Depth from Single Monocular Images

Sheng Y. Lundquist

June 2, 2016

Abstract

We aim to build a Convolutional Neural Network (CNN) to achieve depth inference to reproduce the unary potential of [1]. We first over-segment the image into superpixels using SLIC, and grab a crop of the image around the superpixel as the input to the CNN, and use the average depth of the superpixel as the ground truth. We show comparable results to the paper's reported error scores on a new dataset.

1 Introduction

Information about how far an object is from the camera is extensively used in the field of computational photography. Many algorithms from the field depend on depth information, such as video stabilization [2], 3D media production [3], 3D modeling [4], just to name a few. However, most current work in this field require multiple views either from stereopsis or from video. Many images, such as ones found on Google Image Search, contain only a single snapshot of the scene. However, biological systems can easily tell depth from monocular cues, such as occlusion, size, texture gradients, perspective, and object knowledge. Our goal is to leverage monocular cues be able to extract 3D depth information from static monocular images.

Convolutional Neural Networks (CNN) have been extremely successful in achieving state of the art results on image understanding tasks. For example, CNN have successfully achieved human level performance[1] in an image classification task [5]. One advantage of such a network is that it is application agnostic; the same network trained on image classes can be retrained to achieve depth inference. As encoding expert knowledge system for extracting monocular cues seems intractable and brittle (we as humans determine depth subconsciously), we aim to train a CNN on the task of depth inference.

2 Methods

The goal of this project is to reproduce the unary potential results of [1]. All code was written in Python, and is available at [6].

We use the 2012 stereo dataset from KITTI [7]. The dataset consists of 194 stereo images with LIDAR annotations, retrieved from cameras mounted on a car. Since we are interested in depth from monocular images, we only use the left camera for all experiments. The former 150 images were taken to be the training set, with the latter 44 images held out to be the test set.

Given an input image, we aim to over-segment the image into superpixels. The superpixel algorithm used was Simple Linear Iterative Clustering (SLIC) [8]. As the project used superpixels as a preprocessing step, we used the SLIC implementation in SciKit-Image. Here, input parameters used was 1000 as the number of segments and 10 as the compactness of superpixels. We take a crop of 224×224 pixels centered around the centroid of the superpixel as the input to the CNN. If the crop is outside of the image area, we pad the image crop with 0. Since the original LIDAR depth annotations contains "do not care" regions, we find the mean of all valid depth annotations within a superpixel as the depth for that superpixel, provided to the CNN as

[1] Human level performance is achieved in a very restricted domain. Some categories in ILSVRC require specific domain knowledge, e.g. there are 120 dog breed categories.

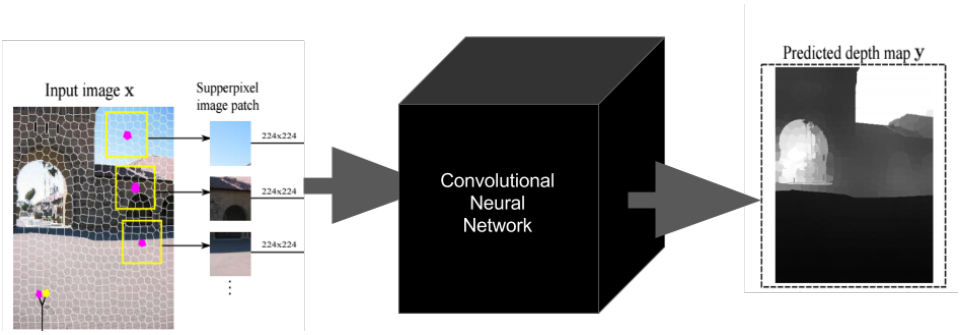

Figure 1: The pipeline of the CNN. We over-segment the image into superpixels and take a 224×224 pixel crop as the input to the CNN. The ground truth value is set to be the mean of the superpixel groundtruth.

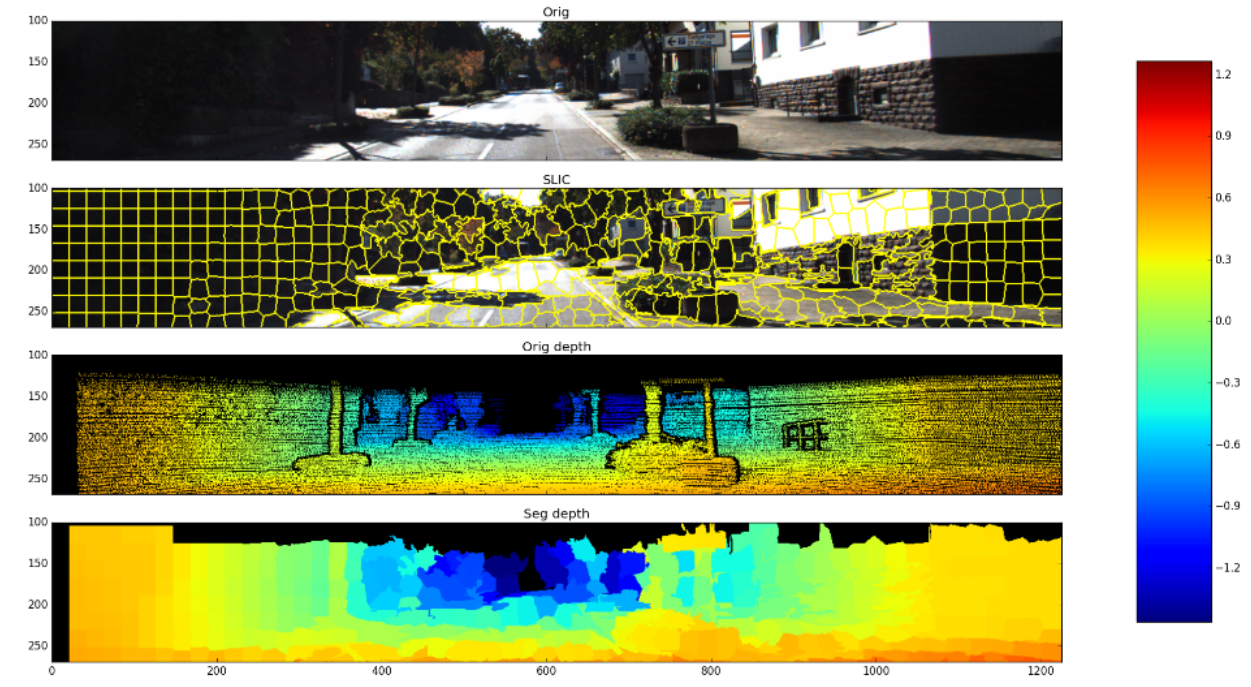

Figure 2: Figure best viewed in color. From top to bottom: (1) An example of an image from the KITTI dataset. (2) SLIC superpixels on the input image. (3) Provided LIDAR data, normalized on a log scale. Black regions are "do not care" regions. (4) Averaged depth map based on superpixels. Black regions are "do not care" regions.

the ground truth. All depth was normalized as $\ln(\frac{d}{\sigma})$, where d is the depth value provided by the dataset, and σ is the standard deviation of depths on the training set. Figure 1 shows the pipeline of the method. Figure 2 shows an example of an input image, its segmentation, the corresponding LIDAR normalized depth map, and the depth map averaged over superpixels.

We now provide details about the CNN. All CNN implementation was done using Tensorflow [9]. Figure 3 shows a detailed schematic of the convolutional neural network. We apply dropout in training to the first 2 fully connected layers with a probability of 0.5 to prevent overfitting. We use the least square loss as follows:

$$L = \frac{1}{2}(y_p - z_p(\Theta))^2 \tag{1}$$

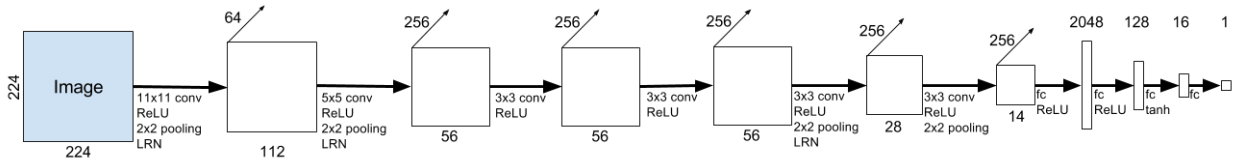

Figure 3: Detailed network architecture of CNN. LRN is a local response normalization as described in [12].

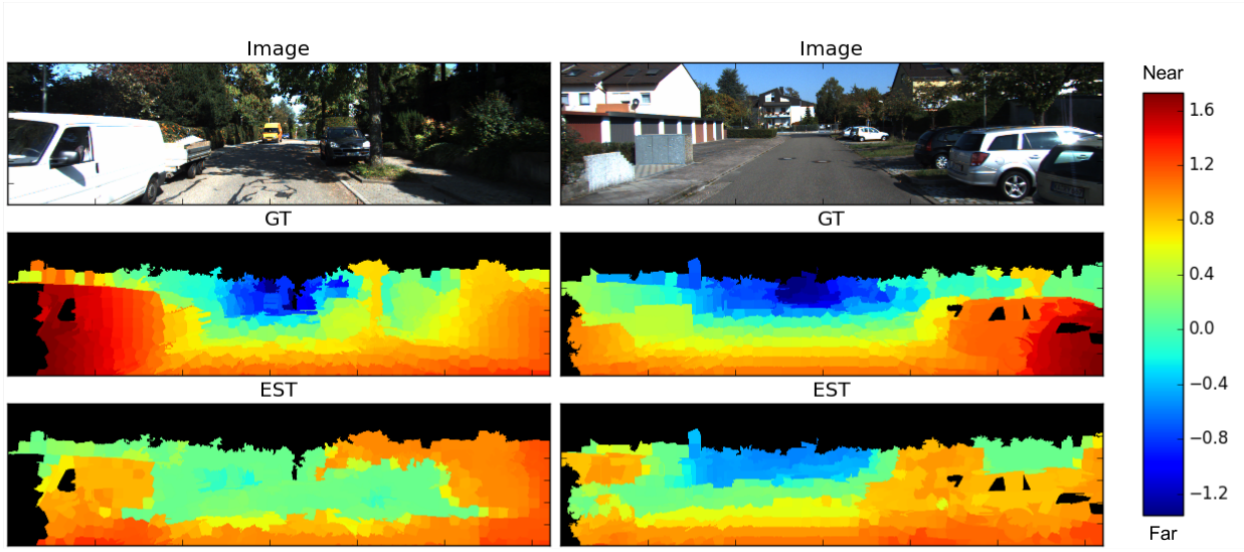

Figure 4: Output from evaluation on test images of the CNN are shown. Depth ground truth and estimates are shown in log scale. Best viewed in color.

Here, y_p is the provided depth at superpixel p in log scale and z_p is the output at superpixel p of the CNN parameterized by Θ. We train Θ using the Adam optimizer [10]. We follow [1] and seed the weights of the first 5 convolutional layers with the weights from [11]. Note that these weights are not trained on a depth inference task. Instead, weights from the model were trained on an image classification task on ILSVRC. Intuitively, incorporating the model with some notion of what objects are (e.g. cars and trees) may provide information about the depth of the object. We set the first 5 convolutional layers to be constant while training the rest of the network until convergence. We then continue training, updating all parameters throughout the network. Due to a constraint on time, the model was only able to train two times through the training set.

There are several differences between [1] and the implemented model. Namely, we added an additional convolutional layer with max-pooling, as well as reducing the first fully connected layer features by half. This was done to ease memory requirements of the model. Additionally, we change the logistic activation function on the 3rd fully connected layer to be a hyperbolic tangent due to the depth being on a log scale. Finally, we add an additional normalization to the output of the 5th convolutional layer due to the ranges of activations diverging during training.

3 Results

Figure 4 shows two example test images from the evaluation of the CNN. In the left example, we find that the network successfully detected the van to be in the foreground. In the right example, the model successfully finds the center of the image to be far away.

We aim to quantitatively determine how successful the model is. Liu et al. [1] uses 3 types of evaluation criteria.

$$\text{Average relative error (rel): } \frac{1}{T}\sum_{p}\frac{\|g_p - e_p\|}{g_p} \tag{2}$$

$$\text{Average log10 error (log10): } \frac{1}{T}\sum_{p}\|\log_{10} g_p - \log_{10} e_p\| \tag{3}$$

$$\text{Root mean squared error (rms): } \sqrt{\frac{1}{T}\sum_{p}(g_p - e_p)^2} \tag{4}$$

Here, T is the total number of superpixels in the test set, g_p is the ground truth depth as provided at superpixel p, and e_p is the output of the CNN (reverting the log normalization) at superpixel p. Table 1 shows the results of the test set on KITTI as compared to that of results reported by [1]. Here, NYU v2 is an indoor scene dataset, which we hypothesize as an easier task for depth inference. Make3D is an outdoor dataset, which is more comparable to that of KITTI. We note that our results are typically worse than that published in [1]. We hypothesize that this is due to the reduction of features in the first fully connected layer, as the model with the original number of features in that layer did not fit into memory on machine used for experiments. Additionally, the model was not trained until convergence due to time constraints, as additional training may provide better quantitative results.

Error (lower is better)			
	rel	log10	rms
NYU v2 [1]	0.295	0.117	0.985
Make3D [1]	0.366	0.137	8.63
KITTI (this work)	0.402	0.150	20.47

Table 1: Baseline results as published by [1] on the unary potential, as compared to the results from this work.

4 Conclusion

We have built a model that achieves depth inference from single monocular images, based on the work of [1]. We successfully infer depth on a novel dataset, and show comparable results to that reported by [1]. Future work includes implementing the pairwise potential that enforces a constraint on neighboring superpixels, as well as training and evaluation on other datasets.

References

[1] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5162–5170.

[2] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 1, p. 4, 2011.

[3] F. Zilly, J. Kluger, and P. Kauff, "Production rules for stereo acquisition," *Proceedings of the IEEE*, vol. 99, no. 4, pp. 590–606, 2011.

[4] P. Beardsley, P. Torr, and A. Zisserman, "3d model acquisition from extended image sequences," in *Computer Vision—ECCV'96*. Springer, 1996, pp. 683–695.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[6] S. Lundquist, "Superpixeldepth," https://github.com/slundqui/superpixelDepth, 2016.

[7] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels," Tech. Rep., 2010.

[9] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[10] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[11] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.