Practical Depth Estimation with Image Segmentation and Serial U-Nets

Kyle J. Cantrell, Craig D. Miller

***Abstract*—Knowledge of environmental depth is required for successful robot navigation. Existing research has shown that it is possible to estimate depth from 2D monovision cameras using convolution neural networks. Recent advances suggest that depth estimate accuracy can be improved when networks used for semantic segmentation are incorporated into the network architecture. Further, convolutional layers pre-trained on large image datasets can learn the hierarchical features required for depth estimation and incorporated into a new network via transfer learning. Novel, hybrid architectures are proposed, tuned, and evaluated on a benchmark dataset.**

***Index Terms*—deep learning, depth estimation, neural networks, segmentation, transfer learning, U-Net.**

I. Introduction

TYPICAL color monovision cameras provide three data points per pixel in a given image. The data points correspond to the red, green, and blue (RGB) intensity levels (from 0 to 255) present in the image at that specific point. Stereo vision cameras are able to provide a fourth datapoint at each pixel, namely depth. Since depth is required for robots to localize and build maps of their environment, many autonomous robotic systems rely on stereo vision. Although stereo vision cameras can provide the required depth data, they are frequently orders of magnitude more expensive than a single monovision camera. Achieving stereo vision levels of performance from a monovision camera is desirable as they are smaller, cheaper, and less sophisticated. As the cost of computational power continues to plummet, the additional processing power required to use a neural network to extract the depth data from a monovision camera feed does not present a large burden. This paper explores the estimation of depth data strictly from 2D RGB values from a monovision camera.

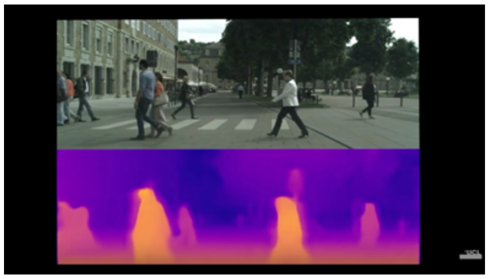
Fig. 1: Depth prediction results on KITTI 2015 [2]

II. Problem Description

An important problem in robotics today is autonomous navigation, an inevitably complex superset of problems involving object detection, trajectory generation, and the colloquially termed ODOA (Obstacle Detection and Object Avoidance). At the heart of this problem is the ability to accurately estimate the depth of an object within a field of view (FOV). This is achievable today with several techniques such as stereovision (using the overlap of two cameras in a computation to triangulate the position of an object in a frame relative to the camera resolutions and position). Another method is pairing a camera with a rangefinding solution like LIDAR, RADAR or ultrasonic sensors. Both of these methods have drawbacks in hardware cost and computational cost. The purpose of this paper is to present a solution to the problem of ODOA via depth estimation with monocular devices, thereby reducing the amount of hardware and computational cost necessary for an autonomously navigated robot.

III. Literature Review

Many researchers have attempted to solve this problem with various levels of success. Early naive approaches involved hand-crafted features and resulted in only modest accuracy. More recent approaches have trained convolutional neural networks (CNNs) to simultaneously predict depth and offer semantic segmentation of an image. A few datasets exist for benchmarking depth estimation. The two most popular sets are KITTI and NYU Depth Dataset V2 (NYUD v2) [1], [13]. Both of these references have been cited thousands of times and are the de facto standards for validating new depth estimation frameworks. State-of-the-art performance on the KITTI dataset is currently a relative square error of 2.00. This result was achieved by Manuel López Antequera (Mapillary, September 2019) and is not yet published. Current state-of-the-art performance on the NYUD v2 dataset is a root-mean-square error of 0.356, achieved by [8]. Performance on both datasets has rapidly improved over the past several years due to the experimentation and implementation of a variety of techniques. Traditional depth estimation networks take RGB values from stereo images as input and the depth values as the ground truth for backpropagation. Once the network is trained, the network can be presented monocular images and it is able to estimate depth. The networks typically utilize an end-to-end, pixel-to-pixel architecture. This means that the size of the input and output vectors will be identical. In [6] and [15], semantic segmentation was shown to improve depth estimation accuracy. [6] uses a hybrid network architecture and defines an attention-driven loss to improve efficacy. Semantic segmentation is a rich field of study on its own and can be approached in many ways. Even though it is a field in its own right, semantic segmentation is an integral part to our objective as the semantic scene labeling is important to our vision of contributing to solving the obstacle detection and avoidance problem. Using methods presented in [16], we hope to utilize semantic segmentation to inform our convolutional network on the correct depth estimation predictions.

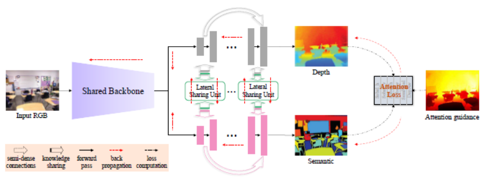

Fig. 2: Network Architecture Combining Depth Estimation and Semantic Segmentation [6]

Given the basis of existing research, we hypothesize that by combining many of the latest techniques, we will be able to achieve near state-of-the-art depth estimation accuracy performance on NYUD v2. Specifically, this model will: Pre-train convolutional layers on the ImageNet dataset Incorporate state-of-the-art semantic segmentation Tune hyperparameters to reduce model error

Employing the practical design process advice detailed in [10], this model will strive to utilize the following outline: Goals: To achieve state-of-the-art accuracy performance on NYUD v2 dataset. Ultimately, we seek to improve robotic autonomous navigation systems by reducing the number of sensors required for ODOA. Pipeline: We will establish a baseline CNN model early. Data from the NYUD v2 dataset will be downloaded, extracted, and organized. Diagnose Performance Bottlenecks: We will diagnose overfitting, underfitting, and software defects. Information loss and accuracy for both the training and test datasets will be measured at each epoch. Hyperparameter Tuning: Several iterations of hyperparameter tuning be performed to identify the values that allow the network to perform best.

The network will be benchmarked on the NYUD v2 dataset. If the network produces accurate output, it could serve as a basis for ODOA and 3D mapping algorithms simply using a monocular camera, the implications of which are current technologies that enable ODOA, such as LiDAR, Radar and Sonar, could be enhanced or replaced. LiDAR, in particular, has many drawbacks but finds itself as one of the more prevalent technologies of choice for this application because of its price point. Some of the drawbacks to this can be seen in [16].

IV. Existing Depth Architectures

Several existing neural network architectures are trained to predict depth from RGB images.

A. Standard CNN

Standard CNN architectures are a default choice for many computer vision tasks such as image classification, digit recognition, and facial recognition. For this application, a sequential CNN with two convolutional layers followed by alternating densely connected and dropout layers was evaluated. Nearly all convolutional and densely connected layers were activated by the ReLU nonlinearity. The output layer was linearly activated.

B. RCNN

Recurrent Convolutional Neural Networks represent a hybrid architecture that can leverage the state-of-the-art in both sequence-based deep learning (RNN) and image classification (CNN). Several commercial and high performance R-CNN models have been successful in related fields such as the R-CNN-192 developed by Ming Liang et al in Recurrent Convolutional Neural Network for Object Recognition. The idea to combine the proven standard Convolutional Neural Network classifier and segmentation ability with a proven sequential predicting Recurrent network was born of the idea that the depth data being trained can inform the LSTM layer(s) that a desired feature to be learned beyond the object segmentation is the relation between the front and the back of the image. An example architecture can be seen in Figure 3.

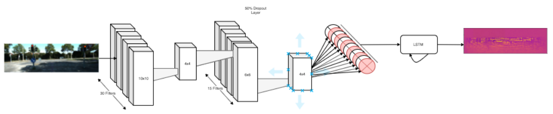

Fig. 3: CNN + LSTM Model

For practical purposes, this is a standard Convolutional Neural Network with a single LSTM layer of 512 Units. The input to the neural network is a single RGB image with a resolution of 640x480 pixels. After two convolutional layers with two pooling layers interleaved in between, a 50 percent dropout layer is inserted. The output of the dropout layer is connected to the LSTM layer input with 1 sample, timestep of 1 and 512 features (the output of the CNN). An alternative RCNN was developed as well where a second hidden LSTM layer was constructed.

C. U-Net

U-Net is an encoder-decoder neural network architecture. The 2D size of U-Net's input and output arrays are equal. U-Net is frequently used for the segmentation of images. Since depth estimation requires a 2D array output equal to the number of input pixels, U-Net is an appropriate model candidate. For segmentation, U-Net is typically appended with a final activation layer for assigning semantic classes. In depth estimation, the output of the final decoding convolutional layer can be taken as a 2D array of 8-bit depth values with which a depth image can be constructed.

U-Net gets its name from the structure of the net itself. The first half of the "U" represents the encoder and the second half represents the decoder. Each step on each side of the U represents a block of convolutional and pooling layers. The output of each of the encoding blocks feeds the next encoding block as well as the analogous decoding block. Higher and lower level image features are available at different blocks of the network by connecting the weights across the U.

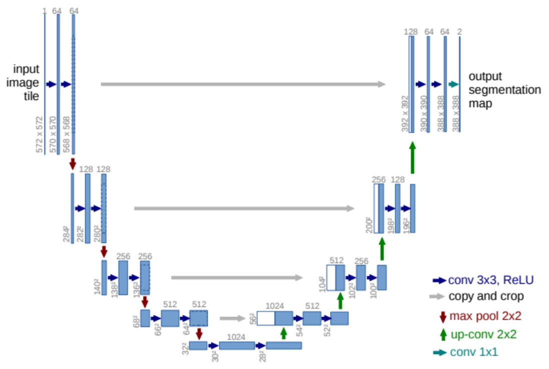

Fig. 4: U-Net Architecture [14]

U-Net architectures can be built using a number of different base models. ResNet, MobileNet, and VGG are popular base model choices. In this experiment, a U-Net architecture with a ResNet 34 base model was loaded with weights pre-trained on the ImageNet dataset. The encoder weights were frozen during the depth estimation training to preserve the feature extractors learned for image segmentation.

V. Experimental Architectures

Four new hybrid architectures are presented here. Each architecture is a combination of existing architectures.

A. *U-Net + CNN*

The U-Net + CNN architecture is a standard CNN appended to the output of a U-Net model (less the final activation layer). The U-Net model is pre-trained to perform segmentation on the ImageNet dataset. The U-Net encoder weights will be frozen during depth estimation training.

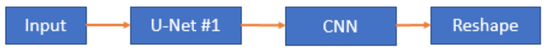

Fig. 5: U-Net + CNN Architecture

B. *U-Net + RCNN*

The U-Net + RCNN architecture is an RCNN model appended to the output of a U-Net model (less the final activation layer). The U-Net model is pre-trained to perform segmentation on the ImageNet dataset. The U-Net encoder weights will be frozen during depth estimation training.

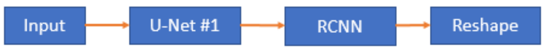

Fig. 6: U-Net + RCNN Architecture

C. *W-Net*

W-Net is a proposed architecture for leveraging semantic segmentation for improved depth estimation. W-Net is composed of two U-Nets in series. The first U-Net is a pre-trained network to perform segmentation. The output from the first U-Net (less the final activation layer) is fed into the beginning of a second untrained U-Net. During the depth estimation training phase, the weights of all of the layers in the first U-Net are frozen.

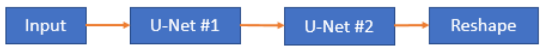

Fig. 7: W-Net Architecture

D. *W-Net Connected*

W-Net Connected is very similar to W-Net with the exception of a connection between the RGB input to the input of the second U-Net. In this way, the second U-Net sees the original image as well as the output from the first U-Net. By connecting the second U-Net to both of these layers, W-Net Connected is able to utilize a pre-segmented image (from U-Net #1 output) to help inform the depth estimate without losing any information from the directly connected original image.

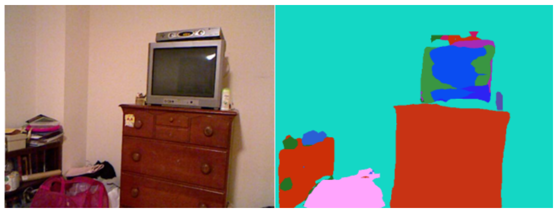

Fig. 8: Original RGB Image and Corresponding Segmentation from U-Net #1

After concatenating the RGB input with the first U-Net output, it is reshaped to (480,640,4) and passed to the second U-Net.

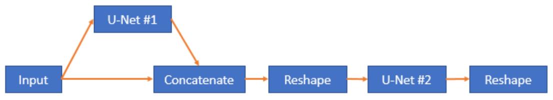

Fig. 9: W-Net Connected Architecture

E. *Loss Functions*

During the research phase, it became apparent that experimenting with the Loss functions was necessary. As noted in [3] and [7], Scale Invariant Error is a log-based objective function that works by penalizing in log steps the percent error of the predicted output versus the ground truth. In addition, it penalizes less when the direction of the output is consistent with the direction of the ground truth. The computation can be seen in (1).

$$L(y, y^*) = \frac{1}{n}\sum_{i}^{n}(d_i)^2 - \frac{\lambda}{n}(\sum_{i}^{n}(d_i))^2 \tag{1}$$

Where y is the predicted output, y^* is the ground truth, $d_i = log(y_i) - log(y^*)$, and λ is a real number between [0,1). Noteworthy for this algorithm is that the raw depth training data does not react well when trained with this loss function because of the grayscale and discrete nature of the image, even when normalized. To deal with this, the log of the predicted and ground truth pixels were "clipped", or, limited to the value of ϵ (1e-07). It was determined during research that this loss function was not compatible with the training data in an unprocessed, or raw format.

Mean Squared Error was eventually chosen as the best performing Loss function and standard performance metric to measure. The equation for Mean Squared Error can be seen in (2).

$$L(y, y^*) = \frac{1}{n} \sum_{i}^{n} (y_i - y_i^*)^2 \tag{2}$$

This is a straightforward calculation measuring the error, per-pixel, between the ground truth and the predicted output image. It is worth noting that the state-of-the-art in this problem space is generally measured in Linear Root Mean Squared Error, Root-Mean-Squared-Log-Error, and Absolute Relative Tolerance.

VI. Methodology

The existing and experimental networks were benchmarked using the linear Mean Squared Error (MSE) loss function and the Adam Optimizer, an adaptive optimization algorithm. This is an appropriate selection for a loss function given that depth estimation is a regression task. Also evaluated was Scale Invariant Loss, a popular log error calculation that includes a "directional" term to correct the gradients with a finer granularity than mean squared error alone. This can be seen in depth in [3]. A learning rate of 0.001 was used on initial evaluations. All models were evaluated using a subset of the NYUD v2 dataset.

A. Evaluation Setup

8,600 RGB and depth image pairs from the NYUD v2 dataset were used for depth estimation training. A selection of 241 RGB and depth pairs were used for validation. All training and test data - both RGB and depth - were normalized from a 0-255 to 0-1 scale. The training dataset was separated into 40 batches to minimize the memory required for training. All models were trained on a NVIDIA RTX 2060.

VII. Results and Discussion

A. Performance Benchmarks

All architectures were first evaluated by training them for 1 epoch per training batch, resulting in 40 total epochs. Most networks were trained with a batch size of 2, with the exception of CNN, RCNN, and U-Net which were increased to 6. A small batch size was used to prevent memory errors during training. Minimum validation MSEs are logged in the tables below from the 40 Epoch training process.

After training, inference times for each model were measured on a desktop with an i5-4460 processor, 8 GB of RAM, and a NVIDIA RTX 2060. Average, minimum, and maximum inference times were logged after performing 100 model predictions. Each model used the same RGB input image for each iteration. However, the different models did not use the same image.

The architectures with the best depth estimation accuracy were then evaluated in a second trial. Each architecture was trained for 5 epochs per training batch, resulting in 200 total epochs.

Regression Performance		
Model Architecture	***MSE***	***RMSE***
CNN	0.0533	0.2309
RCNN	0.0531	0.2304
U-Net	0.0453	0.2128
U-Net + CNN	0.0528	0.2298
U-Net + RCNN	0.0544	0.2332
W-Net	0.0592	0.2433
W-Net Connected	0.0552	0.2349

TABLE I: Regression Results, NYUD v2: 40 Epochs

Inference Benchmarks			
Model Architecture	***Average (ms)***	***Min. (ms)***	***Max. (ms)***
CNN	11.34	10.91	27.83
RCNN	41.49	38.06	54.96
U-Net	38.94	37.54	60.57
U-Net + CNN	43.73	42.10	67.51
U-Net + RCNN	64.22	59.99	78.30
W-Net	69.98	68.26	98.24
W-Net Connected	70.58	68.78	98.07

TABLE II: Inference Benchmarks, NYUD v2: 40 Epochs

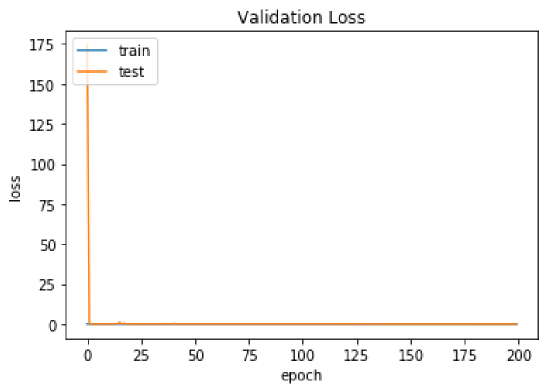

Fig. 10: Mean-Squared Error for U-Net Architecture, 200 Epochs

B. Hyperparameter-tuned Models

After benchmarking several models, the W-Net Connected was selected for further evaluation given its performance and architecture. Looking at the MSE plots from 200 Epoch trial,

Regression Performance		
Model Architecture	***MSE***	***RMSE***
U-Net	0.0447	0.2114
W-Net Connected	0.0484	0.2200

TABLE III: Regression Results, NYUD v2: 200 Epochs

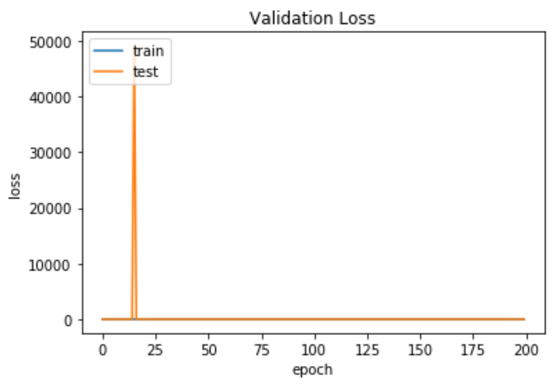

Fig. 11: Mean-Squared Error for W-Net Connected Architecture, 200 Epochs

Inference Benchmarks			
Model Architecture	***Average (ms)***	***Min. (ms)***	***Max. (ms)***
U-Net	39.60	37.96	60.71
W-Net Connected	70.33	68.89	96.33

TABLE IV: Inference Benchmarks, NYUD v2: 200 Epochs

Inference Benchmarks			
Model Architecture	***Average (ms)***	***Min. (ms)***	***Max. (ms)***
W-Net Connected	69.53	67.75	97.80

TABLE VII: Inference Benchmarks, NYUD v2: 640 Epochs

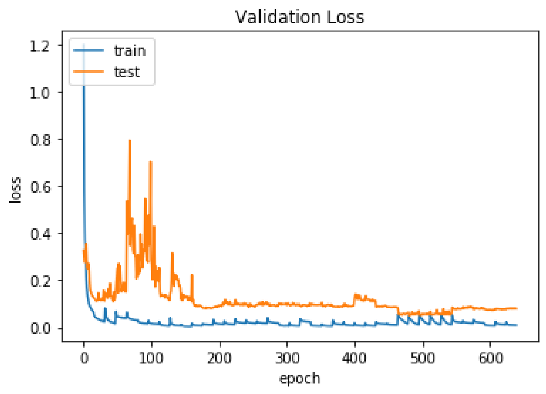

Fig. 12: Mean-Squared Error for W-Net Connected Architecture, 640 Epochs

it seems that the validation loss quickly dropped and then stabilized without much further improvement throughout the majority of the training process. This is indicative of a high learning rate. By reducing the initial learning rate we can improve model performance. The learning rate was reduced from 0.001 to 0.00001. Results are presented within this section.

Oddly, the MSE for W-Net Connected did not decrease in comparison to the 200 epoch trials. This suggests that further hyperparameter tuning is required. Additional performance gains could be realized from utilizing more training data and data augmentation.

C. Visualizations and Intuition

Throughout the training process, depth prediction images were logged after each training batch. As expected, the images gradually transitioned from being generally amorphous to a well-defined image matching the edges in the RGB input image. In intermediate steps, irregular shapes can be seen forming over some of the distinct features in the input image.

D. Observations and Discussion

During the training process, several “phantom features” were observed in the models predicted output. For instance, the models were observed to produce silhouettes of chairs and tables in depth predictions of RGB images that did not include these items. This phenomenon exhibits learned features from previous training sessions indicating an underfitting effect which could be mitigated with proper data augmentation techniques. The “ghost chairs problem”, as it has become colloquially known, is addressed with tenuous hyper parameter tuning, longer training sessions, and more data. In some cases, the networks will produce outputs which exhibit fairly accurate segmentation results, but not entirely correct depth prediction. The trees in the background of the image below are mistakenly estimated to be closer to the observer than the two individuals in the picture.

While training the standard CNN models, it was observed that all of the predicted depth images looked nearly identical regardless of the input RGB image. It is believed that this is a symptom of an underfit model that is simply producing a best guess for image depth without having learned the impact of many of the underlying features. As shown in figure 15, this results in near-depth along the base of the image and far-depth across the top and center. Accurate results were not achieved using a standard CNN.

U-Net and W-Net Connected qualitatively appeared to predict the most accurate depths, despite the MSE score. Unlike the standard CNN, depth predictions from these two networks were highly responsive to variations in the input image.

Hyperparameters			
Model Architecture	***Epochs***	***Batch Size***	***Learning Rate***
W-Net Connected	640	2	0.00001

TABLE V: Model Hyperparameters

Regression Performance		
Model Architecture	***MSE***	***RMSE***
W-Net Connected	0.0511	0.2261

TABLE VI: Regression Results, NYUD v2: 640 Epochs

VIII. Conclusion

In conclusion, the best performing models were U-Net and W-Net Connected. Despite the very closely clustered MSE values reported in Table 1, there is a stark difference in the individual model prediction accuracy. Predictions from CNN and RCNN models were of no value and did not vary when given different input images. Even when the CNN and RCNN models were appended to U-Net, there was not much improvement over the standard models. W-Net Connected appeared to

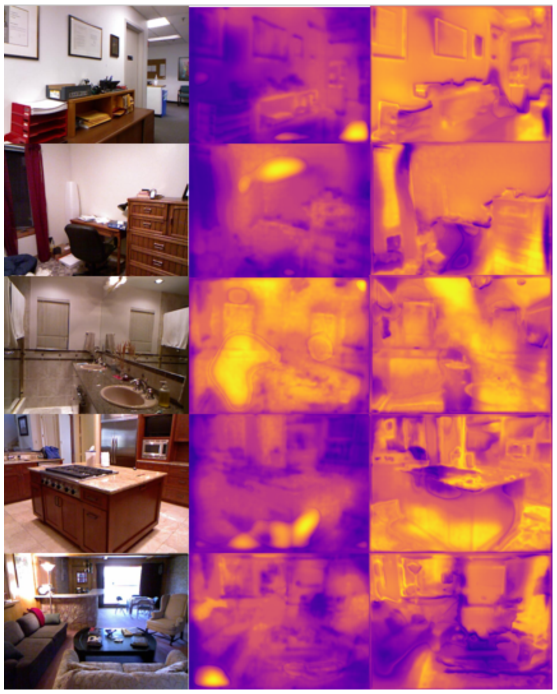
Fig. 13: Left to Right: RGB Input Images, U-Net Depth Predictions (40 Epochs), W-Net Connected Depths Predictions(40 Epochs)

Fig. 14: Input RGB Image for W-Net Connected Architecture

outperform W-Net due to its connection to the original RGB input. Qualitatively, U-Net and W-Net Connected provided the best depth predictions when given different test images. Throughout the training process, both models can be observed as converging toward a correct solution. There is no doubt that the segmentation capabilities provided by a pretrained U-Net was able to improve depth estimation accuracy. W-Net Connected logged the longest inference times with an average runtime of 70ms. U-Net inference times were roughly half of that figure.

A. Future Work

To make pragmatic use of depth estimating neural networks, we intend to develop and maintain a Robot Operating System (ROS) package that can utilize state-of-the-art depth estimation networks to take in a live video feed and publish point-cloud depth topics for ODOA/3D mapping. This could simplify and democratize the integration of depth estimation neural networks into robotic systems. Rigorous characterization of the presented architectures will be completed with the full NYUD v2 datasets using losses as defined in [15]. Features that can be added to the system that will improve its efficacy include Data Augmentation, Custom Loss Function Design, Batch Data Generation for larger, integrated datasets, for instance training on KITTI, NYU and FIELDSafe. Quick improvements can be made by experimenting with more segmentation techniques and even further improved by incorporating semantic segmentation systems such as YOLOv3 or other FRCNN based networks as proven in [6] and [15]. Further architectures to be tested and evaluated include the addition of further LSTM/GRU hidden units such as W-Net Connected + LSTM. Preliminary research in this paper and in accompanying references suggest the time series nature of moving depth image and the interpolated data points in the current datasets can benefit from memory units when deducing depth among sparsely populated depth maps. Another path to take, illuminated by the work done in this paper, is exploring the use of autoencoders for representation learning of depth data to improve the inference time of this system. The end result of which would be the practical real-time production of depth data fed into a generic package for autonomous robotic systems equipped with obstacle detection and avoidance.

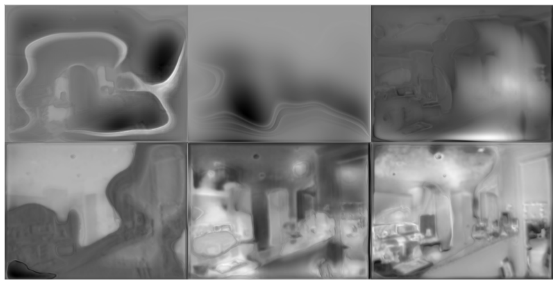
Fig. 15: Evolution of Depth Predictions during training of W-Net Connected Architecture

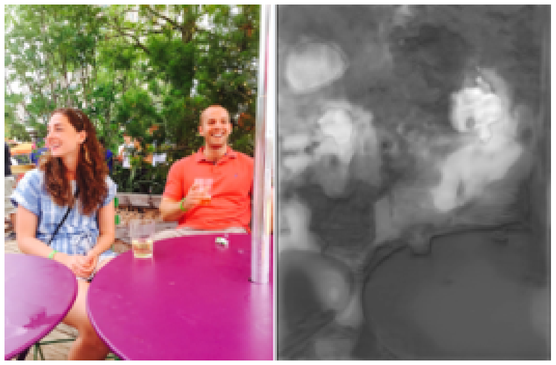
Fig. 16: Correct Segmentation with Inaccurate Depth

Appendix A

All network models discussed in this paper are available at https://github.com/mech0ctopus/depth-estimation.

Fig. 17: Prediction from an underfit CNN model

Acknowledgment

We thank Dr. Carlos Morato for his technical guidance on this project and throughout the semester. We also thank Worcester Polytechnic Institute (WPI) for providing access to the ACE GPU Cluster.

References

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” The International Journal of Robotics Research, vol. 32, no. 11, pp. 1231–1237, 2013.

[2] C. Godard, O. M. Aodha, and G. J. Brostow, “Unsupervised Monocular Depth Estimation with Left-Right Consistency,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[3] D. Eigen, C. Puhrsch, and R. Fergus. ”Depth map prediction from a single image using a multi-scale deep network.” Advances in neural information processing systems (NIPS). 2014.

[4] F. Liu, C. Shen, and G. Lin, “Deep convolutional neural fields for depth estimation from a single image,” 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[5] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, “Deep Ordinal Regression Network for Monocular Depth Estimation,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[6] J. Jiao, Y. Cao, Y. Song, and R.W. H. Lau. Look Deeper into Depth: Monocular Depth Estimation with Semantic Booster and Attention-Driven Loss. In European Conference on Computer Vision, 2018.

[7] J. Lee, and C. Kim. ”Monocular Depth Estimation Using Relative Depth Maps.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

[8] J. Lee, et al. ”From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation.” arXiv preprint arXiv:1907.10326 2019. H. Touvron, et al. ”Fixing the train-test resolution discrepancy.” arXiv preprint arXiv:1906.06423. 2019.

[9] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. Cambridge, MA: MIT Press, 2017.

[10] M. Kragh, P. Christiansen, M. Laursen, M. Larsen, K. Steen, O. Green, H. Karstoft, and R. Jørgensen, “FieldSAFE: Dataset for Obstacle Detection in Agriculture,” Sensors, vol. 17, no. 11, p. 2579, Nov. 2017.

[11] M. Ramamonjisoa, and V. Lepetit. ”SharpNet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation.” arXiv preprint arXiv:1905.08598 (2019).

[12] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor Segmentation and Support Inference from RGBD Images,” Computer Vision – ECCV 2012 Lecture Notes in Computer Science, pp. 746–760, 2012.

[13] O. Ronneberger, P. Fischer, and T. Brox. ”U-net: Convolutional networks for biomedical image segmentation.” International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[14] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. Yuille, “Towards unified depth and semantic prediction from a single image,” 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[15] Yann.lecun.com,2019.[Online]. Available: http://yann.lecun.com/exdb/publis/pdf/farabet-pami-13.pdf. [Accessed: 03- Nov- 2019].

[16] ”‘Anyone relying on lidar is doomed,’ Elon Musk says – TechCrunch”, TechCrunch, 2019. [Online]. Available: https://techcrunch.com/2019/04/22/anyone-relying-on-lidar-is-doomed-elon-musk-says/. [Accessed: 04- Dec- 2019].

Kyle J. Cantrell Kyle Cantrell is a Master’s Student at Worcester Polytechnic Institute in Worcester, MA studying Robotics Engineering. His professional career has spanned several areas of interest including advanced R&D for government agencies, Industrial Automation firmware development, Network Security Consulting and Machine Learning development. His areas of interest personally focus on Deep Learning and Artificial Intelligence. In his spare time, Kyle can be found with his wonderful Wife, Kerri, and two daughters, Callie and Fiona, or breaking a sweat at the gym.

Craig D. Miller Craig Miller is currently pursuing a Master’s degree in Robotics Engineering from Worcester Polytechnic Institute. He holds a B.S. in Mechanical Engineering from Temple University (2013). With WPI’s Music, Perception, and Robotics (MPR) lab, he is developing a quasi-real-time pitch detection system leveraging a state-of-the-art convolutional neural network. Craig has spent his professional career designing, simulating, and programming electromechanical hardware. His current interests include Robot Operating System (ROS), robot perception, and deep learning.