Towards Generalizable Distance Estimation By Leveraging Graph Information

John Kevin Cava Todd Houghton Hongbin Yu
Arizona State University
jcava@asu.edu, tkhought@asu.edu, Hongbin.Yu@asu.edu

Abstract

Approximating the distance of objects present in an image remains an important problem for computer vision applications. Current SO TA methods rely on formulating this problem to convenience depth estimation at every pixel; however, there are limitations that make such solutions non-generalizable (i.e varying focal length). To address this issue, we propose reformulating distance approximation to a per-object detection problem and leveraging graph information extracted from the image to potentially achieve better generalizability on data acquired at multiple focal lengths.

1. Introduction

Development of a generalized, robust, and scalable method for estimating the distance between a monocular camera and the objects present in an observed scene remains a challenging computer vision problem with numerous applications in robotics. Currently, state of the art (SOTA) methods for Monocular Depth Estimation (MDE), utilize supervised neural network Encoder-Decoder architectures [1]. While such networks are capable of estimating depth at every pixel in an input image, some limitations need to be considered. The supervised nature of an Encoder-Decoder model requires that all input images be restricted to a predefined width and height. Changing input size requires a new model to be trained. Secondly, even if dimensions of the input image remain fixed, representation of object distance depends strongly on the imaging system's focal length. Finally, use of an Encoder-Decoder model presents scalability concerns with increasing image resolution, as depth is computed at every pixel, regardless of a given pixel's contribution to depth segmentation.

The limits posed by Encoder-Decoder frameworks ultimately lead to a generalizability problem for MDE; given the myriad number of imaging hardware configurations available, it would be both costly and unstandardized to train a network for each lens/camera combination. To overcome these limitations, we propose an architecture which learns to predict geometric relationships between different objects in a scene, which we argue is analogous to learning geometric perspective. By understanding how objects vary in size with respect to each other, this understanding can be transferred to other camera specifications. Thus, our main contributions are as follows:

- Reformulation of the distance estimation problem from a depth map regression to a multi-object detection-prediction paradigm, which is better suited to solutions involving Graph Convolution Networks (GCNs)

- Construction and training of a GCN, which takes information from a scene as a graph, to predict object distances

- A dataset that includes varying focal lengths, to compare our GCN method with per pixel regression

2. Related Work

2.1. Monocular Depth Estimation

Early solutions to the problem of Monocular Depth Estimation utilized Multi-Scale deep neural networks [2]. These models were comprised of two components, a Global Coarse-Scale Network, which learns global image features, and the Local Fine-Scale Network, which learns small features. Such models highlight both the remarkable quantity and wide variety of feature types present in a given image when various kernel or ROI scales are considered.

Recent work on Monocular Depth Estimation have still generally utilized an Encoder-Decoder architecture. Some have extended the model by using a U-Net, where different sized convolution layers from the encoder are connected with the convolution layers of the decoder [5, 10]. Other works utilize attention from the output latent representations of an encoder to create a conditional random field model to predict the depth map [13]. Multi-task and usage of multi-frames also have been used to increase accuracy for these depth estimation models [2].

2.2. DisNet

While current state of the art methods regress the depth estimation on the pixels of the input image, there are approaches which directly estimate object distances, such as DisNet [3]. This work presents a system which utilizes YOLO v3 [9] to produce bounding boxes of multiple objects within the image. Once objects have been isolated by YOLO, information pertaining of these objects and their bounding boxes are used as features (e.g height, width, length of the diagonal bounding box, etc.). With this, a feedforward neural network of 3 layers each containing 100 hidden units predicts the estimated distance for each object. However, because DisNet effectively learns relationships between geometric features and the camera's total view field, the architecture is limited by focal length.

2.3. Graph Convolutional Neural Networks

A Graph Convolutional Neural Network takes in a graph $G \in \{V, E, A\}$ and outputs a single classification of the graph, or in a semi-supervised approach, classifies the individual nodes in the network. Importantly, GCNs leverage relations between entities within an input graph to generate a classification or regression. [12].

V is defined as the set of vertices or nodes of the graph. E is defined as the set of edges of the graph. A is defined as the adjacency matrix for the graph which indicates which vertices are connected to each other.

3. Methodology

3.1. Problem Reformulation

State of the art Monocular Depth Estimation methods associate every pixel with a computed depth value. This per-pixel depth map can then be used to estimate the distance of any object present in the image. We propose a new methodology and neural network architecture that can incorporate the strategy of DisNet - using the bounding box information of an object to determine distance - and generalize for multiple entities in order to understand the perspective of a scene. We argue that understanding the geometric relations between multiple objects in a scene at multiple focal lengths would lead to robust Monocular Distance Estimation without the computational overhead imposed by a one-to-one pixel depth map.

3.2. Dataset

In order to test our ideas, we propose the construction of a toy dataset that utilizes a varifocal lens installed on a CMOS camera. To the best of our knowledge, there exists no dataset comprised of images acquired at different focal lengths from one hardware configuration. It should be noted that some datasets such as [5] simulate varying focus lengths through post processing transformations that are based upon many technical assumptions. For our proposed dataset, images will be collected at various focal lengths in an urban setting containing a diverse range of objects such as cars, pedestrians, and bicycles. Objects will then be classified by state-of-the-art object detection neural network models. Structurally, our dataset will be comprised of images from varying locations, where each location contains three batches: 1) short focal length, 2) medium focal length, and 3) long focal length.

3.3. Models

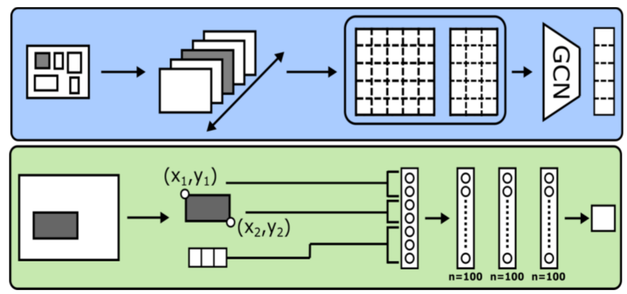

Figure 1. Blue panel represents the GCN model, while the green panel represents DisNet.

We define the architectures of DisNet and our proposed GCN as shown in Figure 1. DisNet is a feedforward neural network which uses a single object's bounding box features (e.g width, height, and length of diagonal line) as input and estimates the distance of that object. We propose using a Graph Convolutional Neural Network (GCN) in order to generalize for multiple objects within the image. For this situation, objects are first detected and classified by a object detection model, and each classification considered the feature of a node. Then for the edges of the graph, the edge weight would be defined as the euclidean difference between the features of object A and object B.

Since there are numerous state-of-the-art Monocular Depth Estimation solutions which utilize an Encoder-Decoder as their core architecture, we will create our own implementation of the Encoder-Decoder model to compare against our distance-per-object estimation method. It may be that our proposed model, which only takes in a graph representing the scene/image may not be adequate. However, if we also construct an Encoder-Decoder model that generally represents the state-of-the-art, we can easily experiment with adding graph information to the model and report any notable outcome. SOTA models estimate depth in a variety of ways (e.g multi-task learning, input sequence through time, etc), but the use of multi-object geometric relationships has not been explored to the best of our knowledge.

4. Experimental Protocol

4.1. Dataset

Raw data will be collected at 10 different sites near a local university, presenting an urban environment which contains pedestrians, vehicles, etc. At each site, data will be collected at short, medium and long focal lengths (10 minutes each, for a total of 5 hours of raw data) using a prefabricated hardware platform consisting of a BASLER puA1280-54uc camera, verifocal lens, and Velodyne HDL32-E LiDAR.

4.2. Models

With 10 different locations, we consider training our models through a 10-fold cross validation, with a MSE loss metric based on per-object LiDAR data. All models will be optimized with ADAM [6] and by a constant learning rate of 1e-3. Pytorch [8] will be used for the construction of these models.

4.2.1 DisNet

For the comparison of DisNet, a feedforward neural network is utilized comprised of 3 hidden layers, each containing 100 neurons. Then we consider training a model for 6 hidden layers, and then 9 hidden layers, each containing 100 neurons, to observe the effects of model scaling when our data set is used for training.

4.2.2 GCN

In the proposed architecture, input images are first passed though an object detection model (i.e YOLO v3) to extract objects and their features. A graph is then constructed from those detections and passed to a GCN for distance estimation. We will follow the GCN model of [7], and construct different depths for the GCN model. We will consider depths of 3, 6, and 9 hidden layers. In addition, we would like to consider how many graph kernels are necessary. Each depth will consider varying kernel sizes of 3,6, and 9.

4.2.3 Encoder-Decoder

For the Encoder-Decoder model, input sizes are equal to the image width and height. We will consider encoder and decoder depths of 3,6, and 9. Output depth maps are superimposed with objects detected by YOLO v3. Object depth is then calculated.

4.2.4 Encoder-Decoder with Graph Information

We will use the same architecture design as the Encoder-Decoder model, but we intend to concatenate scene graph information through the use of a Graph Attention Network [11] (with varying amount of layers, e.g 1,3,and 6), applied to the latent layer after the encoder.

Model	L3	L6	L9
DisNet	25.23m	6.51m	25.33m
GCN	46.64m	174.74m	25.33m
Encoder-Decoder	15.41m	16.30m	14.99m
Graph-Encoder-Decoder	15.18m	14.44m	14.35m
	L6 G1	L6 G3	L6 G6
Graph-Encoder-Decoder	14.44m	15.70m	15.87m

Table 1. Model Test Loss (mean Absolute Distance in meters)

4.3. Evaluation

Models will be evaluated based upon the MSE loss of all the objects within an image. We argue that with our proposed dataset, we can evaluate if some models overfit on a certain focal length if the range of MSE loss values is large. This can be done by creating a boxplot for the models based upon the 10-fold cross validation. Even if we see models that have a particular fold that is the best, but have a large variance of MSE loss, we argue that this model doesn't generalize distance estimation for varying camera specifications. A negative result with the Encoder-Decoder with added graph information would indicate that per-pixel depth regression would be enough to learn the geometric relationship between different objects. A negative result with our GCN method, would indicate bounding box features as not adequate, and one possible future direction could be utilizing representations of points that provide higher level features.

5. Results

5.0.1 Dataset Collection

In accordance with the proposed collection methodology outlined in section section 4.1, synchronous camera and LiDAR data was obtained at 10 distinct sites located on or near Arizona State Universitys campus in Tempe, AZ, USA. Each site contained a mixture of pedestrian, bicycle, and motor vehicle traffic at varying densities in an urban environment. 5 minutes of synchronous footage was recorded using short, medium, and long focal lengths using a varifocal camera lens, resulting in 15 minutes of data per site or 2.5 hours of total footage. All data was collected at 10 frames per second using a Basler HD color camera and a Velodyne HDL-32E high-performance LiDAR unit.

Attentive readers of section 4.1 may note that we were unable to meet the proposed data collection goal of 5 hours. This disparity is primarily due to extreme heat (approx. 43 deg C) and lack of appreciable cloud cover outdoors during

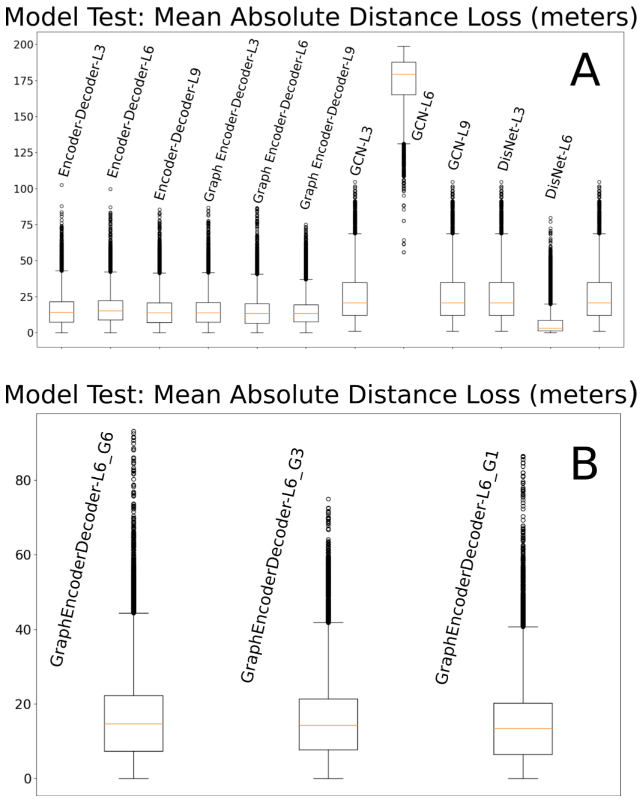

Figure 2. Panel A represents the boxplots of the mean absolute distance loss between DisNet, GCN, Encoder-Decoder, and Graph Encoder-Decoder with varying levels of hidden layers (3,6, and 9). Panel B represents the box plots when Graph Encoder-Decoder with 6 Layers is made constant, and varying layers of GAT is used (1,3 and 6).

late August and early September in the Phoenix metropolitan area. Data with acceptable traffic density for model training was best collected in the afternoon, but collection time was limited to protect experimenters and computer equipment. An additional consideration was the presence of intense, rapidly forming, monsoon rains and dust storms, a common occurrence in the region during early September.

Temporally synchronous images and LiDAR distance data were aligned by first copping 360 degree LiDAR data to the view field of the camera at a given focal length. Next, the cropped point 3D cloud data was transformed from the LiDAR coordinate system (phi, theta, r) to the pixel (x,y) coordinate system of the image, allowing segmented image objects to be associated with LiDAR distance values. Initially, YOLO V3 was to be used as a segmenter since it was used by DisNet. However, when we were aligning the data, and determining which LiDAR points associate with a given object, we decided to use Mask RCNN [4]. This is because Mask RCNN generates form fitting masks of each detected object rather than a bounding box. LiDAR points within the mask are able to better represent the distance of an object, as there a fewer erroneous point associations with the background or other objects in image. Since Mask RCNN is still able to produce the bounding boxes with the masks, we used Mask RCNN for the computation of the bounding boxes for our models. The associated models can be found at our github. [1]

5.0.2 Training

Pytorch was used for the construction and training of the proposed machine learning models outlined in section 4.2. In all models, learning rate constant hyperparameters were set to a value of 1e-3 while ADAM served as the optimizer. Importantly, our experiments diverge from the original proposal regarding the use of 10 fold cross validation. Due to time constraints, a decision was made to train and validate all proposed models equally, rather than perform 10 fold cross validation on a small number of models while leaving most under-validated. In light of this decision, each model underwent a train/validation/test split of 80/10/10. Training was stopped when validation loss fell below a value of one percent. This stopping criteria ensures that all models converge to a similar loss value before they are compared to each other. While this diminishes some ability to make definitive comparison of the models, we were able to train all our proposed architectures.

6. Discussion

Test error loss was computed by taking the absolute value of difference between a given model's predicted object distance and the object's measured ground truth value (also in meters). The mean loss for all objects in the test set for each model is shown in Table 1. As observed in Table 1, our proposed GCN architecture has test loss values for 3,6, and 9 layers which are higher than the DisNet, Encoder-Decoder, and Graph Encoder-Decoder models, indicating worse prediction accuracy for the GCN. For the Encoder-Decoder, test loss values are similar for 3,6, and 9 layers. Intriguingly, the addition of graph information in the Graph Encoder-Decoder model appears to offer a slight improvement in mean absolute distance loss when compared to the standard Encoder-Decoder for the same number of layers. We see that the Graph Encoder-Decoder with 6 layers in the encoder and 6 layers in the decoder achieves lower loss than its Encoder-Decoder counterpart. From these results, we argue that there seems to be a reason that the addition of graph information is able to improve loss. As a supplementary experiment, we took the best Graph Encoder-Decoder from the set of 3, which was 6 layers, and increased the number of GAT layers before concatenation with the convolutional

[1] https://github.com/johncava/iccv_2019

image features of the encoder. The results of this experiment are tabulated in the bottom two rows of Table 1, and it appears that the mean absolute distance loss increases. However, it is interesting to note is that in Figure 2, the box plot for the Graph Encoder-Decoder with 3 GAT layers has exhibits fewer outliers compared to the GATs containing 1 layer and 6 layers.

From Table 1, we see that DisNet with 6 layers exhibits the smallest loss, whereas with 3 and 9 layers, it is worse than both Encoder-Decoder and Graph Encoder-Decoder models. This is also seen in Figure 2, where the box plot of DisNet containing 6 layers shows that 50 percent of the data has a loss less than every other model used. However, this is not the case for DisNet models containing 3 or 9 layers. What this observation implies is that the learning of DisNet, and also by extension our proposed GCN model, are prone to overfitting. Such a conclusion poses a new consideration for future experimenters; when choosing between bounding box and per-pixel estimation methods, it may be wise to regress depth on the pixels of an image rather utilise the limited the geometric features of a bounding box. Per-pixel regression forces the model to make many predictions, and as such, the few errors in the regression are considered negligible compared to the total number of pixels.

An additional point of interest can be found in Table 1. Namely, the mean absolute error distance of the Graph Encoder-Decoder with 9 layers was larger than the Graph Encoder-Decoder of 6 layers. However, as shown in Figure 2, the 9 layer Graph Encoder-Decoder does not generate extreme outliers compared to the one with 6 layers, even when the one with 6 layers has better mean absolute distance loss. Extreme outliers are a concern with regards to both model overffiting and engineering applications such as self driving cars, which rely on consistent distance estimates in a wide variety of situations. These examples demonstrate that with any distance estimation model, both the mean error loss and number of error outliers generated should be considered in a performance assessment. As a final point, we argue that having less extreme error outliers presents a better model, which can avoid critical mistakes in engineering applications.

7. Conclusion

In this paper, we proposed reformulating the monocular depth estimation problem from one of per-pixel regression to one of geometric segment features regressed on ground truth LiDAR data. We proposed the use of a graph convolutional neural network (GCN) as a natural progression from DisNet - previous model that determines distance per object by using bounding box features. In addition, we created a 2.5 hour dataset with varying focal lengths to test our reformulation of the problem. Our results suggest that our proposed model of GCN is not sufficient compared to regressing on pixel values, whereas DisNet is able to in certain layer architectures. Moreover, the addition of graph information to a standard Encoder-Decoder model results in better loss, implying that further research work could be done in utilizing more than just the pixel features of the image to do distance estimation.

References

[1] Amlaan Bhoi. Monocular depth estimation: A survey. *CoRR*, abs/1901.09402, 2019.

[2] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Unsupervised learning of depth and ego-motion: A structured approach. In *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.

[3] Muhammad Abdul Haseeb, Jianyu Guan, Danijela Ristić-Durrant, and Axel Gräser. Disnet: A novel method for distance estimation from monocular camera.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[5] Lei He, Guanghui Wang, and Zhanyi Hu. Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, 27(9):4676–4689, 2018.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

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

[9] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[11] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[12] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[13] Dan Xu, Wei Wang, Hao Tang, Hong Liu, Nicu Sebe, and Elisa Ricci. Structured attention guided convolutional neural fields for monocular depth estimation. *CoRR*, abs/1803.11029, 2018.