Autonomous Navigation: Road Map Generation and Object Detection

Alexander Gao (awg297), Yash Deshpande (yd1282), and Yunya Wang (yw4509)

Abstract: The task of understanding one's surroundings is useful in the context of developing autonomous vehicles that can safely navigate through unpredictable environments, using only limited sensory input. Toward this objective, we propose a method for generating bird's-eye-view road maps and object maps from six monocular perspective views from a moving vehicle.

I. INTRODUCTION

The dataset consists of 133 scenes, 106 of which are unlabeled, and 27 are labeled. Each scene contains 126 samples, with a sample being defined as a group of images from 6 cameras that have been mounted to a running car. The labels consist of ground truth object bounding boxes and a bird's eye binary road map. Our goal is to learn to accurately predict road map and object bounding boxes from the perspective view of a sample.

A note about the organization of this paper: as there were multiple methods we explored in earnest that did not directly make it into our competition submission, but that we feel still merit analysis, we devote a section towards the end of this paper to that work: "FAILURE: AN OPPORTUNITY TO LEARN". We eschew giving background for those methods/approaches in the main body of this paper, in order to maximize the flow and readability of the paper as a whole.

II. RELATED WORK

II.A. Generative Networks

Generative adversarial networks[1] are able to learn to generate images that are indistinguishable from a source distribution, by iteratively training a discriminator network to differentiate between real and generated images, and training a generator network to fool the discriminator into classifying false positives. Since Goodfellow et al. (2014), many improvements and variations have been proposed, such as Deep Convolutional GANs[2] which impose specific architectural constraints to stabilize training and reduce noise. There have also been explorations of GANs whose output can be explicitly directed by some input or latent variable, such as Conditional Generative Adversarial Nets[3], and Artistic Style Transfer[4], which introduces loss terms to explicitly force a generated image to match the low level features of one image and the high-level perceptual features of a another image. We found the work on Image-To-Image Translation[5]) by Isola et al. (2018) provided a strong reference – most notably that the authors are fundamentally interested in mapping an image to another image directly, based on features of the original image. Their design uses a "U-Net" architecture for their generator, as well as a PatchGAN loss. Notably, they also train their generator using dropout, and continue to use dropout at *test time* for stochasticity. This differs from the traditional approach of using random noise as the input to the GAN. In the next section, we discuss our own network design which was inspired by theirs, and discuss domain-specific modifications we made.

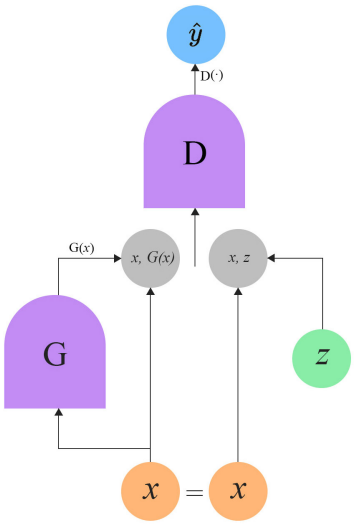

FIG. 1 Architecture of our conditional GAN.

III. OUR APPROACH

III.A. Conditional Generative Adversarial Network

An architectural diagram of our conditional GAN is presented in Fig. 1. A key component of the conditional design of the network is to pass the input (x) to both the generator **and** the discriminator. (In our case, we have chosen to represent the six 3-channel perspective images as a concatenated 18-channel vector.) The discriminator receives a tuple/concatenation of the input, plus a real or generated map. Thus, the discriminator learns to classify *pairs* of {input, map}, rather than maps alone. This is a key component of the conditional network design.

In Isola et al.[5], they use a U-Net architecture for the generator, which is particularly well fit to their application, as they are mostly interested in mapping images to images that share a tightly correlated compositionality, e.g. a sketch outline of a handbag to a fully-rendered handbag. Our application is different in the sense that our input and output do not directly share any compositionality: We are mapping from a high-dimensional RGB perspective image of a scene to a single-channel binary map with an entirely different vantage point. The images do not share much, if any, low level form. Therefore, the symmetrical skip connections that U-Net benefits from do not seem to offer much conceptual value. Thus we remove those skip connections, but maintain the series of residual blocks in the middle of the network, which allow us to build a deep network without worrying about the vanishing gradient problem. Our generator network design is visualized in Fig. 2. All layers except the last use batch normalization[6] and ReLU activations, while the last layer uses a TanH activation. The

generator effectively encodes a (18×256×256) input sample into a 256 × 64 × 64 representation, before being decoded to into a single-channel 256 × 256 binary object mask.

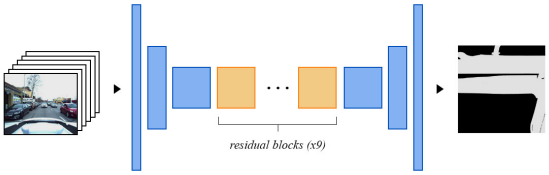

FIG. 2 Our generator network consists of 3 convolutional layers, followed by a series of residual convolutional blocks, and finally 3 transposed convolutional layers.

The discriminator is also fully-convolutional, with 5 layers and no residual blocks. It maps a 19-channel concatenated input/map pair (19 × 256 × 256) to a single-channel "PatchGAN" output (1×30×30). The idea of PatchGAN is that rather than classifying the entire input as real or fake, it has the effect of classifying overlapping patches of the input as such. In theory, during generator training time, this affords the discriminator greater degrees of freedom in backpropagating losses to the generator. It can also be thought of as a form of loss based on high-level perceptual features present in real map/input pairs.

III.A.1. Generating Road and Object Maps

We use the same Generator network to produce both dense binary maps for both road and objects. However, we train the entire network for each task separately (the tasks do not share the same weights or training cycles).

For both road and object maps, the generator output is (1 × 256 × 256) and we perform a simple Bilinear-interpolated upscaling to the final desired (800 × 00) binary map.

We also consider pre-training the generator part of the network on the unlabeled dataset, using image reconstruction `MSE` loss as the objective, in a self-supervised autoencoder fashion, though test results showed this not to help empirically.

III.A.2. Object Contour detection

Once we have the object map, we proceed by finding contours in the smoothed image, making use of OpenCV's `findContours()` function. This derives contours from binary images using the topological algorithm described by Suzuki and Abe [7].

This gives us the pixel co-ordinates of the contour lines for each object. From this, we extract rectangular boxes corresponding to these contours, and obtain box corner points. These are then re-ordered such that they are consistent with the ground truth bounding boxes.

III.B. Loss Functions

We use the following loss functions for the Discriminator network:

$$L_D = \frac{1}{\mathrm{N}} \sum_{n=1}^{N} \ell_{BCE_{D_n}} \tag{1}$$

$$\ell_{BCE_D} = -\frac{1}{\mathrm{P}} \sum_{p=1}^{P} \Big[y_p \log(\mathrm{D}(x,z)_p) + (1-y_p)\log(1-\mathrm{D}(x,\mathrm{G}(x))_p) \Big] \tag{2}$$

where y refers to the target "real - 1" or "fake - 0" label, x refers to the concatenated 18-channel input sample, z refers to the ground truth binary road/object map, $\mathrm{G}(x)$ refers to the generated binary road/object map, N refers to a mini batch of samples, and P refers to the output broadcast shape of the patch GAN target.

The Generator loss functions are as follows:

$$L_G = \frac{1}{\mathrm{N}} \sum_{n=1}^{N} \Big[\ell_{GAN_{G_n}} + \lambda \ell_{L1_{G_n}} \Big] \tag{3}$$

$$\ell_{GAN_G} = -\frac{1}{\mathrm{P}} \sum_{p=1}^{P} \Big[\log(\mathrm{D}(x, G(x))_p) \Big] \tag{4}$$

$$\ell_{L1_G} = \frac{1}{\mathrm{M}} \sum_{m=1}^{M} \|z_m - G(x)_m\|_1 \tag{5}$$

Where M refers to the shape of the road/object maps. For training the Generator, we add an additional L1 loss term, described in Eq. (5). This L1 term has the effect of enforcing the generated images to tend toward the ground truth directly, and leads to more plausible images being output much earlier during training.

IV. EXPERIMENTS

IV.A. Generated Images

We trained our conditional GAN for the road map prediction task for 100 epochs, with Learning Rates for the Generator and Discriminator of 5e-5 and 2e-4 respectively. We experimented with λ values of 1, 10, and 100 for weighting the L1 loss term. With $\lambda = 1$, the effect of the L1 term was greatly diminished, and the resulting generated images were less well-defined, blurrier, and looked less plausible as realistic road maps. The difference between λ value of 10 versus 100 did not seem to be as noticable. Therefore, we went with $\lambda = 100$ to mimic Isola et al.'s setting for this hyperparameter. We used an ADAM optimizer for both Generator and Discriminator, with beta parameters of (0.5, 0.999).

Fig. 3 shows side-by-side comparisons between ground truth and generated maps. The generator's accuracy does not seem to be even across the entire map. We notice that horizontal roads are predicted more reliably than diagonal roads (when the car is presumably mid-turn), and peripheral or intersecting roads are registered albeit imprecisely in the best case, and completely missed in the worst case. We attribute these phenomena to the distribution of the training data manifold: the more commonly certain features appear across many samples, such as horizontal roads,

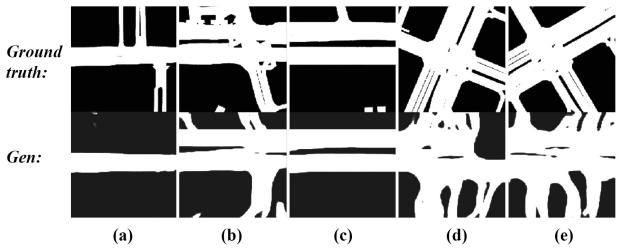

FIG. 3 Test results comparing generated images to ground truth.

the better the generator is able to accurately produce new samples that contain similar features. Our generated maps achieve an average threat score (average of intersection-over-union with thresholds of 0.5, 0.7, and 0.9) of 0.5055.

IV.B. Object Detection

The generated object masks appear plausible, and significantly diverse. However, we noticed that these are subject to pixel noise (Fig. 4(a)). We apply simple smoothing to the masks using the following techniques. **Dilation**: adds pixels at object boundaries, making objects more visible and filling in small holes. **Erosion**: removes pixels at object boundaries, eliminating islands and small objects so that only substantive objects remain. For both operations, we use functions from the `OpenCV` library, and experiment with different kernel sizes for our morphological smoothing. We arrive at a kernel size of 8×8, resulting in smoothing as seen in Fig. 4.

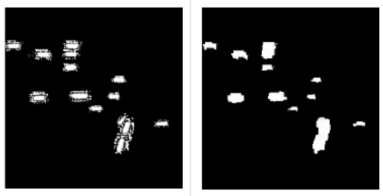

FIG. 4 Object masks generated by our network during inference: (a) before smoothing (b) after smoothing

For object detection, we split the supervised data into a training set consisting of 25 scenes, and a validation set consisting of 3 scenes. We report the average threat scores in Table I, for both the untrained and autoencoder-pretrained generator network.

Model	Train	Val.
Untrained	$2.189e{-5}$	$1.943e{-5}$
Pre-trained	$7.603e{-6}$	$6.012e{-6}$

TABLE I Average threat scores on training and validation sets

To our chagrin, our object detection model underperforms our expectations. We posit this could be due to a number of factors. Ideally, the generator network should learn feature maps that (a) are indicative of a top-down view of each sample; and (b) recognize the parts of each image that correspond to objects. We use a similar methodology for both road map generation and object detection – while the network architecture enables viewpoint shift, it fails to learn features that indicate the presence of objects. If we had additional time remaining, we would explore more sophisticated object detection modules such as Faster-RCNN. Unfortunately, we spent a considerable amount of time investing in another object detection method described in the following section, which left us little time when it failed to produce results we were satisfied with.

V. FAILURE: AN OPPORTUNITY TO LEARN

This section presents a broad overview of some ideas that we spent time working on over the course of the project, and consider to be relevant and promising for future work; these did not make it into the competition submission because we were not able to reach a reasonable baseline using these methods in the time allotted.

V.A. Learning good representations via unsupervised pretraining

The model that we submitted for the competition made use of an untrained deep convolutional network based on ResNet architecture. We considered replacing that network with a pretrained encoder-decoder network, using self-supervised learning methods to obtain good representations of the source images prior to performing the generative task. For this experiment, we used a contrastive method proposed by Misra et al. (2020)[8], in order to learn a pretext-invariant representation.

We trained an 18-layer Residual network (ResNet) using PIRL, coupled with the rotation pretext task defined by Gidaris et al[9]. The training objective is to learn a representation for each training sample that is (1) pretext-invariant and (2) learns meaningful features by maximizing the distance between a [transformed] sample and disparate samples. The PIRL model is optimized using a Noise Contrastive Estimation (NCE) loss function.

We trained the PIRL model on the unlabeled dataset, contrasting each sample against randomly selected 8000 negative samples. After training for 17 epochs, our training loss had reduced from 9.5 to 1.588, and was still decreasing. Without sufficient time to properly incorporate the results of the pretrained encoder from this experiment, we were unable to evaluate its prospective contribution to downstream tasks such as encoding representations for the generative task, or as we describe in the following section, using it as a backbone for a depth prediction network.

V.B. Geometric scene reconstruction via depth prediction

For object detection, we explored using a Pseudo-LiDAR approach similar to that presented in Wang et al. (2020)[10]. If we can predict depth maps directly from monocular images, then projecting the 2D depth map into 3 dimensions as a point cloud might allow us to then perform object detection in voxel space, using established and high-performing methods for true LiDAR 3D object detection. Moreover, if we viewed the 3D reconstructed scene from a top-down view only, then the problem is potentially reduced to a 2D object detection task.

A popular existing method for predicting depth from images is to learn to predict a disparity map from a monocular (single-camera) source, by training a network which has access to ground truth stereo images with a known baseline during training. This disparity map can then be inverted to obtain a depth map of the scene (Godard et al. (2017)[11]).

FIG. 5 Comparison of RGB monocular source images with predicted disparity maps (monodepth2 trained for 5 epochs).

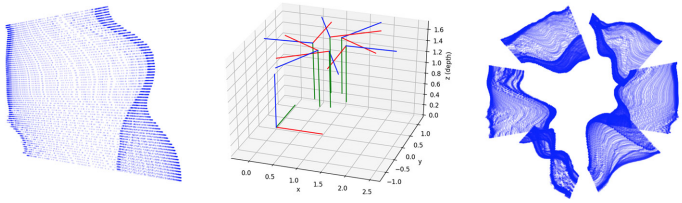

FIG. 6 (a) single top-down view of a point cloud representation of a monocular source. (b) visualization of the 6 camera axes in a shared global space. (c) view of 6 top-down views from a single sample, displayed in the common coordinate system.

In a follow-up work, Godard et al. (2019)[12] expanded their method to be able to learn to predict disparity without the use of stereo images, by using consecutive video frames and introducing an additional network to predict the pose change (relative baseline).

Godard et al. use a Resnet[13] pretrained backbone for their pose and depth predictor networks. As we were restricted from using any outside data or pretrained weights in the competition, we simply trained the monodepth2 model with an untrained Resnet backbone, using a learning rate of 1e-3, and a disparity smoothness of 1e-4 to account for the scaled-down resolution of our input. Training the monodepth2 network was computationally expensive – thus we were only able to train for 5 epochs in total due to resource and time constraints. Fig. 5 displays some of the results of our disparity map predictions. It is evident that there is a wide range in the qualitative accuracy of the predictions. While the results look promising, they exhibit high variance. Perhaps accuracy could be improved by training for far more epochs, or using a pretrained backbone network.

It was nevertheless a worthwhile experiment to take the predicted depth maps and project them into 3D space, taking advantage of prior knowledge of the individual camera rotation, position, and intrinsic matrices. Fig. 6 illustrates process of transforming 6 point clouds into a shared global coordinate space. Due to the inconclusive depth prediction, the resulting top-down global view lacks a clear semantic interpretation. However, we suspect that by improving our depth prediction model training, this approach leaves much to be explored. The next logical step in this approach would be to apply a 2D object detection method to the top-down view, but we decided that the results did not reach a semantically meaningful enough state to proceed further at this time.

VI. FUTURE WORK

We feel that concatenating the six perspective images into a single 18-channel tensor may introduced preventable noise coming from our input signal. In theory, convolutional neural networks perform best in vision tasks when they are able to take advantage of the stationarity, locality, and compositionality that is present in natural image signals. By concatenating the six images prior to obtaining good representations, we acknowledge that artificial discontinuities were introduced between images that we believe hurt our network performance. In future experiments, we would attempt to produce a good representation for each individual image prior to concatenating them into a vector, which we we could then pass through a dense layer to eliminate the issue of signal discontinuity.

Further efforts would be well spent conducting more exhaustive trials and evaluation of our current methods, as well as better integrating some of our disconnected efforts, such as using the self-supervised encoder for the generative task, and for improving the depth prediction network to allow more robust exploration of 3D scene reconstruction, as a means of better understanding the physical world surrounding us.

VII. CONCLUSION

We investigated the use of generative networks for predicting road maps and object maps in the context of autonomous driving. While our results certainly leave performance gains in these tasks to be desired, we do feel that our experiments demonstrate promise for the use of generative networks, given sufficient data to learn from.

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS* (2014).
[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *ICLR* (2016).
[3] S. O. Mehdi Mirza, "Conditional generative adversarial nets," in *arXiv* (2014).
[4] L. A. Gatys, A. S. Ecker, and M. Bethge1, "A neural algorithm of artistic style," in *arXiv* (2015).
[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR* (2018).
[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICLR* (2015).
[7] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," Computer Vision, Graphics, and Image Processing **30**, 32–46 (1985).
[8] I. Misra and L. van der Maaten, "Self-supervised learning of pretext-invariant representations," in *arXiv* (2019).
[9] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *ICLR* (2018).
[10] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *CVPR* (2020).
[11] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR* (2017).
[12] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *ICCV* (2019).
[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR* (2016).