EECS 442 Final Report: A Monocular Local Mapper for Urban Scenes

Ruohua Li ruohuali@umich.edu
Carolyn Melvin camelvin@umich.edu
Kshama Nitin Shah kshama@umich.edu
Jiamin Yang jiaminy@umich.edu

Abstract

In this paper we will discuss the process and relevance of building a monocular local mapper in urban settings. We proposed a model that performs semantic segmentation approaches, object detection, and depth estimation simultaneously. We will explain the details of each process, the integration of our techniques, and relevant researches.

1. Introduction

The recent advancements of deep learning in the area of computer vision have shown its great potential to be applied to the development of navigation and motion planning in urban scenes for robots and autonomous vehicles with cameras. In order to enable these downstream tasks, firstly it is required to implement a system that is able to map the urban scenes into objects that should be avoided and drivable open spaces, in the meantime knowing the distance from the camera to these entities. These requirements inspired us to design a Monocular Local Mapper for Urban Scenes that can perform object detection, semantic segmentation, and object detection in the same time.

2. Related Work

In this section, we discuss the relevant methods for Monocular Depth Estimation using semantic segmentation, Depth Estimation and Object Detection.

2.1. Semantic Segmentation

Semantic segmentation is a computer vision task that involves assigning a class to each pixel of an image. It is also known as pixel-wise classification. The semantic segmentation has three steps: 1. Classifying a certain object in the image 2. Localizing the object by drawing a bounding box around that object 3. Segmentation by grouping the pixels in the localized image by creating a segmentation mask. To perform this task, an encoder is used to extract the features. The encoder consists of the convolution and max pooling layers. The size of the image gradually decreases while the depth gradually increases in the encoder. A decoder is used to gradually upsample the extracted features. The size of the image gradually increases while the depth slowly decreases in the decoder. The output from the semantic segmentation is a high resolution image in which each image is classified to a particular class. One of the most popular architectures that were developed to perform semantic segmentation was the U-Net [9] architecture. It is a Fully Convolutional Neural Network with no fully connected layers. It has the design of an encoder known as contract path and decoder known as expansive path. The former is used to extract features by downsampling, while the latter is used for upsampling the extracted features using the deconvolutional layers. The only difference between the FCN (Fully Connected Network) and U-Net is that the FCN uses the final extracted features to upsample, while U-net uses something called a shortcut connection to do that. Our method uses three steps on each path, each horizontal block is composed of 2 convolutional layers. The upsampling and downsampling operations are done by the pooling and bilinear interpolation operations. We also input a feature map that is obtained from the feature extractor to the UNET instead of passing a pure image as input. DeepLab [1] was another architecture developed by Google to tackle the same challenge of semantic segmentation. It uses CNN as its primary architecture. Unlike the U-net, which uses features from every convolutional block and then concatenates them with their corresponding deconvolutional block, DeepLab uses features yielded by the last convolutional block before upsampling it, similarly to FCN. It uses a modified version of FCN. The DeepLab applies atrous convolution for upsampling to achieve a denser a denser representation for segmentation tasks. Atrous convolution (also known as dilated convolution) is a type of convolution with defined gaps. The advantage of using a dilated convolution is that it reduces the computational cost significantly. Our method is to pass the feature map and resized image combined as the input to the DeepLab model. We use a total of 3 dilations and one image spatial pooling.

2.2. Depth Estimation

Depth estimation is another per pixel task that determines how far each pixel is from the observer. Traditionally,

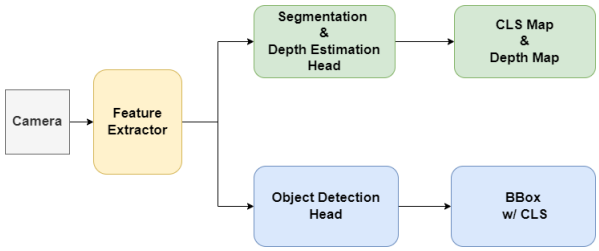

Figure 1. The outer architecture of our model. As shown there is a shared feature extractor and two separate heads for three tasks.

image based depth reconstruction used SLAM approaches. Recent machine learning algorithms have achieved much better results where a CNN was employed to generate a depth map from a single RGB image using supervised learning. Our approach to achieve depth estimation is to use the semantic segmentation model and an extra convolutional layer since the output format for both of them are quite close and they both share the same heuristics.

2.3. Object Detection

The goal of object detection is to recognize instances of a predefined set of object classes and describe the location of each detected object using a bounding box. It consists of two separate tasks namely classification and localization. YOLOv1 [8] is a singe stage object detector, it is a single neural network that predicts class probabilities and bounding boxes directly from the input image. Firstly it divides the input image into a 7x7 grid. If the center of an object falls into a grid cell, the cell is assigned for detecting it.The neural network then predicts the confidence scores, bounding boxes and classes for each cell. The confidence score represents how likely a bounding box contains an object (not a category). Thus, each bounding box has 5 predictions: x, y, w, h and confidence score, where (x, y) are relative to a grid cell and (w, h) is relative to the whole image. So, if a cell does not contain an object, the confidence score should be zero; otherwise, it should be equal to the IOU between the predicted and the ground truth box.A union over intersection operation between the predicted box and the ground truth box is performed to generate the confidence scores. Finally to predict the outputs for the bounding boxes, a 1x1 convolutional layer, hidden dimensions filters, Dropout layers, Leaky ReLu and a 1x1 convolutional layers are used. define confidence score as Probability(Object) * IOU with the ground truth. Additionally, the YOLO predicts C class probabilities per grid cell (regardless of the number of bounding boxes). Therefore, there are $S \times S \times (B \times 5 + C)$ predictions. Our method adopts a simplified version of the YOLOv1 object detection model.

3. Method

3.1. Problem Statement

As mentioned, to enable well-informed navigation in urban scenes, we would like to construct a system based on monocular image stream that is able to (1) detect movable objects, (2) segment out open/drivable spaces, and (3) measure the distance from the camera to the open spaces and objects simultaneously. To resolve these problems with a single deep learning architecture, our design need to perform (1) object detection, (2) semantic segmentation, and (3) depth estimation respectively.

3.2. Design Constraints

The main issue we faced during the design process is lacking computational resources. For the entirety of implementation, training, and evaluation, we need operate only on a single 1050Ti GPU with less than 2GB of CUDA memory. Also for our design to be able to be deployed on edge devices and run with an acceptable speed, we would like the architecture to be as lightweight as possible for both training and inference.

These constraints implies that we needed to restrict the size of our model during both forward and backward pass and the FLOPs needed during inference. To meet such constraints, all parts of our design are significantly reduced in size as opposed to their original counterparts. Also, during training, the batch size, size of dataset, and preprocessing we used were limited, which greatly influenced the final result.

3.3. Train of Thoughts

A natural thought of tackling with this problem is to construct three models to solve the three problems individually. Like use a model proposed by Eigen *et al*. [2] to do depth estimation, a U-Net model [9] to do segmentation, and a YOLOv1 [8] model to do object detection. detection. However, the sort of implementation contradicts with the constraints we were facing, which put restriction on the size of model. Therefore we took a path that enables us to reuse computation during inference as much as possible.

Firstly, we realized that depth estimation and semantic segmentation share close output format and inherent heuristics, so it is possible to implement and train a single neural network to perform both tasks at once. Also it is noticed by us that both process requires the procedure of feature extraction.

Directed by these thoughts, we chose a design as follows. First a shared backbone network will perform feature extraction. Then one downstream network will utilize the feature extracted to perform object detection while the other downstream network will perform segmentation and depth estimation in the same time.

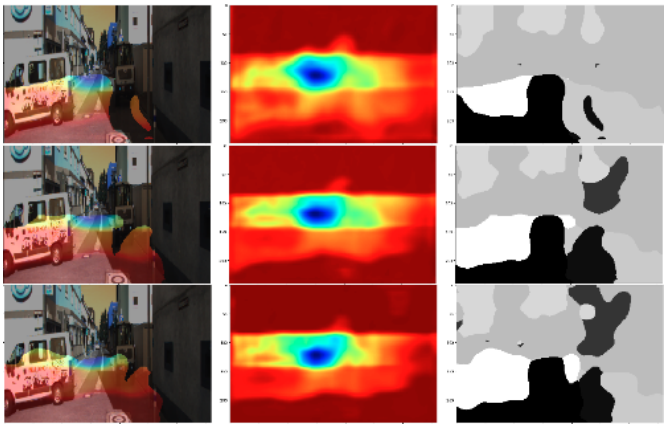

Figure 2. Snapshots of training process for U-Net (from epoch 9, 59, and 109 respectively). The three columns are from left to right are combined effect, depth estimation, and semantic segmentation.

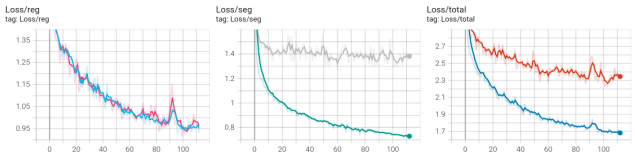

Figure 3. The loss value during the training process.

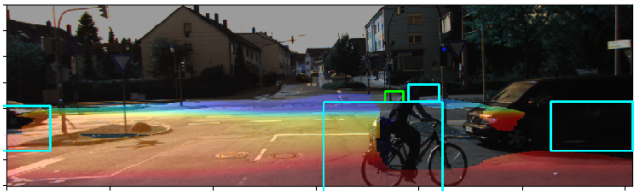

Figure 4. An example of combined effect after training

For the choice of feature extractor, we chose MobileNetV3 [4] pretrained on ImageNet [6] for its optimized size-accuracy balance. During training procedure, we froze the weights of the backbone network and just train the segmentation, depth estimation, and object detection heads.

We provide the details of our implementation of the network for semantic segmentation as well as depth estimation and that of object detection as follows.

3.4. Semantic segmentation & Depth Estimation

We implemented two lightweight architectures for the dual task of semantic segmentation and depth estimation and they showed comparable performance during the experiments. These two architectures was originally both developed solely for semantic segmentation, and we added an extra channel in the final convolution layer to perform dense regression which is the result for depth estimation.

We firstly tried a lightweighted version of U-Net [9]. In the original implementation, there are 5 feature maps generated in both the contracting path and the expanding path. In each path, the numbers of channels follows the sequence of $[64, 128, 256, 512, 1024]$, and the initial input resolution is 572×572. We reduced the computation needed by this architecture significantly by reducing the number of channels in each path to $[64, 128, 128]$ and the input resolution is adjusted to 96×48.

Then as comparison we also implemented a lightweight version of DeepLabV3 [1]. In the original implementation, there was a series of dilated convolution layers prior to ASPP module, and inside the ASPP module there are 4 dilate convolutions each with different dilate rates. In our implementation, the convolution layers prior to the ASPP module was removed and there was only 3 convolution being performed inside the ASPP module.

3.5. Object Detection

We implemented a modified version of YOLOv1 to perform object detection. We used MobileNetV3 instead of the original feature extractor. Furthermore we added Leaky ReLU and BatchNorm inside the architecture to gain better convergence rate.

4. Experiment

4.1. Setup

For experiment setup, we used the aforementioned lightweight version of U-Net and YOLOv1. We trained them separately. During training each of the two networks has their own frozen backbone network, a MobileNetV3, and before inference we simply delete their backbone networks and instantiate a new one for them to share.

4.2. Dataset

For all of depth estimation, semantic segmentation, and object detection, we used KITTI dataset [3], inside which all images are of size $1216 \times 352 \times 3$, where 3 denotes the channel number. To reduce the computational resources required we resized the inputs to $\frac{1}{3}$ of its original size. Additionally, in order to increase the robustness of the model we used various preprocessing of images such as random horizontal flip, color adjusting, and solarization.

4.3. Loss Design

For YOLO detector naturally YOLO loss is used which is a combination of bbox regression loss, classification loss, and confidence loss. For semantic segmentation, we used weighted Cross Entropy loss which gives objects and lanes more weight during training. For depth estimation, we used a combination of L2 loss, SSIM loss [7], and depth smoothness loss [10].

4.4. Training & Evaluation

We trained both networks for 109 epochs (as shown in 2 and 3). For U-Net we used an Adam [5] optimizer with an initial learning rate of 0.0005, and for YOLO we used a SGD optimizer [6] with learning rate of 0.0001. None of the optimizer employed weight decay or momentum since these additions may change the weights of the backbone network even when its weights are frozen.

The final combined effect (as shown in 4) is not quite ideal compared to what we had in mind, especially due to lack of training epochs, the accuracy for object detection and semantic segmentation is lower than expectation. But thanks to the effort in reducing the size of models, our model achieves inference speed of 70 FPS on Nvidia 1050Ti GPU and 0.3 FPS on Intel i7-7th gen CPU, and the speed is expected to be enhanced by possible quantization.

5. Conclusion

In conclusion, we implemented a deep learning-based system that is able to perform object detection, semantic segmentation, and depth estimation simultaneously. We proposed a model that use one neural network to accomplish object detection and a separate one to accomplish semantic segmentation and depth estimation. We let them share the same feature extractor to achieve desirable speed in inference.

As for future perspective, the priority should be further experimenting on hyperparameter tuning and architecture designs that could increase the inference accuracy of the model. Also quantizing the model to lower precision (16 bit or 8 bit) has the potential to further decrease the size of the model.

References

[1] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[2] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network, 2014.

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[4] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. Searching for mobilenetv3, 2019.

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[7] J. Nilsson and T. Akenine-Möller. Understanding ssim, 2020.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, 2016.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[10] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction, 2018.