Multiview Monocular Depth Estimation Using Unsupervised Learning Methods

by

Kevin S. Chan

S.B., E.E. and C.S. M.I.T. (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

© Massachusetts Institute of Technology 2018. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole and in part in any medium now known or hereafter created.

Author .....
Department of Electrical Engineering and Computer Science
May 21, 2018

Certified by .....
Jonathan P. How
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Multiview Monocular Depth Estimation Using Unsupervised Learning Methods

by

Kevin S. Chan

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Existing learned methods for monocular depth estimation use only a single view of scene for depth evaluation, so they inherently overfit to their training scenes and cannot generalize well to new datasets. This thesis presents a neural network for multiview monocular depth estimation. Teaching a network to estimate depth via structure from motion allows it to generalize better to new environments with unfamiliar objects. This thesis extends recent work in unsupervised methods for single-view monocular depth estimation and uses the reconstruction losses for training posed in those works. Models and baseline models were evaluated on a variety of datasets and results indicate that indicate multiview models generalize across datasets better than previous work. This work is unique in that it emphasizes cross domain performance and ability to generalize more so than performance on the training set.

Thesis Supervisor: Jonathan P. How
Title: Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank Professor Jonathan How for his support and guidance through this thesis. Thank you to my parents and family for their unwavering support and to my racecar friends for keeping me busy outside of lab. Dong-ki, remember to eat lunch.

Contents

1 Introduction **10**

2 Related Work **13**

2.1 Learned Structure from Motion 13

2.2 Unsupervised Depth Estimation 13

3 Datasets **15**

3.1 KITTI 15

3.2 FlyingThings3D Dataset 15

3.3 NCLT Dataset 16

3.4 Rover Dataset 16

4 Key Ideas **18**

5 Methods **20**

5.1 Naive Architecture 21

5.2 Simultaneous Disparity and Odometry 22

5.3 Multiview Architecture 23

6 Results **26**

6.1 Baselines 26

6.2 Naive 28

6.3 Multiview 28

7 Discussion **38**

8 Conclusion **39**

A Background Equations **41**

B Metrics **42**

B.1 Relative Losses 42

B.2 Scale Invariant Loss 42

C Network Parameters **44**

C.1 Naive Network Parameters 44

C.2 Multiview Network Parameters 46

D Table of Results **48**

List of Figures

1-1 Absolute error in range estimation versus range for varying angles relative to the camera centerline. Directly in front of the sensor is 90 degrees. Figures taken from [15] 11

3-1 Sample image from the FlyingThings3D dataset. 17
3-2 Jackal ground robot equipped with Velodyne LIDAR unit 17

5-1 Naive two view network architecture 21
5-2 Schematic representation of a depth and odometry prediction network. The final layer of encoder activations goes to a fully connected network that outputs an odometry prediction. Encoder activations also go to the decoder to form a disparity prediction. 22
5-3 Schematic representation of multiview network architecture. Both input images are sent through encoders with shared weights. Final layer activations and skip connections for both encoders are subtracted from one another. 23

6-1 Histogram of weights in first layer of naive network. 31
6-2 Two examples showing shortcomings of multiview compared to monodepth. Multiview is the top row, monodepth is second row, first input image is shown on bottom. In (a), part of the white van is missing. In (b), the traffic light is missing because the camera is stationary. . . . 32
6-3 The bias of the multiview network which can be seen by feeding the network the same two input images. 33

6-4 Models trained on FlyingThings3D evaluated on KITTI. Multiview is the top row, monodepth is second row, first input image is shown on bottom. In (a), monodepth views the entire road as a close up object. Multiview clearly sees the objects in the scene that are moving like the speed limit sign and on coming car. In (b), again, monodepth views almost the whole right side of the input image as a single object. Multiview, however, is able to distinguish the two cars passing through the intersection from the one just coming in to view on the right. . . 34

6-5 Multiview vs monodepth on the rover dataset. Multiview is the center image, monodepth is the bottom image. The first input image is the shown on top. Multiview detects the box in the foreground, but loses the structure of the scene in the background. Monodepth performs better in this low texture environment. 36

6-6 Multiview and monodepth results on the NCLT dataset, red rectangles in monodepth frames added for clarity. In (a), (c), and (d), monodepth sees the legs of the pedestrian but not their upper body. Multiview does a better job at this. In (b), monodepth hallucinates some object on the bottom left corner of the image. In (e), monodepth misses the car in front of the robot completely, whereas multiview clearly sees it. 37

List of Tables

6.1 Performance of baseline methods: Error metrics for monodepth [7] trained on KITTI and FlyingThings3D. Scale invariant loss for a constant uniform prediction on all the evaluation sets. A high scale invariant loss indicates that the furthest true distances are many times the closest and hence a uniform prediction has high scale invariant loss. 27

6.2 Quantiative comparison of monodepth and naive architectures on evaluation metrics. Despite having more input information than monodepth, the naive method does not significantly outperform monodepth on either the KITTI dataset (on which both methods were trained) or the cross domain evaluation tests. 29

6.3 Performance of monodepth and multiview networks trained on both KITTI and FlyingThings3D. The best score for each error metric for each dataset is bolded. 30

C.1 Naive network architecture. + represents concatenation. *: Disparity layers are also concatenated to some activations in the decoder, but are upscaled by a factor of two. 45

C.2 Multiview network architecture. The network has two parallel encoders that have inputs left_0 and left_1 respectively. The decoder has skip connections which are the element-wise difference of the specified layers in the encoders. For example, conv6b_1 - conv6b_0. + represents concatenation. *: Disparity layers are also concatenated to some activations in the decoder, but are upscaled by a factor of two. 47

D.1 Table of results for all experiments 49

Chapter 1

Introduction

Depth estimation from images is an important tool in a variety of applications, especially in robotics. While several dedicated ranging sensors, such as LIDAR and structured light sensors, provide superior depth accuracy compared to visual methods, they are not suited for all applications; LIDAR units are relatively big and expensive, while structured light sensors have poor detection range. Cameras remain a very cost effective and compact sensor choice for small robotic platforms, such as drones or ground robots. Traditionally, binocular or trinocular camera arrangements are used and depth is calculated as the inverse of image disparity. On space limited platforms, however, the baseline of the stereo cameras may be too small to provide meaningful disparity measurements beyond a few meters. Figure 1-1 shows the absolute error in range estimate versus range for various angles to the camera centerline for three different sensors: the StereoLabs ZED stereo camera, the Qualcomm SnapDragon StereoVision sensor, and the Intel Realsense structured light sensor [15]. For objects only 20 degrees off the camera axis ($\theta = 70$ in Figure 1-1), the error in the range estimate grows rapidly beyond a range of 5 to 6 meters to over 10%. This result indicates the ZED and SnapDragon systems cannot effectively compute the disparity between the stereo image pairs to infer depth beyond approximately 5 meters especially off the camera's axis. Beyond this distance, the depth perception problem breaks down to essentially a monocular problem as the disparities between the left and right cameras approaches zero. Using monocular cues could significantly extend

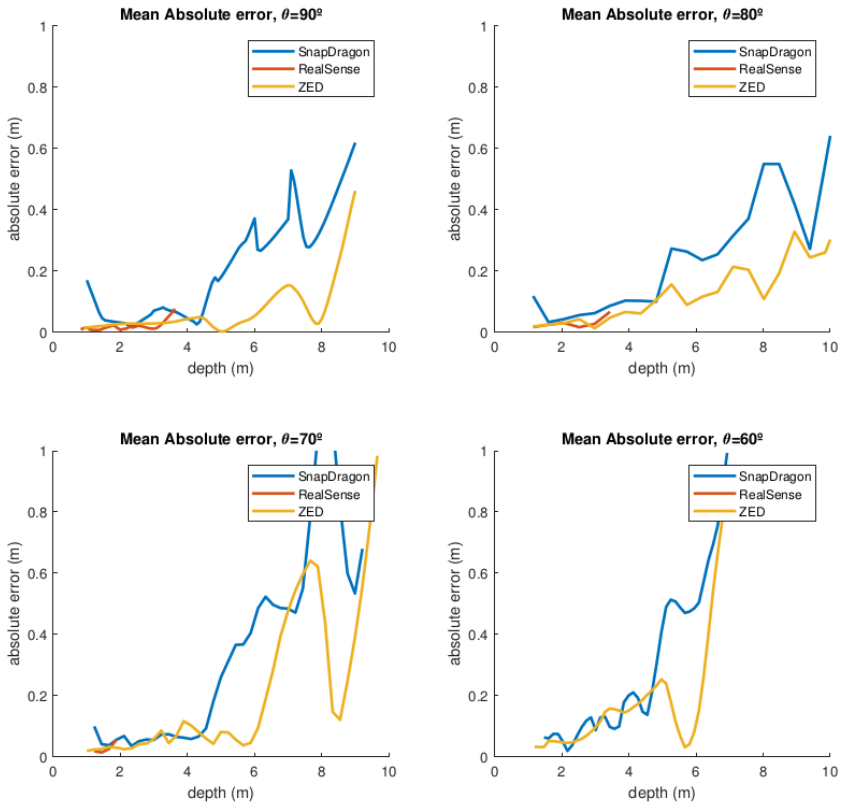


Figure 1-1: Absolute error in range estimation versus range for varying angles relative to the camera centerline. Directly in front of the sensor is 90 degrees. Figures taken from [15]

the effective range of these systems.

Monocular depth estimation is an active area of study in machine learning [7] [16] [18]. While initial supervised learning models for this task have enjoyed success, they require pixel-wise labeled ground truth for training which is very difficult to obtain. Recently, unsupervised methods have used novel reconstruction loss terms to avoid the need for ground truth depth, but utilize only one view of the scene during evaluation [7] [18]. These single-view methods are unable to resolve the inherent scale ambiguity problem in monocular depth perception. They primarily learn implicit sizes for different objects in the training scenes to predict depth, essentially overfitting to the training environment. Thus, these methods will generalize poorly to new environments with previously unseen objects.

One potential way to generalize monocular depth estimation methods is to utilize

multiple views of the scene. Structure from motion can resolve the scale ambiguity in the monocular depth estimation task to within a scale factor of camera displacement [14]. Camera displacement is also relatively easy to measure on some robotic platforms like ground rovers by using wheel encoders or IMUs, allowing these algorithms to recover absolute scale. Deep learning methods for monocular depth estimation have been shown to generalize better to new scenes by learning structure from motion rather than implicit object sizes [16], but are data intensive to train.

The work in this thesis focuses on extending monocular depth estimation techniques to incorporate multiple views of the scene and leverage motion cues in a unsupervised manner. This will enable the estimator to generalize to new environments well without a significant increase in the required inputs for training. This work finds that multiview networks achieve comparable performance to single view networks and generalize to certain test datasets better than single view networks.

The main contributions of this thesis can be summarized as follows:

- Prototyping different network architectures to solve the multiview monocular depth estimation problem by approximating structure from motion.
- Collecting an indoor dataset for testing depth perception networks.
- Testing and analyzing the ability for monocular depth estimation networks to generalize to environments different than the ones they were trained on.

Chapter 2

Related Work

2.1 Learned Structure from Motion

This work is similar in spirit to DeMoN [16] which also attempts to leverage motion cues to improve depth estimation. As a byproduct of depth estimation, DeMoN outputs optical flow which forces the network to use both inputted images. [16] trains with supervised loss terms on depth, optical flow, and camera motion. This data (especially dense optical flow) is difficult to collect when building a real-world dataset, meaning DeMoN is primarily trained on synthetic data.

DeMoN does reinforce the idea that motion parallax is a powerful tool for generalizing depth estimation to new scenes as it achieves good results in previously unseen environments.

2.2 Unsupervised Depth Estimation

New unsupervised monocular depth estimation techniques outperform previous supervised methods. These unsupervised methods forgo the need for labeled pixelwise ground truth depth which is not provided in popular datasets like KITTI [6]. Range ground truth is usually sparse and does not cover the whole image space, so is unsuitable for training dense methods.

[7] predicts stereo disparity with a fully convolutional model trained with a stereo

reconstruction loss term. During training, the network's predicted disparity is used to transform one image to the other image in a stereo pair. The transformed image is compared against the corresponding training example with a photometric loss term. Other optimizations are presented including disparity smoothness loss and consistency beteween disparities predicted on the left and right images. At test time, the method presented in [7] evaluates depth on a single frame from a single camera and outputs its estimate for the disparity in an imaginary stereo camera pair.

While [7] uses stereo image pairs in training, [18] uses monocular video. [18] uses a pose prediction network to warp an image given its predicted depth into the views from the neighboring temporal frames. These reconstructed views are compared to the training video frames with a photometric loss term. The depth and pose estimation networks are trained in an unsupervised manner from this loss. Like in [7], however, the depth network from this work only considers a single view.

Chapter 3

Datasets

Training the networks in this thesis requires calibrated stereo video only. Video is required because the multiview network requires two consecutive frames from each camera. The networks are also evaluated on datasets that provide calibrated monocular video and aligned ground truth depths or disparities.

3.1 KITTI

KITTI [6] is used for training and evaluation. 40,000 of the available color stereo frames in KITTI compose the training set. KITTI also has a set of 200 dense pixelwise disparity maps for 200 stereo examples that are used for evaluation.

3.2 FlyingThings3D Dataset

The FlyingThings3D dataset from [12] is used for training and evaluation. It is a synthetic dataset of flying objects with a varied background. 20,000 stereo examples from this dataset are used for training. The dataset also includes dense disparity maps which are again use for evaluation. Figure 3-1 shows a sample image from the dataset.

3.3 NCLT Dataset

The 2012-02-12 run of The North Campus Long-Term Vision and LIDAR Dataset (NCLT) from [2] is used for evaluation because it features snow which is very visually different from the training data. Methods are evaluated on the forward facing camera (Cam5) of the setup. This dataset includes ground truth depth scans from a Velodyne 32 beam LIDAR which are reprojected into the camera frame for evaluation.

3.4 Rover Dataset

Another evaluation dataset was collected using a modified Jackal ground robot. The Aerospace Controls Lab uses the Jackal robot for navigation tasks in pedestrian rich environments. A monocular depth perception network trained specifically on the indoor scenes the Jackal typically navigates means the Velodyne LIDAR unit on the robot could be replaced with a single camera.

The rover dataset includes images from the front facing camera on the robot and scans from a Velodyne VLP-16 reprojected into the camera frame. The camera to Velodyne transforms were calibrated by hand and data was collected in ROS [13].

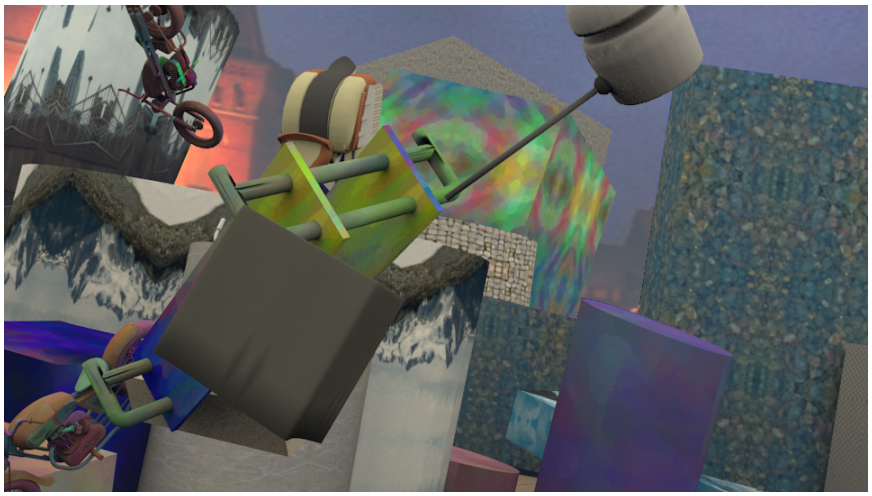


Figure 3-1: Sample image from the FlyingThings3D dataset.

Figure 3-2: Jackal ground robot equipped with Velodyne LIDAR unit

Chapter 4

Key Ideas

The key idea of this thesis is to properly structure a neural network to learn structure from motion while still leveraging the recent unsupervised methods for training depth estimation in [7]. In doing so, we will improve cross domain performance with no increase in training requirements. This work is different from previous work in that it focuses primarily on cross domain performance. It does not find good performance on the training set alone satisfactory, requiring improvements across a range of evaluation sets to show convincing results. Cross domain performance is a good indication of the robustness of the learned algorithms.

Depth estimation from a single view is an inherently ill-posed problem as there are infinitely many environments that can produce the same image from a given view. Adversarial agents in the real world could take advantage of this behavior to confuse a robot's perception algorithms. Additionally, single view methods will have trouble generalizing to new environments with new objects not seen in the training set.

Structure from motion methods are less susceptible to these problems. Assuming a static environment, optical flow can perfectly reconstruct depth. Structure from motion algorithms, however, may not perform as well as single view methods on dynamic scenes where it becomes difficult to determine whether camera motion or subject motion is generating the observed optical flow.

With perfect optical flow between two frames from a single camera, one can perfectly reconstruct the depth of the scene to within a scale factor of camera motion.

This estimation generalizes to all datasets and does not simply over-fit the training set. A similarly perfect predictor for single view monocular depth does not exist. This work focuses on extending the monodepth network from [7] to include multiple input views for this fact.

Chapter 5

Methods

All methods are trained on stereo data and evaluated on pairs of frames from a single camera. All networks output disparity between the left and right cameras of a stereo pair.

The work in this thesis uses the stereo reconstruction loss proposed in [7]. To extend the reconstruction loss to the multiple view problem, the networks output a disparity map aligned with the most recent frame fed into the network. During training, the disparity map is used to transform the most recent image into its corresponding stereo pair image via a bilinear sampler. Then, as in [7], the reconstructed image is compared to the ground truth stereo image with L1 and SSIM [17] loss terms.

When training the multiple view methods described below, a series of augmentations is performed to the training data to improve performance. This thesis refers to the image at frame t as the "first" image and the image at frame $t-1$ as the "second" image. With small probability, the same training image is used for both the first and second views. If images are not repeated, the order of the input images is flipped with even probability during training. This forces the network to learn how to process images while the camera moves forwards and backwards. Then, a series of color augmentations is applied.

All hidden layers use exponential-linear activations [3] as used in [7]. Experiments with leaky ReLU [9] and ReLU [11] found that these did not consistently converge and would drive all output disparities to zero. The network's decoders provide output

at 4 resolution scales and losses are computed at each. Output layers use sigmoid scaled by a constant factor of 0.3.

Models were implemented in Tensorflow [10]. Training was executed on 4 NVIDIA P40 GPU's for 50 epochs which took about 12 hours to train on the KITTI dataset of approximately 40,000 examples of size 256×512. A batch size of 28 was used with the Adam [8] optimizer with the Tensorflow defaults of $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. Please see the appendices and provided code [1] for more details.

5.1 Naive Architecture

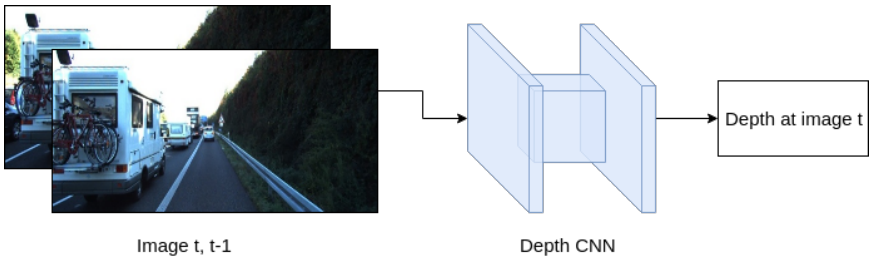


Figure 5-1: Naive two view network architecture

The first approach considered was a naive implementation. Building on the VGG-16 architecture used in [7], the input is changed to be the concatenation of two temporally neighboring frames (a six channel input) rather than just one image (with three channels).

Technically, the information needed to calculate depth from optical flow is given to the network in this architecture; the network receives two views of the scene. It is possible for the weights of the network to learn some correlation function between the two images which would allow the network to determine optical flow and thus depth. The learning objective in this approach, however, is almost certainly not clear enough for effective learning. With no constraints to force the network to consider optical flow, this network was expected to essentially ignore the image at time $t - 1$ and revert to the single view result. This network architecture is shown schematically in Figure 5-1. Network parameters are shown in Appendix C.1.

[1] Available at https://github.com/kschan/multiview

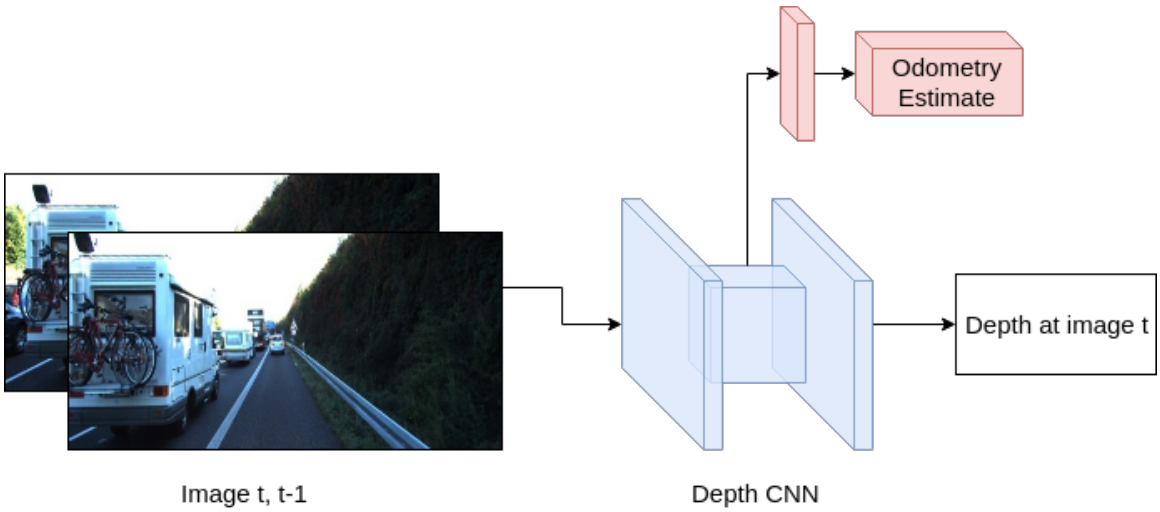


Figure 5-2: Schematic representation of a depth and odometry prediction network. The final layer of encoder activations goes to a fully connected network that outputs an odometry prediction. Encoder activations also go to the decoder to form a disparity prediction.

5.2 Simultaneous Disparity and Odometry

Because depth from structure from motion is resolved to within a scale factor of camera motion, it is useful to predict camera odometry. Predicting odometry could also be an effective learning objective to force the network to use both input images as odometry cannot be predicted from a single image alone. Supervised odometry prediction is possible because ground truth odometry is easy to collect and included in popular datasets like KITTI.

The architecture in Figure 5-2 was investigated to perform simultaneous disparity and odometry prediction. It is based on the same VGG-16 architecture as the naive architecture from Section 5.1, but adds a small fully connected network at the final activations of the encoder. This fully connected network would output an odometry prediction. During training, a supervised odometry prediction loss term would back propagate though the fully connected network and the encoder. The idea was that the encoder would learn a more consistent keypoint descriptor across its channels to form a better odometry prediction.

This approach was largely inpsired by [18] which uses a similar network architecture to predict camera egomotion and an explainability mask as part of their training

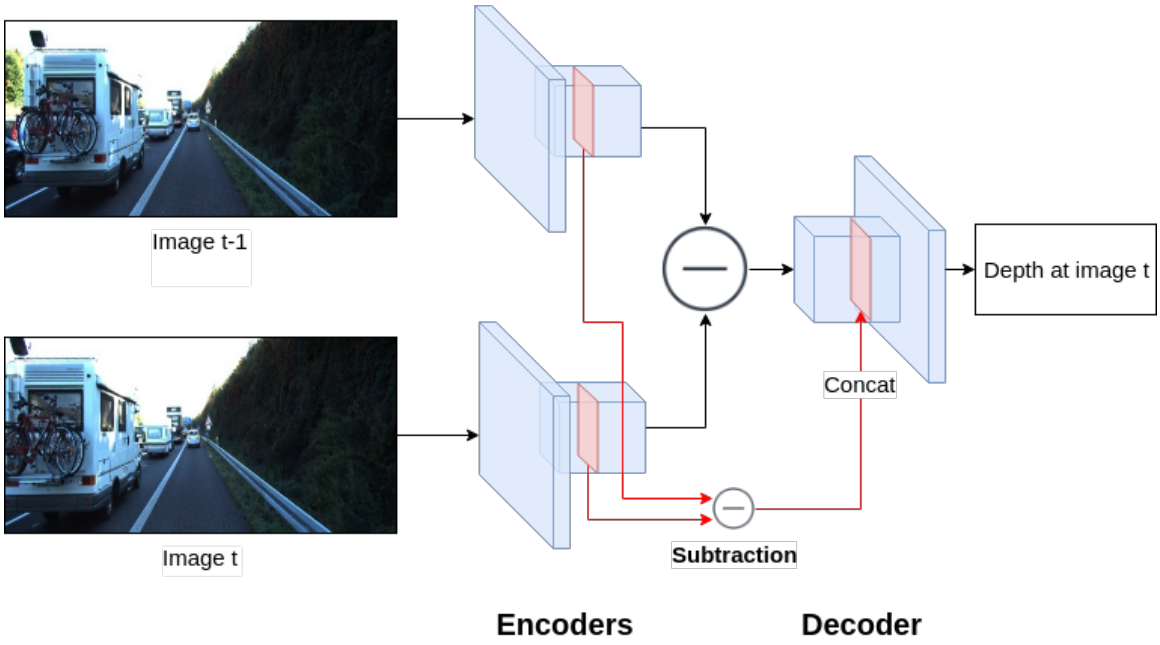


Figure 5-3: Schematic representation of multiview network architecture. Both input images are sent through encoders with shared weights. Final layer activations and skip connections for both encoders are subtracted from one another.

pipeline.

Experiments with this architecture ultimately failed, however, as it was very difficult to make both learning objectives converge simultaneously. Ultimately, larger changes to the network architecture were made to force equal consideration of both input images as will be discussed in the next section.

5.3 Multiview Architecture

The results on the naive network discussed in Section 6.2 confirmed that without a clearer learning objective or major architectural changes, a multiview depth perception network would just ignore one of the input images and default to the single view case. The main innovation presented here is a network architecture that predicts depths based on the difference of features across both input images which reinforces a structure from motion learning objective.

The naive architecture was modified to force both images to be used equally. The new architecture has parallel encoders with shared weights for processing both the

first and second images independently. The final activations in the encoders are then subtracted from each other before being sent into the decoder. Skip connections to the decoder are also replaced with the element-wise difference of each layer in the encoder. Figure 5-3 shows the network architecture schematically. Appendix C.2 shows all network parameters. The purpose of shared encoder weights and subtractions is explained below.

Sharing weights in the decoder has several functions. Forcing the weights in both encoders to be equal by using weight sharing simplifies the learning problem. For example, in the naive architecture, the network needs to learn the appropriate first layer filter weights to detect corners twice: once in the channels corresponding to the first image and again for the second image. The naive network, however, never learns these redundant filters, preferring to ignore the second input image. Sharing encoder weights fixes this problem by forcing the two images to be encoded in the same way. The learned encoder for both images ideally should detect a sort of keypoint that could be tracked by the decoder between the two images.

Subtracting the activations on the final layer and all skip layers combines the parallel encoder activations with behavior that guides the network to learn structure from motion. If the activations in the encoder represent the presence of different keypoints the network has learned to detect, subtracting activations between the two images eliminates keypoints that have not moved between the two images. This architecture enforces the idea that objects that have not moved between two images are very far away and thus have low disparity. This operation allows the network to easily ignore the background in the scene. Whereas single view methods have to use cues such as occlusion to determine what is the background, this multiview network does this automatically.

Activations are combined using an element-wise subtraction instead of another convolution layer because the network may learn to ignore the activations from the second image like in the naive network. Subtraction forces both images to be used equally in the depth prediction task. Subtraction, however, is a very destructive operation and can remove lots of information useful for depth prediction. Objects

moving with the camera will appear to be very far away because of the subtraction operation in multiview. Other operations like division instead of subtraction were tried but encountered numerical issues and were unstable during training.

While the subtraction processes distant background object with little flow automatically, keypoint correlation for foreground objects has to be handled by the decoder. Normally, structure from motion algorithms, visual inertial odometry (VIO), and visual SLAM methods have explicit keypoint matching steps where descriptors are matched between frames. This may be accomplished with RANSAC [5] or other methods. The learning objective for learning these correlations in the decoder weights is not explicit, but the results presented in the next section seem to indicate that similar behavior was learned regardless.

The method of subtracting input features across two images to mimic structure from motion is the major innovation and contribution of this work. As mentioned above, the decoder only learns an approximate correlation function to form its depth estimate. This work accepts this simplification from true structure from motion algorithms that have explicit feature matching steps in order to minimize training inputs and use the loss functions defined in [7].

Appendix A shows how structure from motion algorithms are able to recover depth to a scale factor of camera motion. In training, however, the algorithm predicts absolute depth in the form of stereo disparity and is not normalized against camera movement between two input frames. If the network learns to track the flow of a keypoint between two images, it must learn the average displacement between frames in the training set in order to estimate disparity. In this way, the network is expected to overfit to the scale of the training dataset, but should still be able to determine the overall scene shape accurately for new environments.

Chapter 6

Results

A full table of results can be found in the appendix. Relevant parts of the whole table are repeated here for readability. Error metrics are defined in Appendix B.

6.1 Baselines

The algorithms in this thesis are compared against the current state of the art in monocular depth estimation and a constant estimation function.

The monodepth network presented in [7] was retrained on the KITTI data and evaluated on the KITTI, FlyingThings3D, NCLT, and rover evaluation datasets. A network trained on FlyingThings3D was similarly tested. The results of these experiments are presented in Table 6.1.

Monodepth's strong performance on KITTI is replicated. Performance on cross domain evaluation, however, is predictably weak. Monodepth does not inherently have enough information to make accurate predictions about the shape of objects and scenes it has not seen before. This work hopes to demonstrate that a multi-view network can achieve lower scale invariant loss than monodepth on cross domain evaluation tests.

As another comparison point, the scale invariant error metric for a constant prediction was calculated. Each method should of course perform better than a constant estimation. The scale invariant error metric of a constant estimation indicates the

Table 6.1: Performance of baseline methods: Error metrics for monodepth [7] trained on KITTI and FlyingThings3D. Scale invariant loss for a constant uniform prediction on all the evaluation sets. A high scale invariant loss indicates that the furthest true distances are many times the closest and hence a uniform prediction has high scale invariant loss.

		Lower is better					Higher is better		
Model (train dataset)	**Test dataset**	**abs_rel**	**sq_rel**	**rms**	**log_rms**	**scale_inv**	**a1**	**a2**	**a3**
monodepth (kitti)	kitti	0.0869	1.184	4.998	0.165	0.029	0.918	0.969	0.986
monodepth (kitti)	rover	3.0047	57.1109	15.482	1.321	0.2544	0.039	0.091	0.176
monodepth (kitti)	flying	0.7906	19.8081	18.755	0.983	0.561	0.127	0.255	0.391
monodepth (kitti)	umich	0.8721	16.1107	13.638	0.699	0.3454	0.262	0.504	0.671
monodepth (flying)	kitti	0.584	8.1656	15.413	1.186	0.3719	0.100	0.201	0.306
monodepth (flying)	rover	0.5859	3.2786	5.247	1.036	0.331	0.072	0.183	0.341
monodepth (flying)	flying	0.512	144.0073	27.363	0.453	0.188	0.488	0.768	0.888
monodepth (flying)	umich	0.758	13.2022	17.252	1.557	0.6431	0.069	0.139	0.215
constant	kitti					0.34			
constant	rover					0.2986			
constant	flying					0.231			
constant	umich					0.413			

range of the true depths for each dataset; if the furthest distances are multiple times the smallest distances, a uniform prediction will have a high scale-invariant loss. Table 6.1 has the scale invariant losses for all of the evaluation datasets used.

6.2 Naive

The naive network was trained on the KITTI dataset. It sees a marginal improvement over monodepth in KITTI, but only performs on par with monodepth in the cross domain evaluation despite having more input information. It scores a slightly better scale invariant error on all the datasets. A comparison of the two methods is provided in Table 6.2.

Looking at the histogram of the first layer weights of the naive network somewhat confirms the theory that the network is largely ignoring the second image fed into the network. The two input images are concatenated to create a six channel input. Figure 6-1 shows the histograms of weights for the input channels corresponding to the first and second images separately. If the network were actually matching points from the first image in the second image to detect optical flow and depth, one might expect the first filter weights to have the same distribution as the first filters try to detect common low level features like edges and corners. Clearly, the weights for the second image are much more narrowly centered around 0. This suggests that the network is not using the second image fully in making its disparity prediction. Without more constraints on the learning objective, the network will just default to learning the single view case.

6.3 Multiview

As with monodepth, this model was trained on both KITTI and FlyingThings3D and evaluated across all datasets.

Multiview was first trained on KITTI and compared to the monodepth baseline also trained on KITTI. On the KITTI evaluation set, multiview is outperformed by

Table 6.2: Quantiative comparison of monodepth and naive architectures on evaluation metrics. Despite having more input information than monodepth, the naive method does not significantly outperform monodepth on either the KITTI dataset (on which both methods were trained) or the cross domain evaluation tests.

		Lower is better					Higher is better		
Model (train dataset)	**Test dataset**	**abs_rel**	**sq_rel**	**rms**	**log_rms**	**scale_inv**	**a1**	**a2**	**a3**
monodepth (kitti)	kitti	0.0869	1.184	4.998	0.165	0.029	0.918	0.969	0.986
naive (kitti)	kitti	**0.0834**	**1.1386**	**4.906**	**0.161**	**0.0275**	**0.921**	**0.971**	**0.987**
monodepth (kitti)	flying	0.7906	**19.8081**	**18.755**	0.983	0.561	0.127	0.255	0.391
naive (kitti)	flying	**0.7885**	29.4412	19.177	**0.955**	**0.534**	**0.131**	**0.265**	**0.405**
monodepth (kitti)	rover	**3.0047**	**57.1109**	**15.482**	**1.321**	0.2544	**0.039**	**0.091**	**0.176**
naive (kitti)	rover	3.1297	63.1636	16.094	1.341	**0.2528**	0.036	0.09	**0.176**
monodepth (kitti)	umich	**0.8796**	16.5257	13.63	0.700	0.3413	**0.262**	**0.507**	**0.673**
naive (kitti)	umich	0.8962	**16.312**	**13.52**	**0.692**	**0.3227**	0.252	0.497	**0.673**

Table 6.3: Performance of monodepth and multiview networks trained on both KITTI and FlyingThings3D. The best score for each error metric for each dataset is bolded.

		Lower is better					Higher is better		
Model (train dataset)	**Test dataset**	**abs_rel**	**sq_rel**	**rms**	**log_rms**	**scale_inv**	**a1**	**a2**	**a3**
monodepth (kitti)	kitti	**0.0869**	**1.184**	**4.998**	**0.165**	**0.029**	**0.918**	**0.969**	**0.986**
multiview (kitti)	kitti	0.1203	1.7182	6.048	0.200	0.0406	0.874	0.955	0.981
monodepth (flying)	kitti	0.584	8.1656	15.413	1.186	0.3719	0.100	0.201	0.306
multiview (flying)	kitti	0.4499	6.5321	14.107	0.849	0.3019	0.22	0.412	0.569
monodepth (kitti)	flying	0.7906	19.8081	18.755	0.983	0.561	0.127	0.255	0.391
multiview (kitti)	flying	0.6872	**16.4283**	**18.218**	1.035	0.337	0.108	0.222	0.344
monodepth (flying)	flying	**0.512**	144.0073	27.363	**0.453**	**0.188**	**0.488**	**0.768**	**0.888**
multiview (flying)	flying	0.7197	284.2464	57.375	0.502	0.225	0.483	0.755	0.877
monodepth (kitti)	rover	3.0047	57.1109	15.482	1.321	**0.2544**	0.039	0.091	0.176
multiview (kitti) [skip = 1]	rover	2.9804	60.6089	17.206	1.329	0.3548	0.066	0.147	0.236
multiview (kitti) [skip = 10]	rover	1.7103	22.5902	9.95	0.997	0.4061	0.141	0.285	0.424
monodepth (flying)	rover	0.5859	3.2786	5.247	1.036	0.331	0.072	0.183	0.341
multiview (flying)	rover	**0.5498**	**2.6881**	**4.068**	**0.607**	0.3342	**0.247**	**0.494**	**0.707**
monodepth (kitti)	umich	0.8796	16.5257	13.63	0.700	0.3413	0.262	0.507	0.673
multiview (kitti)	umich	0.7337	11.2404	**12.331**	**0.645**	**0.2986**	**0.27**	**0.521**	**0.700**
monodepth (flying)	umich	0.758	13.2022	17.252	1.557	0.6431	0.069	0.139	0.215
multiview (flying)	umich	**0.5756**	**9.4867**	15.132	0.97	0.4561	0.191	0.362	0.508

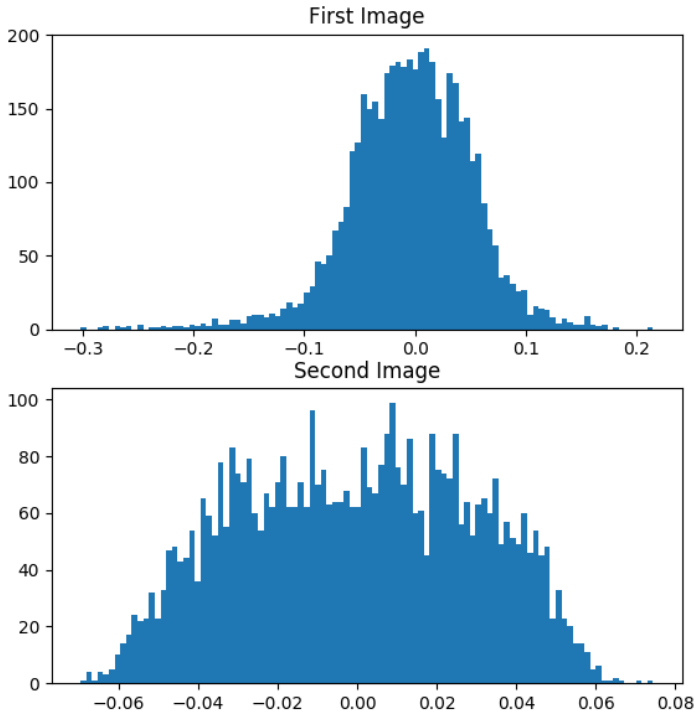
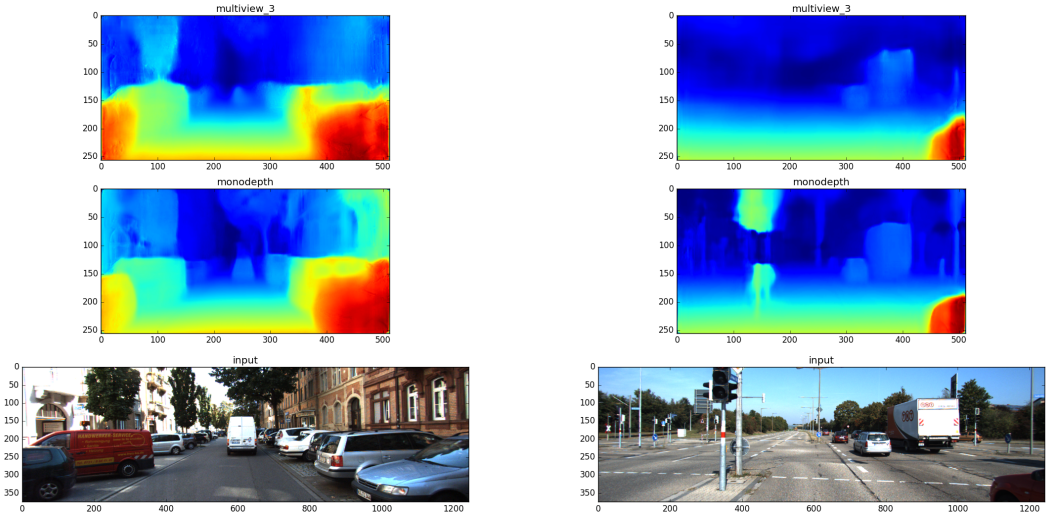


Figure 6-1: Histogram of weights in first layer of naive network.

monodepth. This is somewhat predictable as the subtraction in multiview which allows it to disregard background objects is likely to negatively impact its perception of things in the foreground.

Overall, the disparity maps generated by multiview on KITTI are comparable to those by monodepth even though they have fairly different mechanisms. Figure 6-2 shows two examples that show shortcomings of multiview network compared to monodepth. In (a), the white van in front of the camera is moving at nearly the same speed as the camera and generates no optical flow, so multiview predicts that the top right corner of it is part of the background. In (b), the car is stopped at a red light. There is no flow generated by the stationary traffic signal, so it is assigned a low disparity. The cars continuing straight, however, still appear in the disparity map. For applications with moving robots, systems are primarily interested in depth perception as it relates to tasks like collision avoidance. In these cases, it is more important to accurately detect what things are moving relative to the camera than it



(a) (b)

Figure 6-2: Two examples showing shortcomings of multiview compared to monodepth. Multiview is the top row, monodepth is second row, first input image is shown on bottom. In (a), part of the white van is missing. In (b), the traffic light is missing because the camera is stationary.

is to detect things with no relative motion. This means this behavior, while non-ideal, is not breaking.

Multiview has interesting and nicely explainable behavior when given identical input images. When there is absolutely no change in the input images, the final layer activations and skip connections after subtraction will all be zero vectors. The resulting disparity map is only the result of the bias terms in the decoder layers which represents the best prior for the structure of the scene. The training regime was modified so during training the same image for the first and second input images is given to the network with some small probability to force the bias to represent the whole dataset and not just certain examples with small or zero displacement. These changes produced the bias seen in Figure 6-3. The bias shows the network expects some flat ground below the camera and some expectation of objects on the left and right of the camera. Note that the bias is very symmetric because training images are flipped along the left-right axis with even probability.

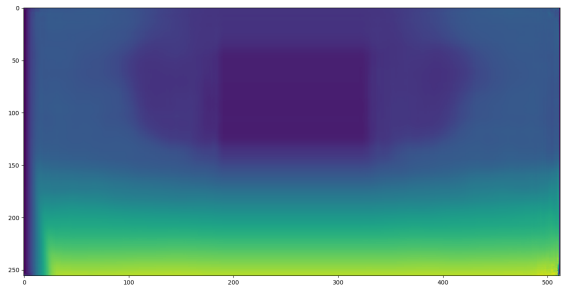
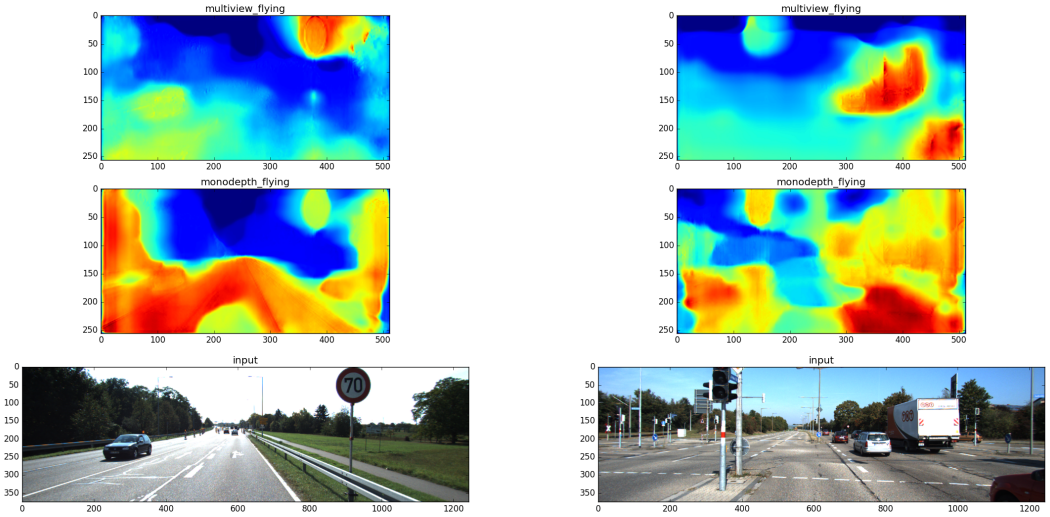


Figure 6-3: The bias of the multiview network which can be seen by feeding the network the same two input images.

Monodepth was expected to outperform multiview on the KITTI evaluation. In general, monodepth is expected to perform better than multiview on the training dataset. To test whether multiview generalized better across datasets, the KITTI trained models were evaluated on the FlyingThings3D dataset. Multiview performs noticably better than monodepth, especially on the scale invariant error which is the best comparison metric for cross domain evaluation. These methods, however, have higher scale invariant error than the constant prediction on the FlyingThings3D dataset. FlyingThings3D has a very high density of scattered and moving subjects, so low scores are expected when evaluating on this.

A model was then trained on FlyingThings3D to evaluate on KITTI. In theory, the highly textured environments in FlyingThings3D should allow the model to learn a relatively robust set of parameters. Some examples of the FlyingThings3D trained models' predicted disparities on KITTI are show in Figure 6-4. The multiple views used in multiview allow it to more accurately distinguish objects from the background. In Figure 6-4a, monodepth views the entire road and a portion of the trees on the side of the road as a single object that is very close to the camera. Multiview does a better job of reconstructing the overall scene structure here and clearly notices the oncoming car and approaching speed limit sign. Figure 6-4b is the same stationary frame from Figure 6-2b. As the cars in the lanes to the right pass the camera, they generate a lot of optical flow and multiview clearly detects them. It is worth noting



(a) (b)

Figure 6-4: Models trained on FlyingThings3D evaluated on KITTI. Multiview is the top row, monodepth is second row, first input image is shown on bottom. In (a), monodepth views the entire road as a close up object. Multiview clearly sees the objects in the scene that are moving like the speed limit sign and on coming car. In (b), again, monodepth views almost the whole right side of the input image as a single object. Multiview, however, is able to distinguish the two cars passing through the intersection from the one just coming in to view on the right.

that although the disparity prediction in 6-4b is generally greater than 6-2b, the overall scene structure is recovered reasonably well. Monodepth seems to struggle with this particular scene as it seems to detect the whole right half of the image as one object very close to the camera.

Trials on the FlyingThings3D dataset showed that multiple views are necessary when objects in the evaluation dataset are very different from those in the training set. A network that just tries to learn the inherent size and shape of objects in the training set will struggle to predict depth in an evaluation set where it cannot find those objects in the new scene.

The KITTI trained methods were evaluated on the rover dataset as a sort of real-world generalization test. Monodepth beat multiview on the scale invariant error metric. One possible explanation considered was that multiview likely learns the av-

erage displacement between camera frames in the training dataset. With two views, depth can only be recovered to within a scale factor of camera motion. So in order to make good predictions on the training dataset, the model must learn the average displacement of the camera to maximize its prediction accuracy. In an attempt to improve multiview's prediction accuracy on the rover dataset, the average displacement between input frames in the rover dataset was forced to be the same as in KITTI by feeding the network images several frames apart.

According to the odometry information in the KITTI dataset, the average forward velocity of the car is approximately 7.8 meters per second. At a capture rate of 10 frames per second, the average displacement between frames in KITTI is about 0.7 meters. Collecting similar measurements on the rover dataset shows that the average displacement between frames is 0.07 meters. So in order to enforce the displacement between frames in the KITTI dataset and the rover dataset to be equal, the network is presented images that are 10 frames apart from each other in the rover dataset. The result of doing this compared to sequential frames is presented in Table 6.3 as multiview with [skip = 1] vs [skip = 10]. Increasing the distance between frames did not improve accuracy.

Looking at the disparity generated by multiview and monodepth on the rover data suggests why multiview is outperformed. The rover dataset features relatively textureless regions on the wall of the lab's drone flight-space and floor. Figure 6-5 shows how multiview does not detect the structure of the wall in front of it. In multiview's disparity there is a large hole in its disparity prediction for the back wall, suggesting that the textureless region was deleted in the center of the image via the subtraction operations in the network. The network reverted to its bias from the KITTI training set in that portion of the wall.

This shows that multiview struggles in low texture regions similar to the wall in Figure 6-5. The subtraction operation removes certain occlusion and shading cues that could be used to predict a large flat wall in front of the camera. Monodepth was able to detect the wall clearly without this subtraction.

Evaluating the KITTI trained models on another real world dataset would deter-

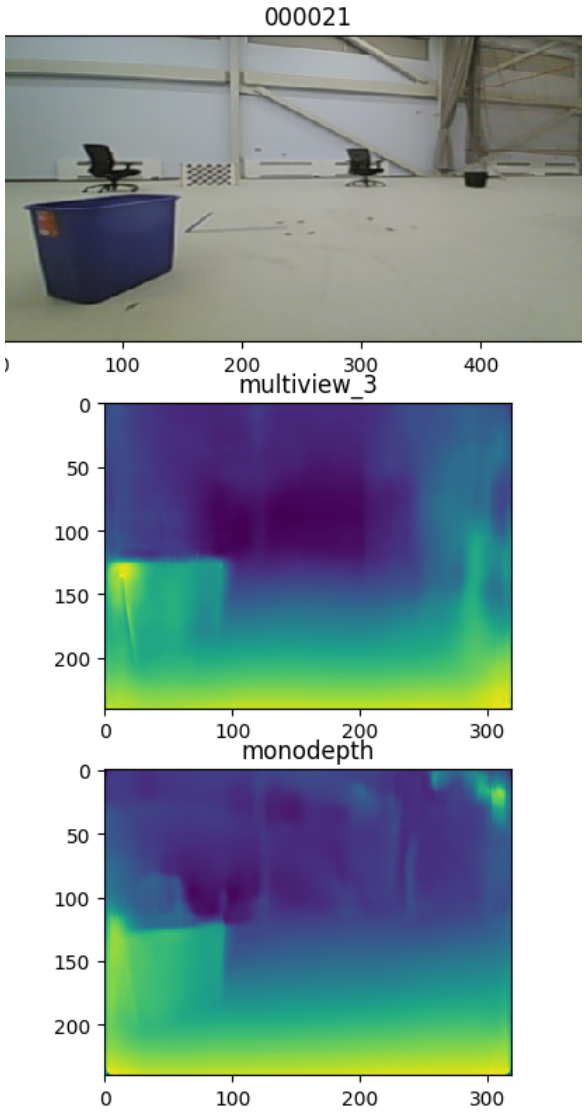


Figure 6-5: Multiview vs monodepth on the rover dataset. Multiview is the center image, monodepth is the bottom image. The first input image is the shown on top. Multiview detects the box in the foreground, but loses the structure of the scene in the background. Monodepth performs better in this low texture environment.

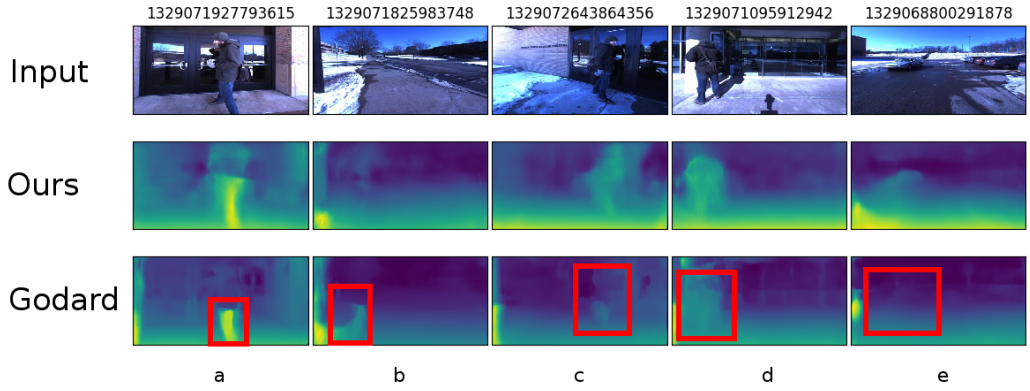


Figure 6-6: Multiview and monodepth results on the NCLT dataset, red rectangles in monodepth frames added for clarity. In (a), (c), and (d), monodepth sees the legs of the pedestrian but not their upper body. Multiview does a better job at this. In (b), monodepth hallucinates some object on the bottom left corner of the image. In (e), monodepth misses the car in front of the robot completely, whereas multiview clearly sees it.

mine if the rover results are representative. It is possible that the structure of the scenes in the rover dataset could have matched monodepth's bias better than multiview's, thus leading to better performance on the scale invariant errors. Multiview was evaluated on the NCLT dataset from the University of Michigan [2]. The 2012-02-12 run of the dataset was chosen because it featured snow which is visually very different from the training set.

The quantitative error measures are provided in Table 6.3. Multiview trained on KITTI achieves the lowest scale invariant error on the dataset and beats monodepth across on the rest of the error metrics. Looking at the disparity maps generated by both networks for some scenes in the evaluation set shows that monodepth struggles with detecting pedestrians very close to the camera and even the cars in many scenes. In KITTI, however, pedestrians never walk as close to the camera as in Figure 6-6a and cars are not seen from the same perspective as in 6-6e. Despite the fact that monodepth had seen these objects in the training set, the new way in which they are presented in the NCLT dataset causes monodepth's predictions to fail.

Chapter 7

Discussion

This work demonstrated an effective way of incorporating multiple input views for monocular depth estimation while still using the unsupervised reconstruction losses posed in [7]. The multiview architecture generalized to new datasets more effectively than monodepth in textured environments. On the FlyingThings3D and NCLT datasets, multiview's ability to effectively filter out background objects from foreground objects allowed it to make accurate depth predictions even for objects it had not seen in the training set. In low texture environments like the one in the rover dataset, however, multiview struggles to create smooth depth predictions along large surfaces and reverts to its training bias in these areas without optical flow. This is largely expected behavior due to the nature of the subtraction operations in the network architecture.

This work also showed that the structure from motion learning objective using the parallel encoders and subtraction operations achieves a good improvement over the naive multiview implementation. There are no mechanisms in the naive architecture to force the network to use both input images, so it tends to just ignore the second input image and defaults to the single view case. By sharing encoder weights and using the subtraction operation instead of more convolutions, the network architecture forces the network to use both images equally in its depth estimate. This architecture also has nice intuitive properties like easily interpretable behavior in the no movement case.

Chapter 8

Conclusion

This thesis had set out to improve the ability for existing unsupervised monocular depth estimation algorithms to generalize to new datasets by incorporating multiple views. Learning structure from motion would allow the network to resolve depth to a scale factor of camera motion, but doing so with the reconstruction losses posed in recent work would alleviate the data requirement required to do so. Using the multiview architecture in combination with the reconstruction loss from [7], this thesis shows the multiview network is able to recover the shape of scenes that are very different visually and structurally from the training scenes more accurately than single view methods.

Multiview is not without its own limitations. For one, multiview relies on optical flow to detect objects, so requires at least two views of the same scene. This means multiview cannot be evaluated on single images or examples without camera motion. As noted in the results, multiview does not perform well in low texture environments where the subtraction operation is very destructive and causes the network to default to the training bias. Multiview also does not have any inputs on odometry which could give it a good prior for optical flow. An odometry input term could improve the learning efficiency even more. Odometry is a relatively simple measurement to capture during data collection and is often included in datasets like KITTI, so it is a fairly cheap training input especially compared to labeled depths.

Further work in building a recurrent multiview network may be promising as

tracking objects across more than two frames may improve accuracy. Currently, the network relies on the decoder of the network to learn to correlate features in the two input images to form the depth predictions, but a better structured network or more defined learning objective may accomplish this more reliably.

Appendix A

Background Equations

The networks in this thesis output disparity from one camera to the other in the training stereo pair. During evaluation, there need not be another actual camera, but given an image, the network will predict the disparity to the stereo pair from the training setup. Disparity is inversely proportional to depth, so depth D is easily calculated from disparity in pixels, d, by:

$$D = \frac{fb}{d}$$

where f is the focal length of the camera in pixels and b is the stereo baseline in meters.

The goal is to create algorithms that create an approximation of a dense flow estimate to produce the output depth. Depth, Z, can be derived from optical flow, V, and camera velocity v_{cam} by:

$$Z = v_{cam}\frac{D}{V}$$

where D is the distance of the point x, y from the focus of expansion of the image [1]. This shows that depth can be recovered up to a scale factor of camera velocity with a perfect pixelwise optical flow prediction.

Appendix B

Metrics

B.1 Relative Losses

These error metrics are used for a prediction Y and truth Y^* of n pixel-wise depth estimates:

- RMSE: $\sqrt{\frac{\sum (Y - Y^*)^2}{n}}$
- RMSE log: $\sqrt{\frac{\sum (\log Y - \log Y^*)^2}{n}}$
- Absolute Relative (abs_rel): $\frac{\sum \frac{Y - Y^*}{Y^*}}{n}$
- Square Relative (sq_rel): $\frac{\sum \frac{(Y - Y^*)^2}{Y^*}}{n}$

These same scores are published by previous methods and allow for a fair comparison.

B.2 Scale Invariant Loss

The scale invariant loss posed in [4] is used to assess each method's ability to reconstruct the overall scene structure. Recovering the absolute scale of a scene is very

difficult, especially when evaluating across datasets with drastically different environments or camera parameters. The relative error terms used will not capture a network's ability to determine the overall shape of a scene if its prediction is a scale factor off from the ground truth.

- Scale Invariant Error (scale_inv): $\frac{1}{n}\sum_i (\log Y_i - \log Y_i^* + \alpha(Y, Y^*))^2$

  where $\alpha(Y, Y^*) = \frac{1}{n}\sum_i (\log Y_i^2 - \log Y_i)$

To have a low scale invariant error, pixels in the prediction must differ by the same factor as in the ground truth. This metric allows for a better comparison of cross domain performance.

Appendix C

Network Parameters

C.1 Naive Network Parameters

Encoder				
layer	**k**	**s**	**chns**	**input**
conv1	7	2	6/32	left_1 + left_0
conv1b	7	1	32/32	conv1
conv2	5	2	32/64	conv1b
conv2b	5	1	64/64	conv2
conv3	3	2	64/128	conv2b
conv3b	3	1	128/128	conv3
conv4	3	2	128/256	conv3b
conv4b	3	1	256/256	conv4
conv5	3	2	256/512	conv4b
conv5b	3	1	512/512	conv5
conv6	3	2	512/513	conv5b
conv6b	3	1	512/514	conv6
conv7	3	2	512/515	conv6b
conv7b	3	1	512/516	conv7

Table C.1 continued from previous page

Decoder				
layer	**k**	**s**	**chns**	**input**
upconv7	3	2	512/512	conv7b
iconv7	3	1	1024/512	upconv7 + conv6b
upconv6	3	2	512/512	iconv7
iconv6	3	1	1024/512	upconv6 + conv5b
upconv5	3	2	512/256	iconv6
iconv5	3	1	512/256	upconv5 + conv4b
upconv4	3	2	256/128	iconv5
iconv4	3	1	128/128	upconv4 + conv3b
disp4	3	1	128/2	iconv4
upconv3	3	2	128/64	iconv4
iconv3	3	1	130/64	upconv3 + conv2b + disp4*
disp3	3	1	64/2	iconv3
upconv2	3	2	64/32	iconv3
iconv2	3	1	66/32	upconv2 + conv1b + disp3*
disp2	3	1	32/2	iconv2
upconv1	3	2	32/16	iconv2
iconv1	3	1	18/16	upconv1 + disp2*
disp1	3	1	16/2	iconv1

Table C.1: Naive network architecture. + represents concatenation. *: Disparity layers are also concatenated to some activations in the decoder, but are upscaled by a factor of two.

C.2 Multiview Network Parameters

Encoder				
layer	**k**	**s**	**chns**	**input**
conv1	7	2	3/32	left
conv1b	7	1	32/32	conv1
conv2	5	2	32/64	conv1b
conv2b	5	1	64/64	conv2
conv3	3	2	64/128	conv2b
conv3b	3	1	128/128	conv3
conv4	3	2	128/256	conv3b
conv4b	3	1	256/256	conv4
conv5	3	2	256/512	conv4b
conv5b	3	1	512/512	conv5
conv6	3	2	512/513	conv5b
conv6b	3	1	512/514	conv6
conv7	3	2	512/515	conv6b
conv7b	3	1	512/516	conv7

Decoder				
layer	**k**	**s**	**chns**	**input**
upconv7	3	2	512/512	conv7b
iconv7	3	1	1024/512	upconv7 + (conv6b_1 - conv6b_0)
upconv6	3	2	512/512	iconv7
iconv6	3	1	1024/512	upconv6 + (conv5b_1 - conv5b_0)
upconv5	3	2	512/256	iconv6
iconv5	3	1	512/256	upconv5 + (conv4b_1 - conv4b_0)
upconv4	3	2	256/128	iconv5
iconv4	3	1	128/128	upconv4 + (conv3b_1 - conv3b_0)
disp4	3	1	128/2	iconv4

Table C.2 continued from previous page

upconv3	3	2	128/64	iconv4
iconv3	3	1	130/64	upconv3 + (conv2b_1 - con2b_0) + disp4*
disp3	3	1	64/2	iconv3
upconv2	3	2	64/32	iconv3
iconv2	3	1	66/32	upconv2 + (conv1b_1 - conv1b_0) + disp3*
disp2	3	1	32/2	iconv2
upconv1	3	2	32/16	iconv2
iconv1	3	1	18/16	upconv1 + disp2*
disp1	3	1	16/2	iconv1

Table C.2: Multiview network architecture. The network has two parallel encoders that have inputs left_0 and left_1 respectively. The decoder has skip connections which are the element-wise difference of the specified layers in the encoders. For example, conv6b_1 - conv6b_0. + represents concatenation. *: Disparity layers are also concatenated to some activations in the decoder, but are upscaled by a factor of two.

Appendix D

Table of Results

Table D.1: Table of results for all experiments

		Lower is better					Higher is better		
Model (train dataset)	**Test dataset**	**abs_rel**	**sq_rel**	**rms**	**log_rms**	**scale_inv**	**a1**	**a2**	**a3**
monodepth (kitti)	kitti	0.0869	1.184	4.998	0.165	0.029	0.918	0.969	0.986
multiview (kitti)	kitti	0.1203	1.7182	6.048	0.200	0.0406	0.874	0.955	0.981
naive (kitti)	kitti	0.0834	1.1386	4.906	0.161	0.0275	0.921	0.971	0.987
monodepth (flying)	kitti	0.584	8.1656	15.413	1.186	0.3719	0.100	0.201	0.306
multiview (flying)	kitti	0.4499	6.5321	14.107	0.849	0.3019	0.22	0.412	0.569
constant	kitti					0.34			
monodepth (kitti)	flying	0.7906	19.8081	18.755	0.983	0.561	0.127	0.255	0.391
multiview (kitti)	flying	0.6872	16.4283	18.218	1.035	0.337	0.108	0.222	0.344
naive (kitti)	flying	0.7885	29.4412	19.177	0.955	0.534	0.131	0.265	0.405
monodepth (flying)	flying	0.512	144.0073	27.363	0.453	0.188	0.488	0.768	0.888
multiview (flying)	flying	0.7197	284.2464	57.375	0.502	0.225	0.483	0.755	0.877
constant	flying					0.231			
monodepth (kitti)	rover	3.0047	57.1109	15.482	1.321	0.2544	0.039	0.091	0.176
multiview (kitti) [skip = 1]	rover	2.9804	60.6089	17.206	1.329	0.3548	0.066	0.147	0.236
multiview (kitti) [skip = 10]	rover	1.7103	22.5902	9.95	0.997	0.4061	0.141	0.285	0.424
naive (kitti)	rover	3.1297	63.1636	16.094	1.341	0.2528	0.036	0.09	0.176
multiview (flying) [skip = 1]	rover	0.5498	2.6881	4.068	0.607	0.3342	0.247	0.494	0.707
multiview (flying) [skip = 10]	rover	0.4774	2.2617	4.144	0.631	0.3234	0.244	0.479	0.682
monodepth (flying)	rover	0.5859	3.2786	5.247	1.036	0.331	0.072	0.183	0.341
constant	rover					0.2986			
multiview (kitti)	umich	0.7337	11.2404	12.331	0.645	0.2986	0.27	0.521	0.700
monodepth (kitti)	umich	0.8796	16.5257	13.63	0.700	0.3413	0.262	0.507	0.673
naive (kitti)	umich	0.8962	16.312	13.52	0.692	0.3227	0.252	0.497	0.673
monodepth (flying)	umich	0.758	13.2022	17.252	1.557	0.6431	0.069	0.139	0.215
multiview (flying)	umich	0.5756	9.4867	15.132	0.97	0.4561	0.191	0.362	0.508
constant						0.413			

Bibliography

[1] Patrizia Baraldi, Enrico De Micheli, and Sergio Uras. Motion and depth from optical flow, 1989.

[2] Nicholas Carlevaris-Bianco, Arash K. Ushani, and Ryan M. Eustice. University of Michigan North Campus long-term vision and lidar dataset. *International Journal of Robotics Research*, 35(9):1023–1035, 2015.

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[4] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2366–2374. Curran Associates, Inc., 2014.

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

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[7] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *CoRR*, abs/1609.03677, 2016.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[9] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. 2013.

[10] et al. Martin Abadi, Paul Barham. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[11] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[12] N.Mayer, E.Ilg, P.Häusser, P.Fischer, D.Cremers, A.Dosovitskiy, and T.Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[13] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[14] Davide Scaramuzza, Friedrich Fraundorfer, Marc Pollefeys, and Roland Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. *2009 IEEE 12th International Conference on Computer Vision*, pages 1413–1419, 2009.

[15] Jess Tordesillas Torres. Comparison of the Snapdragon Flight, Realsense and ZED stereo cameras. Technical report, Aerospace Control Laboratory - MIT, 2017.

[16] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. *CoRR*, abs/1612.02401, 2016.

[17] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Trans. Img. Proc.*, 13(4):600–612, April 2004.

[18] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. *CoRR*, abs/1704.07813, 2017.