Mapping the Bird's Eye View from the Ground: Binary Road Map and Object Detection with a Self-Supervised Depth Model

Justin Mae Brina Seidel Derek Yen

Abstract

We use depth estimation generated by self supervised learning to apply existing approaches to binary road map classification and object detection to a novel bird's eye view (BEV) generation task. We find that the architectures of such large models are flexible enough to learn the bird's eye road map when given depth information and sufficient training data. This approach is less successful at detecting objects from above. Our best road map model achieves a validation threat score of 0.7143, and our best bounding box model achieves a validation threat score of 0.0069.

1. Introduction

In this paper, we explore two tasks – a segmentation task and an object detection task – that use images taken from a car on the ground to predict a bird's eye view (BEV) of the surroundings. Transitioning from a ground view to a BEV requires an implicit understanding of where each pixel of the input image resides in three-dimensional space. To address this issue, we provide depth information to existing models that perform well on traditional two-dimensional segmentation and object detection tasks.

To predict a BEV binary road map from ground-level images, we use a modified version of HRNet (Wang et al., 2019a). To predict BEV bounding boxes for nearby vehicles from ground-level images, we use a modified version of YOLO v3 (Redmon & Farhadi, 2018). We provide both of these models with depth maps created by a version of Monodepth2 (Godard et al., 2018), a self-supervised depth model that we retrain on our labeled and unlabeled dataset.

We find that the architectures of such large models are flexible enough to learn the bird's eye road map when given depth information and sufficient training data. This approach is less successful at detecting objects from above. Our best road map model achieves a validation threat score of 0.7143, and our best bounding box model achieves a validation threat score of 0.0069.

2. Literature Review

2.1. Image Segmentation

Creating a binary road map is a variant of the classic image segmentation problem of dividing a picture into semantic regions. While early solutions to this problem used thresholding and edge detection, state-of-the-art methods today use deep convolutional neural networks to discriminate between regions. For example, Mask R-CNN adds a branch to the Faster R-CNN architecture to output a segmentation map (He et al., 2017), and DeepLab uses a novel method of up-sampling the low-resolution layers to segment an image (Chen et al., 2016).

Many such models drastically reduce the resolution of the input image, then increase it again before making a prediction. HRNet departs from this by maintaining a high-resolution representation of the input throughout the network while also learning from lower-resolution representations in separate branches (Wang et al., 2019a). We incorporate its architecture into our road map model for this reason.

2.2. Object Detection

2D Object Detection. Though BEV object detection from ground-level images is not a classic 2D problem, we do ultimately produce 2D bounding boxes, and thus 2D object detection methods can be useful.

Single-stage object detectors like Retinanet and YOLO have become increasingly popular for their processing speed. Retinanet corrects for the fact that a large portion of a typical image is easily classified as background by using focal loss, which means that the model is penalized more for hard examples (Lin et al., 2017). YOLO divides the image into a grid, then predicts a fixed number of bounding boxes with centroids in each grid region as well as its confidence about those boxes (Redmon et al., 2015), and variations of YOLO incorporate additional methodologies like generating feature maps at different scales, which improve its accuracy (Redmon & Farhadi, 2018).

3D Object Detection. Much of the literature on 3D object detection is based on LiDAR point cloud data, and predicting 3D bounding boxes from monocular images remains

an area of active research. Such 3D approaches would be directly relevant to our problem since it is possible to take the top corners of a 3D bounding box as the BEV box. For example, Roddick et al. (2018) propose an orthographic feature transformation which maps ground-level image features into BEV features using camera properties and then computes bounding boxes with those top-down features.

In contrast, Wang et al. (2019b) explicitly estimate depth, then use that estimate and camera properties to project pixels themselves into 3D coordinates, which they refer to as pseudo-LiDAR. Existing 3D object detectors for point cloud data, such as the frustum PointNet (Qi et al., 2017) or AVOD (Ku et al., 2017) can then be used.

3. Models

Our proposed idea is to use depth estimation generated by self supervised learning to improve on existing approaches to binary road map classification and object detection. Our hypothesis is that additional depth information would help the models learn 3D aspects of the image in order to learn the mapping from ground to BEV.

3.1. Binary Road Map

Our road map model uses a modified version of HRNet that accepts six images instead of one.

We experiment with two ways to make the model accept six input images. First, we tried simply stacking the six images on top of each other, providing the model with 18 RGB input channels (*stacked input*). However, this method obscures the fact that corresponding pixels in different channels may refer to wildly different locations in physical space if those channels came from different input images. Second, we tried passing the six input images to the model separately, performing a convolution, then concatenating the resulting representations (*tiled input*). The representations were tiled according to the relative spatial positions of the original images (front left, front center, etc.). While avoiding the drawbacks of the stacked input approach, this method falsely suggests that the six input images represent locations that are next to each other, rather than rotated views from the same origin point.

We also modified the model to use only two output classes (road and not-road), and to produce an output of 800x800 pixels.

Model weights were initialized using a Gaussian normal distribution and then trained from scratch on our data. All road map models were trained to minimize the binary cross entropy loss between the prediction and the target using stochastic gradient descent with momentum.

3.2. Object Detection

We used YOLO v3 to learn to detect objects from a BEV. Like HRNet, this model was modified to take six stacked images as inputs for a total of 18 RGB channels. The model was trained using labeled data to predict bounding boxes without classification information. Because YOLO v3 expects bounding boxes to be aligned with the axes of the image, we altered the labels on our training data to create the smallest possible box aligned with the image's axes that encloses the true original bounding box.

YOLO v3 uses a multi-part loss that penalizes both classification error and bounding box coordinate error. Our model only includes the bounding box coordinate error because we modified the model to consider all predictions as a single class. The bounding box coordinate error only penalizes the predictor that is responsible for the ground truth box or the highest IOU of any predictor in that grid cell.

The model outputs bounding box coordinates to which we apply non-maximum suppression. We used a confidence threshold of 0.06 and, because cars cannot (safely) overlap from an overhead perspective, an IOU threshold of zero.

Model weights were initialized using PyTorch default values, and models were trained with the Adam optimizer.

3.3. Depth Map

Translating from the ego car's perspective to a BEV requires implicit knowledge of each point's position in 3D space. For this reason, we wanted to model the depth of each point in the input images.

Since we do not have ground-truth information on object depth, we considered self-supervised approaches to depth estimation. In particular, we tried Monodepth2, proposed by Godard et al. (2018), which reconstructs an input image using only the depth estimation of the input image from a depth network and two temporally adjacent frames (in the case of our problem, the prior and later samples from a given scene). By training with a per-pixel minimum reconstruction loss and using depth projections at different scales, the depth network is able to learn meaningful features. Furthermore, since this method is self-supervised, we were able to train this model from scratch using images from both the labeled and unlabeled scenes and all six camera angles. We used all images from 120 scenes as training, with the remaining 14 scenes as validation to determine the best model. Model weights were initialized using default PyTorch values without downloading any pretrained weights.

Since the model requires input to be a factor of 32 for scaling purposes, we reshape each input image to have a width of 288 pixels and a height of 256 pixels using bicubic interpolation. Resulting depth map outputs then get reshaped

again to the original dimensions.

We were then able to add the depth output of this model as an additional input channel to both our road map model and our object detection model with a PyTorch transformation class, which evaluates our trained depth network to estimate depth from a given image at test time. An example output is shown in Figure 1. While some of the nearby objects get encoded, such as the cars to the right of the ego car, the depth model has more difficulty with the sky and clouds. From observation, it is possible that the depth map is encoding features related to color as well.

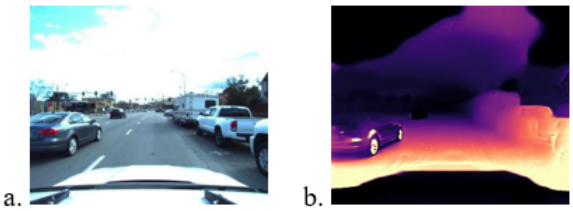

Figure 1. (a) original input image (b) depth map estimate

4. Experiments

All models were trained on 2,646 samples from 21 scenes, with 7 additional scenes serving as a validation set. We performed the train/val split by scene to ensure that our model would generalize to unseen settings.

4.1. Binary Road Map

We varied the learning rate and input method when training our road map model. We also trained versions of the model with and without a depth channel produced by the self-supervised model as input. Table 1 displays the results of these experiments.[1]

We find that adding a depth channel usually improves the model performance. We achieve the best performance on the validation set using a learning rate of 10e-4, tiled input, and depth information.

4.2. Object Detection

We varied the initial learning learning rate and IOU threshold when training our object detection model. All models were trained using the stacked input method. The results are displayed in Table 2.

We see consistently better performance using an IOU threshold of 0.3 during training. Unlike the road map model, the object detection model does not seem to consistently benefit from the addition of a depth channel as input. It's possible that this is because we used the stacked input approach when adding the depth channel to this model, whereas we used the tiled input approach when adding the depth channel to our road map model.

The best RGBD input uses a learning rate of 10e-6 and an IOU threshold of 0.3. It achieves a validation loss of 3.4588 and a validation threat score of 0.0069.[2]

Table 1. Validation Threat Scores for Road Map Model Specifications

Learning Rate	Input Type	RGB Input	RGBD Input
10e-4	stacked	0.6924	-
10e-4	tiled	0.6909	**0.7143**
10e-5	stacked	0.7001	-
10e-5	tiled	0.7083	0.7120
10e-6	stacked	0.6925	-
10e-6	tiled	0.6780	0.6649

Table 2. Validation Loss for Bounding Box Model Specifications

Initial Learning Rate	IOU Threshold	RGB Input	RGBD Input
10e-4	0.2	4.9171	-
10e-4	0.3	3.5015	3.4783
10e-5	0.2	6.7592	-
10e-5	0.3	**3.4370**	3.5065
10e-6	0.2	5.5398	-
10e-6	0.3	3.4766	**3.4588**

5. Error Analysis

5.1. Binary Road Map

Perhaps unsurprisingly, our road map model is best at mapping the road that the ego car is driving on, especially when that road is straight (Figure 2). It performs worse when mapping side roads. We attribute this to the fact that side roads are often obstructed by trees, buildings, etc. in the input images (Figure 3).

We also find that our model has trouble with instances in which the ego car is not parallel to the road. As shown in Figure 4, the model struggles to map roads that are not aligned with the axes of the image.

5.2. Object Detection

Our object detection model has difficultly learning to map BEV from the ground (Figure 5). A limitation of YOLO v3 is that bounding box outputs are aligned to edges of the image, whereas in the dataset bounding boxes can be rotated. Another limitation of YOLO v3 is its difficulty detecting

[1] We used the validation threat score as our evaluation metric to determine the best model. This is because, during training, the validation threat score continued to improve long after the validation loss stopped decreasing.

[2] One RGB model slightly outperformed this one, but it did not run in time for the submission deadline.

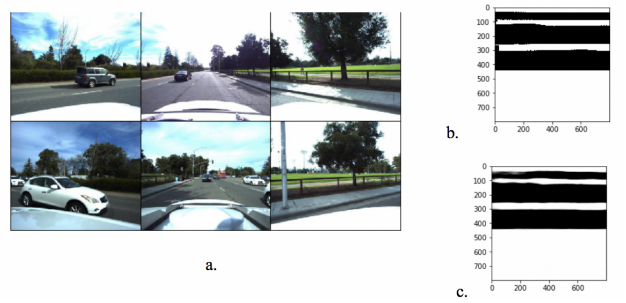

Figure 2. (a) Input images, (b) true road map, (c) predicted road map.

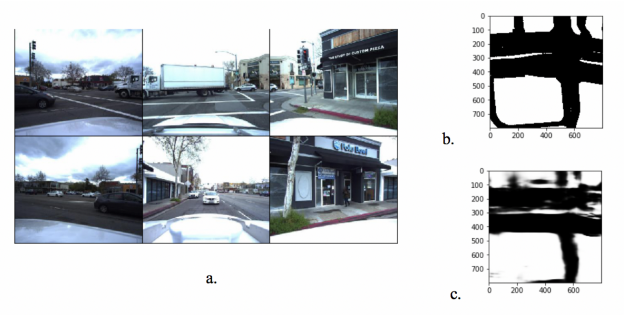

Figure 3. (a) Input images, (b) true road map, (c) predicted road map. The input images do not contain a clear view of the side roads that are behind the car, shown on the left of the true road map. These side roads are missing from our model's prediction.

small objects, such as pedestrians and bicycles.

6. Conclusion

We generate depth maps for each image using a self-supervised approach. We find that augmenting the data with depth estimation improves performance for the binary road map problem, which suggests that depth data contains useful three-dimensional information about the scene. As for the object detection problem, we found that a two-dimensional model had difficulty learning the BEV representation, even with the inclusion of depth information. Further research is needed to understand how 2D object detection models can be applied to BEV tasks.

References

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2016.

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

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn, 2017.

Ku, J., Mozifian, M., Lee, J., Harakeh, A., and Waslander, S. L. Joint 3d proposal generation and object detection from view aggregation. *CoRR*, abs/1712.02294, 2017.

Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. Frustum pointnets for 3d object detection from RGB-D data. *CoRR*, abs/1711.08488, 2017.

Redmon, J. and Farhadi, A. Yolov3: An incremental improvement, 2018.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection, 2015.

Roddick, T., Kendall, A., and Cipolla, R. Orthographic feature transform for monocular 3d object detection. *CoRR*, abs/1811.08188, 2018.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., and Xiao, B. Deep high-resolution representation learning for visual recognition, 2019a.

Wang, Y., Chao, W.-L., Garg, D., Hariharan, B., Campbell, M., and Weinberger, K. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019b.

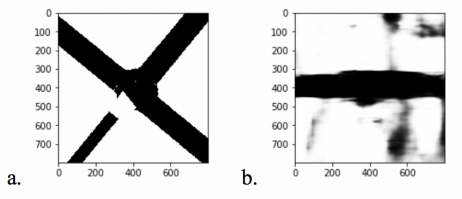

Figure 4. (a) True road map, (b) predicted road map. The model clearly expects there to be a main road that is parallel with the x-axis.

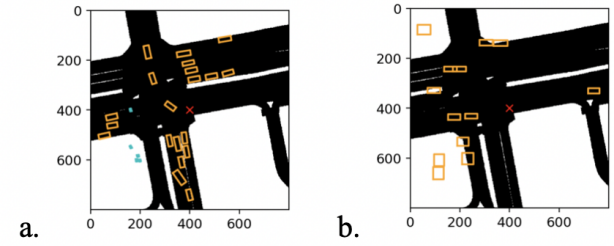

Figure 5. Object Detection (a) true road map with bounding boxes for cars and pedestrians (b) YOLO v3 cannot output rotated bounding boxes and cannot detect small objects, i.e., pedestrians