 dataset, unlike previous work we use ResNet-50(He et al. 2016) as the feature extracting layers which act as the encoder.

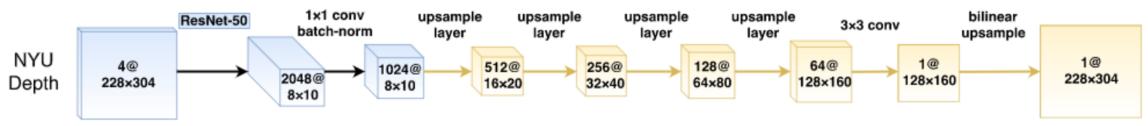


Figure 25: CNN architecture taken from (Ma and Karaman 2018). Encoding layers are in blue, decoding layers are in yellow.

6.2 Hyperparameters

To train our network, we utilize a few key hyperparameters that are tailored for unique use-cases. We compare results across combinations of these hyperparameters in the following sections.

- *num_samples*: Number of sparse depth samples being passed to the network. Possible options-
  - 20
  - 50
  - 100
  - 200
- *sparsifier*: Sparse sampling strategy used. Possible options-
  - *uar*: probabilistic sampling strategy from (Ma and Karaman 2018)
  - $feat$: New Proposed Feature-based sampling strategy
- *criterion*: Loss function used. Possible options-
  - $l1$: L1 loss
  - *active*: *Active Learning* paradigm, adds additional L1 loss for predictions that deviate from interpolated depths
  - *edge*: New proposed loss defined in equation 5.1

6.3 Training Environment

All experiments were run on two NVIDIA Tesla V100 GPUs with 16GB of memory, with all code written in Python using the PyTorch(Paszke et al. 2017) framework.

The weights of the encoding layer were pretrained on the ImageNet(Deng et al. 2009) dataset.

For model training, we use the same fixed hyperparameters as (Ma and Karaman 2018) for benchmarking; a batch size of 16, 20 training epochs, a learning rate of 0.01 that is reduced by 20% every 5 epochs, and a weight decay of 10^{-4} for regularization.

As mentioned in Chapter 2, the dataset used is the NYU-D V2 dataset.

6.4 Results

The metrics used for comparing performance are the ones described in Chapter 2 and the Edge metric proposed in Chapter 5.

- Root Mean Square Error(RMSE): Lower value is better
- Mean Absolute Relative Error(REL): Lower value is better
- Mean Absolute Error(MAE): Lower value is better
- δ_1, δ_2, δ_3: Higher value is better
- Edge: Lower value is better

In the following tables, $\downarrow$ indicates that a lower value is better for the metric, while an $\uparrow$ indicates a higher value is better.

6.4.1 $feat$ versus uar

We first check the performance of our proposed feature-based sparse sampling strategy against the existing state-of-the-art model. As expected, while the proposed sparse sampling strategy does perform better than the existing 2d laser scan based fusion approach(Liao et al. 2017), it doesn't outperform the state-of-the-art performance since sparse maps obtained from this method possess information about the entire RGB image and aren't clustered around detected features as discussed in section 3.2.

num_samples	sparsifier	RMSE(↓)	REL(↓)	δ_1(↑)	δ_2(↑)	δ_3(↑)
20	uar	**0.351**	**0.078**	**0.930**	**0.984**	**0.996**
	feat	0.504	0.135	0.826	0.956	0.989
50	uar	**0.300**	**0.066**	**0.949**	**0.988**	**0.997**
	feat	0.467	0.121	0.854	0.968	0.991
100	uar	**0.273**	**0.056**	**0.961**	**0.991**	**0.998**
	feat	0.437	0.114	0.872	0.971	0.992
200	uar	**0.245**	**0.050**	**0.969**	**0.993**	**0.998**
	feat	0.423	0.110	0.877	0.971	0.992
225	(Liao et al. 2017)	0.442	0.104	0.878	0.964	0.989

Table 1: $feat$(feature-based sparse sampling) versus uar(probabilistic sparse sampling)

6.4.2 $feat$ versus $feat$ with *active learning*

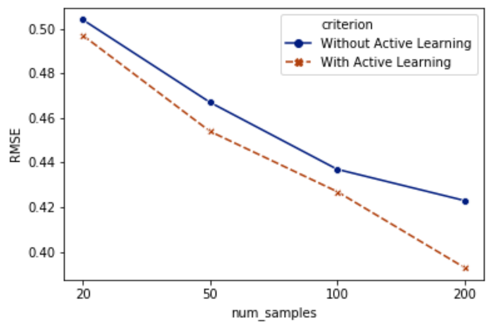
Next, we quantify the performance improvements obtained from using the *active learning* framework described in Section 4.2. We benchmark the frameworks performance against the baseline performance of our proposed feature-based sparse sampling strategy and we see an improvement in all the metrics across all the parameters.

num_samples	criterion	RMSE(↓)	REL(↓)	δ_1(↑)	δ_2(↑)	δ_3(↑)
20	l1	0.504	0.135	0.826	0.956	0.989
	active	**0.497**	**0.132**	**0.832**	**0.961**	0.989
50	l1	0.467	0.121	0.854	0.968	0.991
	active	**0.454**	**0.119**	**0.859**	**0.969**	0.991
100	l1	0.437	0.114	0.872	0.971	0.992
	active	**0.427**	**0.111**	**0.877**	**0.971**	0.992
200	l1	0.423	0.110	0.877	0.971	0.992
	active	**0.393**	**0.101**	**0.893**	**0.977**	**0.993**

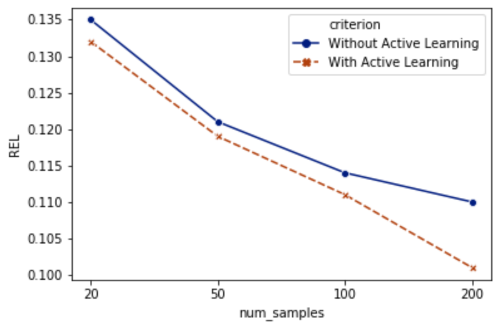
Table 2: Comparing model performance between $l1$(no *active learning*) versus *active*(with *active learning*) with fixed $feat$ sparsifier. *active* outperforms $l1$ in all settings.

6.4.3 Effects of Incorporating Proposed Loss

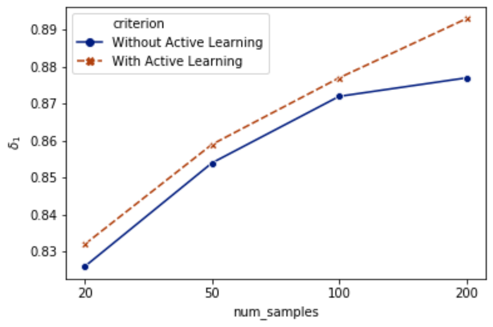
Finally, we check the effects of incorporating our proposed loss function in the current state-of-the-art model(model with $\theta = 1.0$ in the table). We clearly see a noticeable improvement in most of the primary metrics even with minor contribution from loss associated with the edge metric. This improvement becomes more noticeable as the number of samples increases.



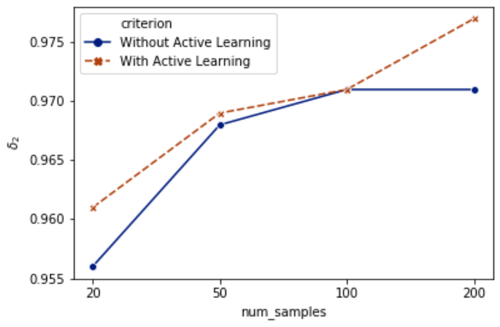
(a) RMSE



(b) REL



(c) δ_1

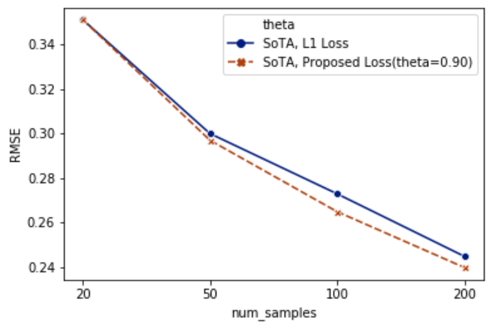


(d) δ_2

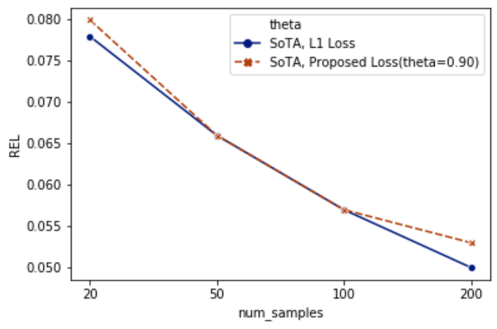
Figure 26: Metric Comparison for model with active learning versus without. Top row: lower is better; bottom row: higher is better.

num_samples	θ	RMSE(↓)	REL(↓)	Edge(↓)	δ_1(↑)	δ_2(↑)	δ_3(↑)
20	1.0	0.351	**0.078**	0.474	**0.930**	**0.985**	0.996
	0.95	0.359	0.080	0.468	0.926	0.982	0.995
	0.90	**0.351**	0.080	0.457	0.927	0.985	0.996
	0.75	0.358	0.087	0.447	0.919	0.984	0.996
	0.50	0.368	0.090	**0.443**	0.915	0.984	0.996
	0.25	0.386	0.100	0.445	0.903	0.981	0.995
	0.0	0.441	0.120	0.469	0.864	0.973	0.993
50	1.0	0.300	0.066	0.431	0.949	0.989	0.997
	0.95	0.294	**0.064**	0.416	0.952	0.989	0.997
	0.90	0.297	0.066	0.420	**0.952**	**0.990**	0.997
	0.75	**0.293**	0.066	**0.398**	0.949	0.989	0.997
	0.50	0.310	0.073	0.403	0.942	0.990	0.997
	0.25	0.326	0.083	0.399	0.930	0.987	0.997
	0.0	0.351	0.091	0.403	0.920	0.985	0.996
100	1.0	0.273	0.057	0.417	0.961	0.992	0.998
	0.95	0.272	**0.055**	0.390	0.960	0.991	0.997
	0.90	**0.265**	0.057	0.390	**0.962**	**0.992**	0.998
	0.75	0.275	0.061	0.385	0.958	0.991	0.998
	0.50	0.265	0.061	**0.357**	0.956	0.991	0.998
	0.25	0.294	0.072	0.372	0.946	0.990	0.997
	0.0	0.317	0.082	0.373	0.937	0.988	0.997
200	1.0	0.245	0.050	0.388	0.970	0.994	0.998
	0.95	**0.232**	**0.046**	0.364	0.971	0.994	0.998
	0.90	0.240	0.053	0.370	**0.971**	0.994	0.999
	0.75	0.245	0.053	0.356	0.968	0.994	0.999
	0.50	0.249	0.057	0.344	0.965	0.994	0.998
	0.25	0.252	0.062	**0.333**	0.960	0.992	0.997
	0.0	0.280	0.068	0.348	0.957	0.992	0.998

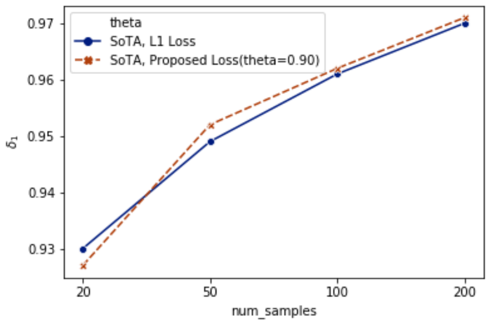
Table 3: Comparing model performance for different values of θ with fixed *edge* criterion and *uar* sparsifier. $\theta = 1.0$ corresponds to *l*1 criterion.



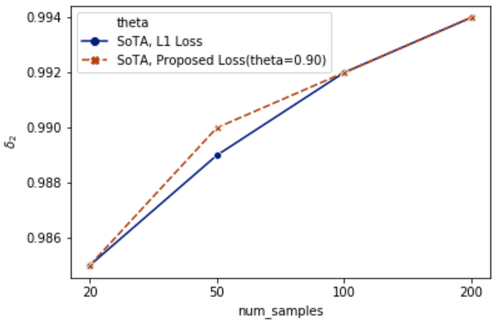
(a) RMSE



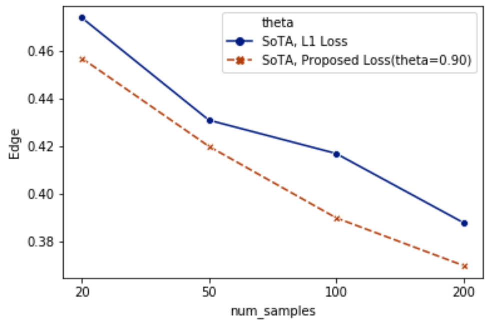
(b) REL



(c) δ_1

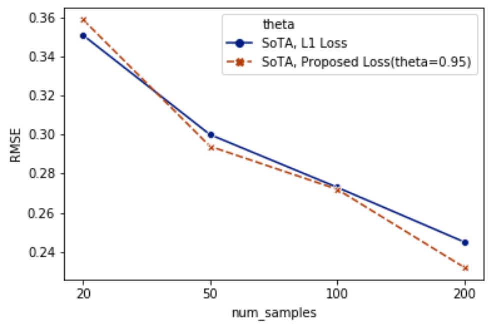


(d) δ_2

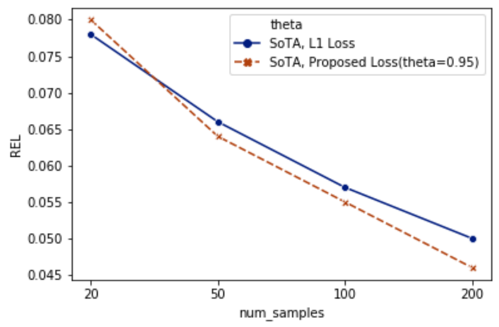


(e) Edge

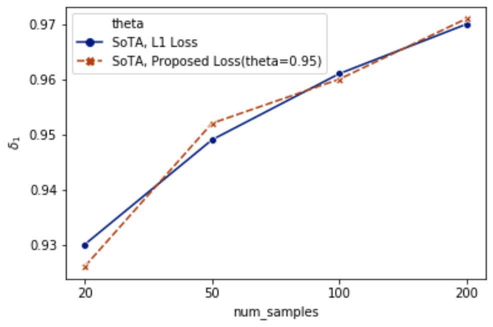
Figure 27: Metric Comparison for SoTA model with $l1$ loss versus SoTA model with proposed loss ($\theta = 0.9$). Top row: lower is better; middle row: higher is better; bottom row: lower is better.



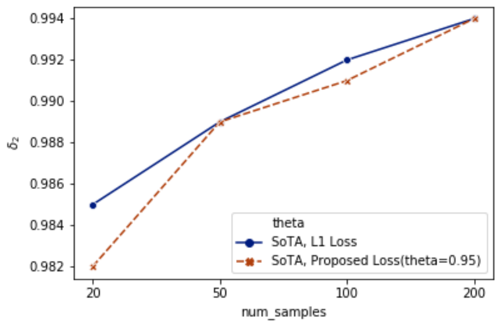
(a) RMSE



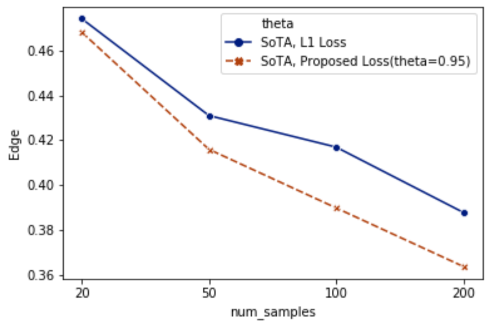
(b) REL



(c) δ_1



(d) δ_2



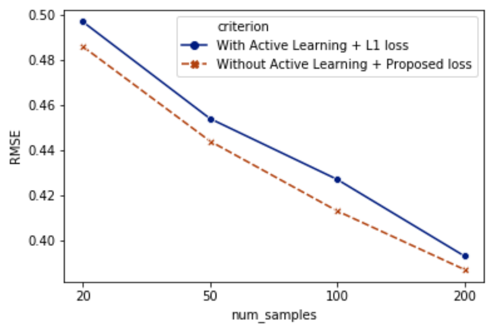
(e) Edge

Figure 28: Metric Comparison for SoTA model with $l1$ loss versus SoTA model with proposed loss ($\theta = 0.95$). Top row: lower is better; middle row: higher is better; bottom row: lower is better.

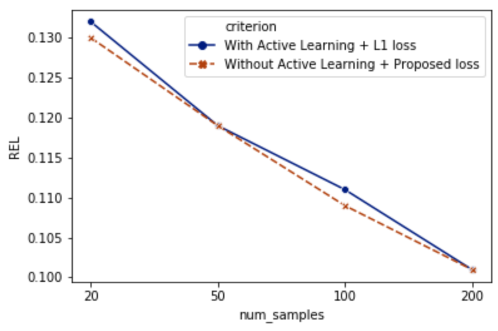
To further validate the efficacy of the proposed loss, we use it in our proposed feature-based sampling strategy and benchmark the results against the results obtained from using the *active learning* paradigm with the $feat$ sparsifier. We use $\theta = 0.9$ for our proposed loss as it showed good results while benchmarking with the *uar* sparsifier. We see that using a $feat$ sparsifier based model with our proposed loss and without active learning performs better than a $feat$ sparsifier based model with $l1$ loss *active learning.*

num_samples	criterion	RMSE(↓)	REL(↓)	Edge(↓)	δ_1(↑)	δ_2(↑)	δ_3(↑)
20	active	0.497	0.132	0.558	0.832	0.961	0.989
	edge	**0.486**	**0.130**	**0.536**	**0.837**	**0.964**	0.990
50	active	0.454	0.119	0.519	0.859	0.969	0.991
	edge	**0.444**	0.119	**0.492**	**0.863**	0.969	0.991
100	active	0.427	0.111	0.488	0.877	0.971	0.992
	edge	**0.413**	**0.109**	**0.469**	**0.884**	**0.973**	0.993
200	active	0.393	0.101	0.460	0.893	0.977	0.993
	edge	**0.387**	0.101	**0.444**	**0.896**	0.977	0.994

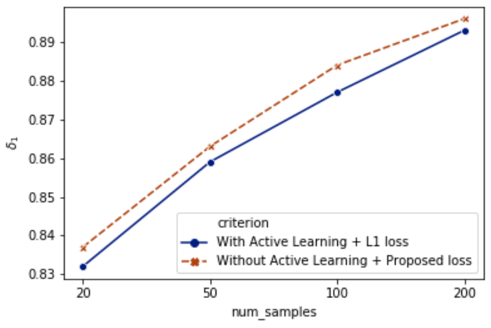
Table 4: Comparing model performance between *active*(model with *active learning*) versus *edge*(with $\theta = 0.9$) with fixed $feat$ sparsifier. *edge* outperforms *active* on all metrics.



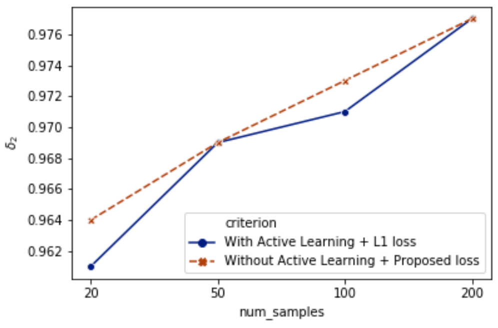
(a) RMSE



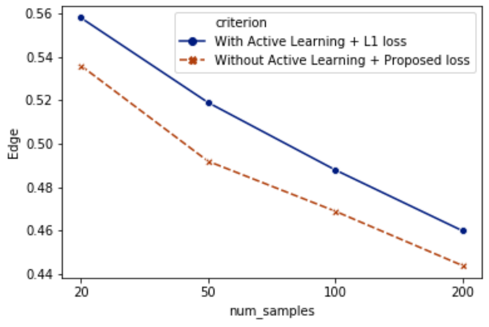
(b) REL



(c) δ_1



(d) δ_2



(e) Edge

Figure 29: Metric Comparison for $feat$ sparsifier based active learning model with $l1$ loss versus model with proposed loss ($\theta = 0.9$) and no active learning. Top row: lower is better; middle row: higher is better; bottom row: lower is better.

6.4.4 Comparing Performance Against Previous Work For Camera-only Systems

Finally, we quantitatively validate our original hypothesis that previous work (Ma and Karaman 2018) does not set realistic baselines for camera-only systems as the sparse patterns used are probabilistic in nature, and are therefore independent of the contents of an image. This is in stark contrast to our method where we use a feature-based sparse sampling strategy.

To verify our hypothesis across a fixed number of sparse samples we take two equivalent models, where one model is trained using the probabilistic sparse sampling strategy proposed by previous work, while the other is trained on sparse samples generated from feature detectors. We then test this model on the same RGB data along with sparse samples generated from the feature-based sampling strategy. This is done to check whether a model trained using the probabilistic sampling strategy can show the same performance improvements on test data that is similar to camera-only systems. It can clearly be seen that this isn't the case, and that our proposed method vastly outperforms previous work and therefore is a more realistic baseline for research work that focuses on utilizing information from only cameras. We compare results on test data that uses a feature-based sparse sampling strategy with and without interpolation.

num_samples	sparsifier	RMSE(↓)	REL(↓)	Edge(↓)	δ_1(↑)	δ_2(↑)
20	uar	0.670	0.196	0.625	0.711	0.932
	feat	**0.487**	**0.131**	**0.536**	**0.837**	**0.962**
50	uar	0.650	0.196	0.626	0.716	0.929
	feat	**0.452**	**0.123**	**0.501**	**0.857**	**0.967**
100	uar	0.645	0.189	0.620	0.697	0.917
	feat	**0.432**	**0.114**	**0.490**	**0.872**	**0.972**
200	uar	0.839	0.219	0.710	0.590	0.799
	feat	**0.402**	**0.106**	**0.463**	**0.888**	**0.977**

Table 5: Comparing model performance between uar(probabilistic sparse sampling) versus $feat$(feature-based sparse sampling) on test data with feature-based sparse sampling **without** interpolation. $feat$ outperforms uar on all metrics.

num_samples	criterion	RMSE(↓)	REL(↓)	Edge(↓)	δ_1(↑)	δ_2(↑)
20	uar	1.567	0.439	1.155	0.443	0.717
	edge	**0.486**	**0.130**	**0.532**	**0.839**	**0.962**
50	active	1.469	0.450	1.263	0.391	0.674
	edge	**0.447**	**0.120**	**0.497**	**0.862**	**0.969**
100	active	1.066	0.327	1.015	0.437	0.783
	edge	**0.419**	**0.110**	**0.476**	**0.880**	**0.973**
200	active	0.949	0.276	00.866	0.463	0.793
	edge	**0.383**	**0.100**	**0.447**	**0.898**	**0.978**

Table 6: Comparing model performance between uar(probabilistic sparse sampling) versus $feat$(feature-based sparse sampling) on test data with feature-based sparse sampling **with** interpolation. $feat$ vastly outperforms uar on all metrics.

We also validate our results in real world scenarios. In the following video we compare our model versus previous work on the monocular video feed from a single camera of a stereo camera pair. We use the RGB image from the single camera of a stereo camera along with the sparse depths generated from feature-based triangulation from both the cameras of the stereo camera(Hartley and Zisserman 2003). The visual results further prove our hypothesis as we can clearly see that depth predictions of our model clearly distinguish individual objects in a scene while the predictions of the previous work are haphazard.

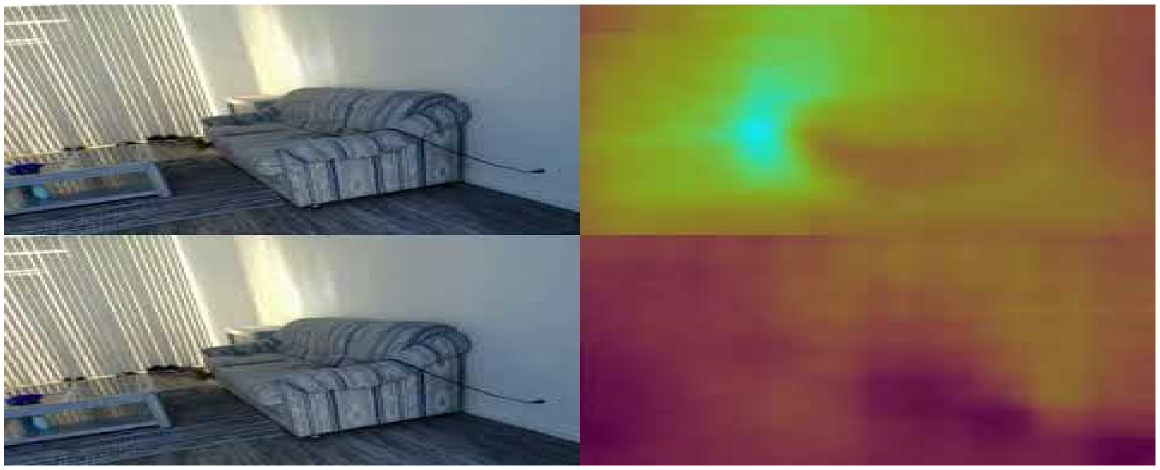


Figure 30: Video Results of Monocular Camera Feed Comparing Our Work With Previous Work. Link: https://youtu.be/-b0fYdtLe9M

Chapter 7

CONCLUSION

7.1 Summary

Research in monocular depth estimation, and computer vision in general, has been moving along at a breakneck pace. This rapid speed of innovation makes it imperative that correct baselines and metrics are established right away so that future work can build on top of realistic baselines to develop specialized algorithms without spending a lot of time first hypothesizing and producing results for a reasonable baseline. In this thesis we take steps towards establishing that baseline for a camera-only system with no additional sparse depth sensors.

- In Chapter 3 we first set up the groundwork for what a realistic camera-only system's baseline could look like and establish the differences between such a system and one that utilizes an additional sparse sensor. We quantitatively show why our work is a more realistic baseline in Chapter 6.
- In Chapter 4 we propose a framework that exploits the clustered nature of the feature-based sparse samples generated from a camera-only system to estimate additional depth information. We also show that these estimated depths can be used a constraint during model training to place additional focus on outliers.
- In Chapter 5 we propose a new metric to quantify performance on hard-to-classify regions. We also show in Chapter 6 that utilizing this metric as an additional component in existing loss functions leads to improvement in both the metric itself, and in most of the other primary metrics.

7.2 Future Work

Since monocular depth estimation is such an open ended research problem, there are multiple avenues that can be explored by building on top of the work presented in this thesis.

- Explore the effects of using feature selectors other than ORB for feature-based sparse depth sampling.
- Experiment with the active learning framework to adaptively weight effects of training on samples depending on the nature of the outlier.
- Explore the effects of refining the threshold values of the Edge metric by utilizing it like the δ_1 parameters instead of using it like the Root Mean Square metric as a training loss.

REFERENCES

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. 2017. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation.” *IEEE transactions on pattern analysis and machine intelligence* 39 (12): 2481–2495.

Chen, Zhao, Vijay Badrinarayanan, Gilad Drozdov, and Andrew Rabinovich. 2018. “Estimating Depth from RGB and Sparse Sensing.” In *The European Conference on Computer Vision (ECCV).* September.

Cohn, David, Les Atlas, and Richard Ladner. 1994. “Improving generalization with active learning.” *Machine learning* 15 (2): 201–221.

Delaunay, Boris, et al. 1934. “Sur la sphere vide.” *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7 (793-800): 1–2.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. “Imagenet: A large-scale hierarchical image database.” In *2009 IEEE conference on computer vision and pattern recognition,* 248–255. Ieee.

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

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

Hartley, Richard, and Andrew Zisserman. 2003. *Multiple view geometry in computer vision.* Cambridge university press.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. “Deep residual learning for image recognition.” In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 770–778.

Karsch, Kevin, Ce Liu, and Sing Bing Kang. 2014. “Depth transfer: Depth extraction from video using non-parametric sampling.” *IEEE transactions on pattern analysis and machine intelligence* 36 (11): 2144–2158.

Konrad, Janusz, Meng Wang, and Prakash Ishwar. 2012. “2d-to-3d image conversion by learning depth from examples.” In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops,* 16–22. IEEE.

Laina, Iro, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. 2016. "Deeper depth prediction with fully convolutional residual networks." In *2016 Fourth international conference on 3D vision (3DV),* 239–248. IEEE.

Liao, Yiyi, Lichao Huang, Yue Wang, Sarath Kodagoda, Yinan Yu, and Yong Liu. 2017. "Parse geometry from a line: Monocular depth estimation with partial laser observation." In *2017 IEEE International Conference on Robotics and Automation (ICRA),* 5059–5066. IEEE.

Liu, Fayao, Chunhua Shen, and Guosheng Lin. 2015. "Deep convolutional neural fields for depth estimation from a single image." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 5162–5170.

Liu, Miaomiao, Mathieu Salzmann, and Xuming He. 2014. "Discrete-continuous depth estimation from a single image." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 716–723.

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

Lowe, David G, et al. 1999. "Object recognition from local scale-invariant features." In *iccv,* 99:1150–1157. 2.

Ma, Fangchang, and Sertac Karaman. 2018. "Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image."

Nathan Silberman, Pushmeet Kohli, Derek Hoiem, and Rob Fergus. 2012. "Indoor Segmentation and Support Inference from RGBD Images." In *ECCV.*

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. "Automatic differentiation in pytorch."

Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. 2011. "ORB: An efficient alternative to SIFT or SURF." In *ICCV,* 11:2. 1. Citeseer.

Saxena, Ashutosh, Sung H Chung, and Andrew Y Ng. 2006. "Learning depth from single monocular images." In *Advances in neural information processing systems,* 1161–1168.

Willmott, Cort J, and Kenji Matsuura. 2005. “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance.” *Climate research* 30 (1): 79–82.