A Generic Approach to Predict Bird's-Eye-View from Multi-View Scenes through Self-Supervised Monocular Depth Estimation

Alfred Ajay Aureate Rajakumar   Mohith Damarapati,   Vikas Patidar

Abstract

This work talks about an easy and effective approach to address a very challenging and interesting task of road layout estimation in complex driving environment. From six camera images encompassing the whole 360° view, we try to predict the bird's-eye-view of the road and surrounding objects of the ego car. We present an effective generic approach to handle both the tasks by reducing each of them to an Instance segmentation problem. Further by leveraging the unlabeled dataset and data augmentation techniques, we estimate depth in an unsupervised manner. Finally using pretrained depth and novel architectures, we accurately generate bird's-eye-view of a scene.

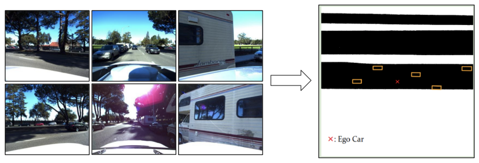


Figure 1. Problem statement: Predicting bird's-eye-view from multi-view scenes

1. Introduction

Generating bird's-eye-view from multi-view scenes is a crucial task for other downstream control operations in a fully self-driving environment. Given multiple camera views obtained from an ego car, we focus on generating the road map and detecting objects in the bird's-eye-view (as shown in Fig. 1). Also, we approach this problem as a self-supervised learning task due to the limited availability of the labeled dataset.

We used the temporally spaced unlabeled data to obtain 3D depth information for each of the images as described in (Godard et al., 2018)

We then used these pre-trained monocular depth encoders (MDEs) to create depth maps for labeled data (on the run). This depth is combined with its corresponding (RGB) image for 6 different camera views and trained for both roadmap and object detection tasks separately [1].

2. Related Work

The paper (Zhu et al., 2018) discusses generating bird's-eye-view from multi-view images using an intermediate homography view estimate which is then corrected using GANs. But, it used a lot of labeled data for training the generators and the discriminators.

In (Roddick & Cipolla, 2020), authors discuss about generating bird's-eye-view from multi-view scenes for obtaining the road maps and detecting moving objects surrounding the ego car using just an end-to-end deep learning architecture. They achieve this using a pyramid occupancy network consisting of 4 parts, including a dense transformer layer where the image features are transformed into bird's-eye-view. But it also required a lot of labeled data. This work also suggests that depth information could be used as an alternative approach for solving the same problem.

Acquiring per-pixel ground truth depth data information is an arduous task. Self-supervised learning is a promising alternative to learn depth directly from the images without the labeled data.(Godard et al., 2018) describes obtaining depth in a self-supervised manner based on adjacent frames in time.

3. Methodology

3.1. Supervised Training and Backbone Architecture

We pose both road map prediction and object detection tasks as instance segmentation problems. We used the same encoder-decoder architecture for both the tasks which is inspired from UNet. Our initial experiments suggested that feeding six different views through siamese encoders worked better than multiple view-specific encoders. Stacked siamese encoder representations are then passed through a decoder to generate the road mask(for road map prediction task) and object mask (for object detection task) (as shown in Fig. 2).

For object detection, we find all the contours with label 1 from the object map. For each such contour, we determine

[1] https://github.com/Mohith22/autonomous-driving

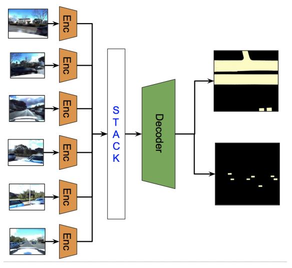


Figure 2. Backbone architecture

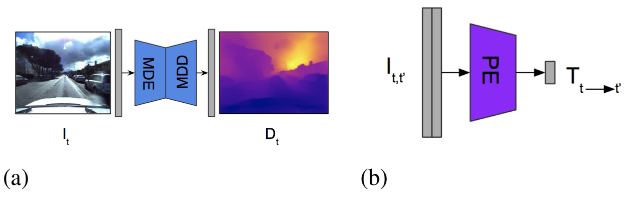


Figure 3. Depth estimation networks. (a) Depth network (b) Pose network

the tightest bounding boxes around it with minimum area. This way we could generate bounding boxes at any angle without explicitly predicting the angle.

3.2. Self-Supervised Training

3.2.1. Image-level pre-text tasks

- Our simple ConvNet designed to do six-class classification to learn which camera view the given image belongs to, obtained an accuracy of 98 % in just two epochs. This result led us to conclude that image-level pretext tasks like jigsaw, flipping, cropping, rotating etc. might not work because they are not hard enough to learn representations that can help to solve our task.

- Another pre-text task we implemented was SimCLR(4x)-based (Chen et al., 2020). We basically took an image from a particular view and augmented it with two different instances t' and t'' of the random transforms mentioned in SimCLR (resize, crop, horizontal flip, color jittering, grayscale and gaussian blur). The transformed images were passed through our UNet inspired RGB encoder with additional layers including MLP layers that convert the image-level features to 256 output channels. Using contrastive learning, we reduce the loss between the output features of t' and t'', whereas maximizing the loss between features from different samples.

3.2.2. Depth estimation

From the depth maps, we can obtain important information to solve our task like the distance of the ego car from other cars, difference between roads and cars, and about other movable and immovable objects.

Consecutive temporal frames provide the supervision signal for estimating depth. We frame the learning problem as one of novel view-synthesis by training our model to predict a target frame from the viewpoint of its nearby frames. Depth here acts as an intermediate variable which needs to be learned to perform better view-synthesis.

Our ideas are inspired from (Godard et al., 2018) where we formulate depth estimation as the minimization of a photo-metric re-projection error at training time. We have two networks namely depth network and pose network (as shown in figures 3 (a) and (b)). The goal of the pose network is to estimate the relative pose ($T_{tt'}$) of adjacent frames ($I_{tt'}$) with respect to the target frame (I_t). We specifically consider next (I_{t+1}) and the previous (I_{t-1}) frames to estimate the relative pose. After estimating the relative pose, we use that relative pose information of nearby frames, depth of the target frame and camera intrinsic parameters to estimate the appearance of the target view. We then use STN (Jaderberg et al., 2015) to sample the source images and SSIM (Wang et al., 2004) to get our photo-metric re-projection error.

We also used other techniques like per-pixel minimum re-projection loss and auto-masking stationary pixels stated in (Godard et al., 2018) to improve the self-supervised depth estimation

3.3. Downstream Prediction with Depth Information

Foe our downstream tasks of Roadmap Prediction and Object Detection, we incorporate pre-trained depth network into our encoder-decoder pipeline.

3.3.1. RGB-D Input

Instead of RGB, we pass RGB-D inputs to the Siamese encoders and follow the same pipeline(3.1).

3.3.2. Separate RGB and Depth Encoders

In the last experiment, we didn't get much of the improvement because depth information was not being used properly. We added separate depth encoders(with shared weights) in order to encode depth-related features separately and then stacked them together with the encoded RGB features followed by the decoder (refer Fig. 3).

3.3.3. Orient Net

We added a new module between the siamese encoder and decoder, which we call "Orient Net", consisting of few Convolutional and Batch Normalization layers (refer Fig. 5). In this setting encoded RGB and depth features for a camera-view are stacked together and passed through this module.

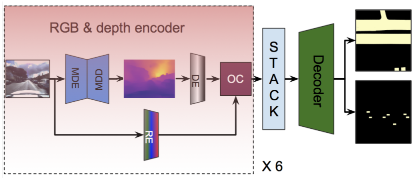


Figure 4. Depth based architectures. **DE**: UNet inspired Depth Encoder; **RE**: UNet inspired RGB Encoder; **Decoder**: UNet inspired Decoder; **OC**: Orient Net; **MDE**: Monocular Depth Encoder; **MDD**: Monocular Depth Decoder

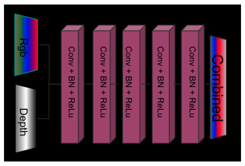


Figure 5. Orient Net: Coupling RGB features with depth features

The motivation behind this was to generate the bird-eye view in a hierarchical manner - first build RGB and depth features independently, then couple view specific features through Orient Net and dedicate few layers to learn the coupling and then all the output feature maps are combined and passed through a decoder to project the features on the bird's-eye-view. We observe that this setting works better especially for Object detection instead of stacking all 12 feature maps (6 RGB and 6 depth) as it makes the projection task easier with far less number of parameters.

3.3.4. Double Decoder

Instead of passing depth information as an input we try to reconstruct the depth maps from the input image and then have two decoders - one for the downstream tasks and one for the depth reconstruction. In this way we encode richer representations and force the network to make use of depth for the downstream tasks. (refer Fig. 6 (a)).

$Cost = C_{task} + \lambda C_{Depth}$

3.3.5. Triple Decoder

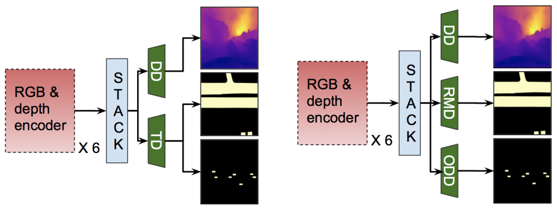
This approach builds on the top of the Double decoder approach. Here we try to perform both the downstream tasks together along with Depth reconstruction. (refer Fig. 6 (b)).

$Cost = C_{Roadmap} + C_{ObjDet} + \lambda C_{Depth}$

For both Double decoder and Triple decoder, we use λ as 0.1224 to normalize the output space(800 x 800 for roadmap and object detection and 256 x 306 for depth map).

3.3.6. Loss functions

For Road map prediction, we use standard Binary Cross Entropy loss as the number of road and non road pixels are



(a) (b)

Figure 6. Multiple decoder architectures. (a) Double decoder (b) Triple decoder. DD: Depth Decoder; TD: Task (dependent) Decoder - could be one of RMD or ODD depending on the task; RMD: Road Map Decoder; ODD: Object Detection Decoder. All the decoders are UNet inspired networks

comparable. Whereas in object detection the background pixels are in abundance, which creates class imbalance(0: background, 1:object). To solve this problem, we use Dice loss, which handles this very well. For object detection, Dice Loss worked much better than BCE loss.

$$DiceLoss(p, \hat{p}) = 1 - \frac{2\sum(p_{h,w}\hat{p}_{h,w}) + 1}{\sum(p_{h,w} + \hat{p}_{h,w}) + 1}$$

3.3.7. Data Augmentation

We used color jittering data augmentation techniques like brightness, contrast, saturation and hue to enhance both our labeled and unlabeled datasets. The effect of data augmentation can be seen in tables 1 and 3.

4. Results

We conducted a lot of experiments on multiple architectures and based on multiple ways of stacking the view-representations. But, we only report the results for our best model (i.e. Siamese encode-decoder) for both road map (refer Fig. 7 (a)) and object detection tasks (refer Fig. 7 (b)) with and without depth information. We took three scenes (i.e. 378 samples) from the dataset as our validation set. We trained all models with both BCE and dice loss. Data augmentation is also investigated for the best models. Validation scores for models without depth (only supervision) are shown in table 1 and with depth is shown in Table 2. Best models in Table 2 are investigated with data augmentation and the results are shown in Table 3. These results are based on self-supervised depth estimation model trained only for 5 epochs. We expect to have 8-10 % and 3-4 % improvement in road map and object detection results respectively when depth estimation model is trained for more than 30 epochs.

5. Conclusion

We found that learning projections to map from camera view to bird-eye view is impossible through simple image level pretext tasks like 6-class classification and Jigsaw as

Model	Roadmap		Obj. Det.	
(Loss)	Dice	BCE	Dice	BCE
Siamese (no data aug.)	72.92%	73.16%	0.72%	0.59%
Siamese (data aug.)	73.32%	73.68%	0.79%	0.58%

Table 1. Validation set performance - without depth (supervised)

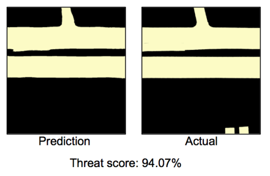
Model	Roadmap		Obj. Det.	
(Loss)	Dice	BCE	Dice	BCE
With Orient Net	77.45%	78.32%	1.16%	0.98%
No Orient Net	77.13%	79.36%	1.08%	0.84%
Double decoder	76.82%	77.24%	0.75%	-
Triple decoder	76.42%	76.84%	0.71%	-

Table 2. Validation set performance - with depth (self-supervised)

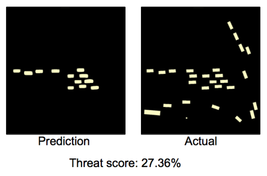
they are way too easy. Also, validation accuracy for both the tasks got improved significantly after incorporating data augmentation and depth information, which implies that depth is indeed very critical for this task as we anticipated. In object detection, we made one important observation that many predicted bounding boxes were falling in IOU range of 0.4-0.5, resulting in an overall IOU of 0 with threshold as 0.5. So with a little more training(currently trained for 5 epochs only), results can be improved by 3-4%.

6. Future work

There are many potential ideas, that we thought might work, but didn't get a chance to work on them. Unsupervised depth predictions for Occlusions can be improved by injecting random noise on the super-pixel level instead of a rectangular box. We can take RGB encoder features from SimCLR pretraining and depth encoder features from mono



(a)



(b)

Figure 7. Results: (a) Roadmap prediction (b) Object detection

Data Aug.	Roadmap Prediction	Object Detection
Yes	**81.66%**	**1.23%**
No	79.36%	1.16%

Table 3. Validation set performance - best models. For roadmap prediction, BCE loss was used without Orient Net. Whereas for object detection task, Orient Net was used with Dice loss.

depth pretraining, freeze these networks and then fine-tune the decoder. This will help a lot as we will already have well built features for both these encoders and very less number of parameters to train. One idea that we really want to try is unsupervised learning of optical flow for object detection task similar to Selflow (Liu et al., 2019). As optical flow maps are very good in defining object boundaries, we can use optical flow and depth maps together for object detection, which should significantly improve the results. Unsupervised homography matrix estimation could also be explored to directly convert the camera-view images to top-down view. When linked with GAN, this could improve object detection significantly.

References

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations, 2020.

Godard, C., Aodha, O. M., Firman, M., and Brostow, G. Digging into self-supervised monocular depth estimation, 2018.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. Spatial transformer networks, 2015.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Liu, P., Lyu, M., King, I., and Xu, J. Selflow: Self-supervised learning of optical flow, 2019.

Roddick, T. and Cipolla, R. Predicting semantic map representations from images using pyramid occupancy networks, 2020.

Wang, Z., Bovik, A. C., Sheikh, H. R., Member, S., Simoncelli, E. P., and Member, S. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004.

Zhu, X., Yin, Z., Shi, J., Li, H., and Lin, D. Generative adversarial frontal view to bird view synthesis, 2018.