FarSight: Long-Range Depth Estimation from Outdoor Images

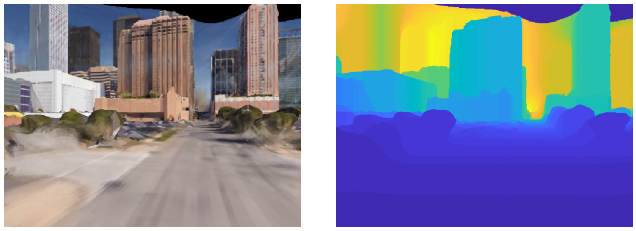
Md. Alimoor Reza[1], Philip David[2] and Jana Košecka[1]

***Abstract*— This paper introduces the problem of long-range monocular depth estimation for outdoor urban environments. Range sensors and traditional depth estimation algorithms (both stereo and single view) predict depth for distances of less than 100 meters in outdoor settings and 10 meters in indoor settings. The shortcomings of outdoor single view methods that use learning approaches are, to some extent, due to the lack of long-range ground truth training data, which in turn is due to limitations of range sensors. To circumvent this, we first propose a novel strategy for generating synthetic long-range ground truth depth data. We utilize Google Earth images to reconstruct large-scale 3D models of different cities with proper scale. The acquired repository of 3D models and associated RGB views along with their long-range depth renderings are used as training data for depth prediction. We then train two deep neural network models for long-range depth estimation: i) a Convolutional Neural Network (CNN) and ii) a Generative Adversarial Network (GAN). We found in our experiments that the GAN model predicts depth more accurately. We plan to open-source the database and the baseline models for public use.**

I. INTRODUCTION

The ability to detect and classify objects and activities at a long range of distances will be an important capability when autonomous vehicles and robotic systems eventually become operational in outdoor urban environments. For example, the speed at which autonomous vehicles will be able to operate will depend, in part, on the size of the surrounding region in which these systems are able to reliably observe and detect objects and events. The larger this region, the more time the autonomous system will have at a given speed to react to events in its environment. Security operations in large urban environments will be similarly enhanced by the ability of autonomous systems to detect and classify objects at greater distances than is currently possible.

A typical image of an outdoor (urban) environment may contain objects a meter in front of the observer all the way out to a kilometer or more. Humans are able to understand most of what they observe, even for objects that subtend very small regions in their view. While state of the art object detection and classification systems work extremely well when objects occupy a sufficiently large number of pixels in an image, they often fail when the object size is reduced at a given resolution; but at this failure point, humans are usually still able to successfully complete the task. Our goal is to mimic this capability in autonomous perception systems. It has been shown [12] that using depth cues with the object detection pipeline brings significant improvements to perception systems. Existing monocular depth estimation algorithms [1][3] provide depth estimates only up to approximately 100 meters in outdoor environments. This limitation arises primarily from the range limitations of the depth sensors (LIDAR, laser, or stereo) that are used to generate the training data [2][23]. The focus of this work is on extending the reach of single view depth estimation algorithms by order of magnitude. This in turn requires the acquisition of depth training data that is an order of magnitude longer in range.



(a) Rendered RGB (b) Rendered depth

Fig. 1: Our objective is to estimate depth out to 1000 meters from a single color image. We first propose a novel pipeline for generating large collections of image and depth pairs of various cities in the United States. Then, using this dataset, we train two deep neural network architectures to estimate long-range depth.

One useful direction for generating long-range depth is to synthetically create a 3D virtual world and generate depth renderings of the model from different view points [21], [22]. In this work, instead of using synthetic 3D CG models, we leverage realistic renderings of real-world environments in Google Earth[1]. We used the snapshots of renderings along with a state of the art Structure from Motion (SfM) algorithm to create partial 3D models of a number of cities. The renderings of the 3D world models are utilized to train data-intensive Deep Neural Network (DNN) models to estimate depth. The apparent ease at which humans are able to roughly estimate depth motivates us to extend single-view depth estimation using this data driven approach: Humans learn to estimate depth using a variety of monocular depth cues [11], including perspective, absolute and relative image size (subtended visual angle) of known objects, occlusion of more distant objects by closer objects, object surface texture which changes with depth, haze, and position of objects relative to the horizon.

We make the following contributions in this paper:

- We extend the outdoor depth estimation problem to

[1]Md. Alimoor Reza and Jana Kosecka are with the Department of Computer Science, George Mason University, Fairfax, VA, USA {mreza,kosecka}@gmu.edu

[2]Philip David is with the U.S. Army Research Laboratory, Adelphi, MD, USA philip.j.david4.civ@mail.mil

[1]https://www.google.com/earth/

a range that is an order of magnitude higher than previously proposed. The maximum range in our dataset is 1000 meters.
- We propose a novel strategy for generating large collections of 3D reconstructed models of urban environments. Our models extend existing 3D world repositories [21], [22].
- We formulate the problem of depth estimation as that of image-to-image translation [18] using a Generative Adversarial Network (GAN). The depths are quantized into a fixed number of bins, and the network is trained to predict the bin value for each pixel.

The rest of the paper is structured as follows. We first discuss relevant works for depth estimation and 3D modeling from synthetic data. Then, we describe our approach to generating the long-range depth dataset. We next discuss our experiments with two deep neural network methods for depth estimation and provide their comparison. And finally, we review our findings and future directions.

II. Related Work

In the related work, we discuss relevant methods for monocular depth prediction and synthetic data generation.

Monocular Depth Estimation: Make3D [1][2] introduces depth prediction from a single image in outdoor settings as a labeling problem in the Markov Random Field framework. It relies on different handcrafted features for the model learning process and uses laser range data associated with views for training and evaluation. Liu et al. [8] follow the handcrafted features route but use semantic labels in image as an extra signal to recover the depth. Eigen [3] revisits the problem of single image depth recovery utilizing more powerful convolutional neural network features in both indoor and outdoor settings. It learns one network for coarse resolution depth prediction and another network for fine-scale depth estimation. The work of [9] formulates a joint estimation of depth and semantic segmentation and outperforms the single depth recovery when carried out alone. Mousavian et al. [7] also perform joint estimation of depth and semantic segmentation tasks using a multiscale convolutional neural network. It learns a shared underlying feature representation during the joint training of the two tasks. Other approaches in the domain of joint depth and semantic segmentation include [10]. Eigen et al. [4] extend the problem of depth estimation together with two other tasks: surface normal estimation and semantic segmentation. The work of Liu et al. [6] revisited the problem by segmenting the image into superpixels and predicted the depth of each superpixel in a continuous Conditional Random Field framework. The work of Cadena et al. [5] followed a different route than the more popular CNN architecture and tackled depth estimation using an Auto-Encoder architecture. They demonstrated that the Auto-Encoder learns a shared representation during the joint reconstruction of three different modalities: image, depth, and semantics. In our work, we follow an alternative architecture and formulate the depth prediction task as domain translation in the framework of the conditional Generative Adversarial Network (cGAN). Depth prediction can be thought of as an image-to-image translation problem. The work of Isola [18] demonstrates that the cGAN can be generalized to different translation tasks such as maps to aerial photos, black and white to color images, or sketch to photo translation. We demonstrate in this work that the cGAN can be used to predict depth from single color images.

Synthetic Data: There has been an effort to enhance the synthetic data generation of urban scenes using 3D virtual worlds [22][21]. Gaidon et al. [22] generate a 3D virtual world which is a synthetic clone of the existing videos from KITTI [23]. The virtual world automatically allows them to generate pixel-level labeling of objects in the rendering as well as the depth associated to them. Additionally, they also automatically generate tight bounding boxes around objects of interest (e.g., cars). Experimentally, they show that the performance of deep networks can be improved with a pre-training step of the model on synthetic data for the task of multi-object tracking (MOT). SYNTHIA [21] is another virtual world-based synthetic dataset for automatic generation of large number of pixel-level labels in urban scenes for the task of semantic segmentation. The work of Richter et al. [24] proposed a method for automatic generation of pixel-level labels from commercial computer games. In indoor settings, SceneNet [25] provides a framework for generating annotated 3D indoor scenes. Our work is closely related to the work of [25] in the way we obtain our rendered images from the 3D models.

III. Approach

A large collection of color images of cities across the United States can be extracted using Google Earth. As the corresponding depth images are not available from Google Earth, a key component in the generation of RGB and depth pairs is the ability to reconstruct 3D models of city scenes. Our approach starts with the accumulation of color images of city scenes from various viewpoints followed by 3D reconstruction from these multiple views. From the 3D model of a city, we render the RGB and depth image pairs from the viewpoint of a moving observer in a predefined trajectory on the ground plane of the 3D model. Any number of trajectories can be defined through a 3D model thereby allowing us to render a huge collection of RGB and depth pairs. The subcomponents of the *Data Generation* method are discussed in detail in the following sections.

A. Scene Image Generation

For our task of depth estimation in urban environments, we are primarily interested in street scenes. To this end, we leverage the large scale model repositories of cities in different geolocations in the United States provided by the publicly available Google Earth. Unfortunately, directly exporting 3D models from Google Earth has been disabled. We circumvent this issue by reconstructing large portions of a city in 3D from images acquired from multiple viewpoints. To create a 3D reconstruction of a scene, we start by recording a video clip of the desired scene using Google

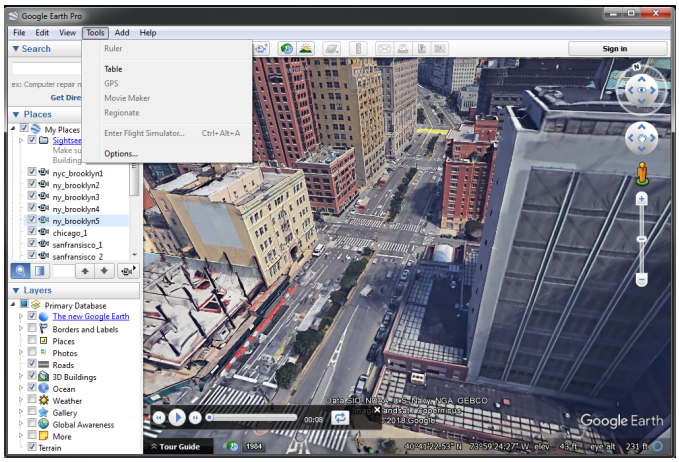


Fig. 2: Collection of a video clip from Google Earth Pro in New York City. The *Movie Maker* feature enables this option. Images are exported from the recorded video clip at 30 fps.

Earth Pro. The duration of the video clips ranges from 2 to 15 minutes. Once the clip has been recorded, images from the entire clip are extracted at a rate of 30 frames per second. Figure 2 shows an example of the image collection process. Notice that these images are more realistic than synthetic images generated from any traditional game engine [21], [22] since Google Earth's model is generated from real images of the scene.

B. 3D Reconstruction

We 3D reconstruct the city model from the extracted images of a scene. There are several available reconstruction pipelines such as VisualSfM [14], COLMAP [16]. Most of these SfM-based pipelines produce sparse 3D point-cloud reconstructions of the scene [15]. The resulting sparse point cloud can be augmented with denser 3D points via a multi-view stereo (MVS) step [17]. We follow a similar 3D reconstruction pipeline using Autodesk Remake[2] and generate the 3D reconstructed model using a subset of the images extracted from Google Earth. The reconstructed models are exported to 3D model files (in the *.obj* format) for our rendering pipeline to generate unlimited numbers of rendered color and depth images. For each 3D reconstruction, we manually select a few viewpoints from which to generate the color/depth pairs.

C. Rendering pipeline

Structure from motion, even with a calibrated camera, is able to recover 3D geometry only up to an unknown scale. One advantage of generating 3D models from Google Earth imagery is that we can solve for the scale ambiguity by measuring the actual geographical distance (in meters) between two locations[3] and then use it to constrain the distance between the corresponding locations in the 3D reconstructed model. Figure 4 demonstrates the distance association between two locations in a 3D reconstructed model of *Chicago*. Figure 4a depicts the geographical distance (in meters) between two such locations. These locations correspond to *Point 2* and *Point 3* in the reconstructed model in Figure 5. Figure 4b shows the corresponding distance in the 3D reconstructed model (in millimeters). Notice that Figure 4b is just the textured map view of the model in Figure 5. The ratio between the two distances allows us to scale the depth values during the rendering process correctly. Let's define the ratio between these two distances to be the scale factor $scale$. We manually assign these location associations in the 3D reconstructed model and their corresponding locations from Google Maps. The street intersections are selected as seed-locations for data-association. Figure 5 shows the reconstructed 3D model from *Chicago* and 3D points that are selected as the seeds for the scale alignment of the physical world to the model data.

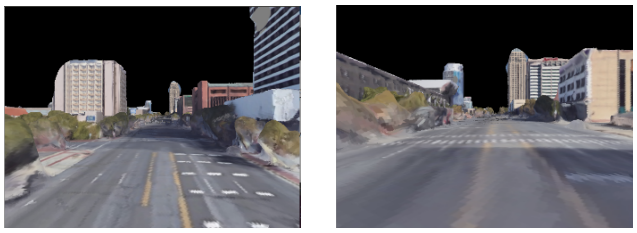


Fig. 3: Two snap-shots of the rendering process in two different virtual camera trajectory locations inside the *Salt Lake City* reconstructed 3D model.

Between two seed locations A and B in the 3D model, we can generate a trajectory of a moving virtual camera. The 3D world coordinates of the seed locations are denoted by X_A and X_B, respectively. We can move the position of the virtual camera by a fixed number of steps from X_A towards X_B. In each virtual camera position, we render the 3D scene using the OpenGL engine. The units of the rendered vertices are converted into meters by multiplying them by the $scale$ factor defined earlier. The near and far planes of the frustum are set to the values 0.1 and 1000 meters respectively to allow the renderer to generate depth values within a maximum range of 1000 meters for any given virtual camera position. Figure 3 shows two snapshots taken during the rendering process of a 3D model from our repository. The images are rendered at a resolution of 640×480 pixels. The intrinsic parameters of the camera are set to $f_x = 420$, $f_y = 420$, $c_x = 320$ and $c_y = 240$.

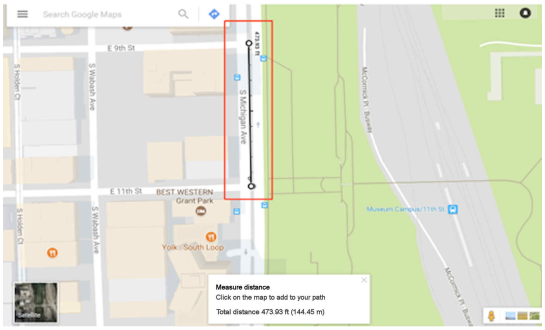
IV. Experiments

We have generated a large collection of rendered RGB and depth image pairs from four scenes in four cities (*New York City, Salt Lake City, Houston,* and *Miami*) in the United States.[4] To validate our long-range depth estimation problem,

[2]The functionality of Autodesk Remake has been moved into Autodesk ReCap Pro, https://www.autodesk.com/products/recap/overview.

[3]We measure the distance using http://maps.google.com

[4]An additional 10 city scenes, which have already been collected, will augment the current dataset prior to its public release.



(a) The geographical distance (144.45 meters) between locations *Point 2* and *Point 3*.

(b) The model distance (3.47 millimeters) between locations *Point 2* and *Point 3*.

Fig. 4: This figure demonstrates the distance association between two geographical locations in the 3D reconstructed model shown in Figure 5. The distance ratio is used to scale depth values in our rendering pipeline (best viewed in color).

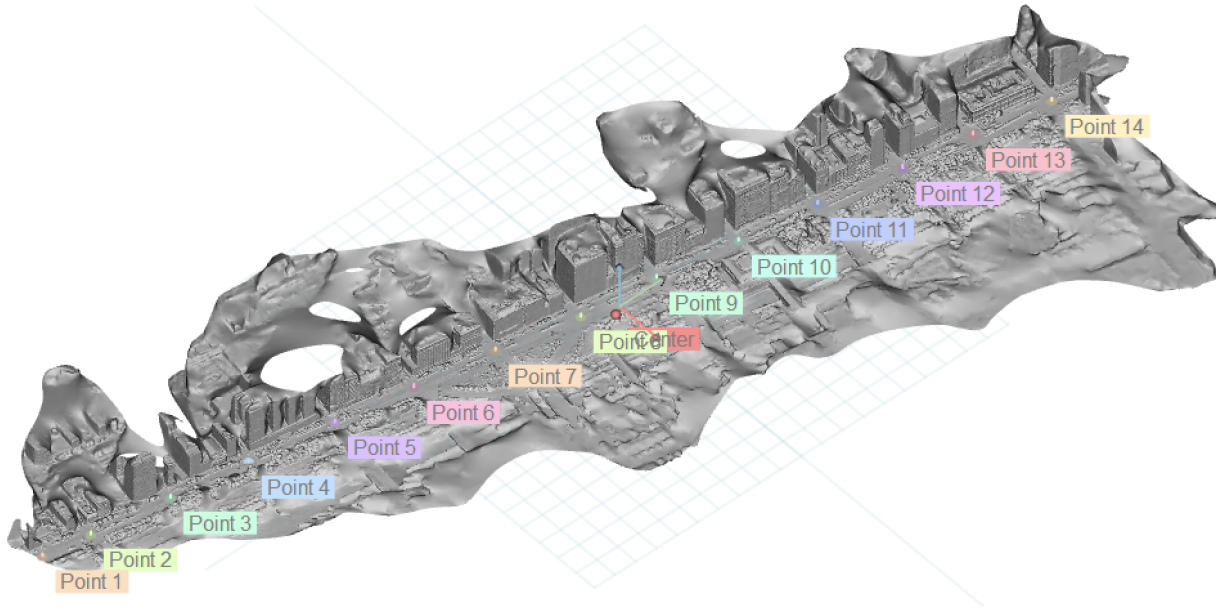


Fig. 5: An example 3D reconstructed model from our dataset (best viewed in color). The keypoints (*Point 1*, *Point 2*, *Point 3*, etc.) are manually selected in the 3D model. The real-world locations of these points are determined using Google Maps, and these locations are used to compute the scale factor between units in the 3D model and the real-world values. Note that the texture map has been turned off.

we picked two different Deep Neural Network (DNN) architectures to train on this dataset. For our experimental setup, we used subsets of the rendered RGB and depth images to train and test our DNNs on. Table I lists the number of images from each city model used in our experiments. The depth values are quantized into a fixed number of bins. The quantization factor used in our experiments is 4: depth values from 0 to 4 meters are put into bin number 1; depth values from 5 to 8 meters are put into bin number 2; and so on. The predicted value is multiplied by the same factor 4 to obtain the depths in meters. The evaluation is performed in units of meters for all experiments.

Convolutional Neural Network (CNN): We picked a variation of the multi-scale CNN architecture proposed in the work of [7]. This method jointly trains a CNN architecture for the combined task of semantic segmentation and depth estimation. The architecture learns a shared underlying feature representation during the joint training of the two tasks. We discard the semantic module from the architecture and directly train the network for depth estimation. We divide

Scene	# Frames
New York City	300
Miami	250
Houston	250
Salt Lake City	100
Total	900

TABLE I: Number of selected frames for long-range depth prediction

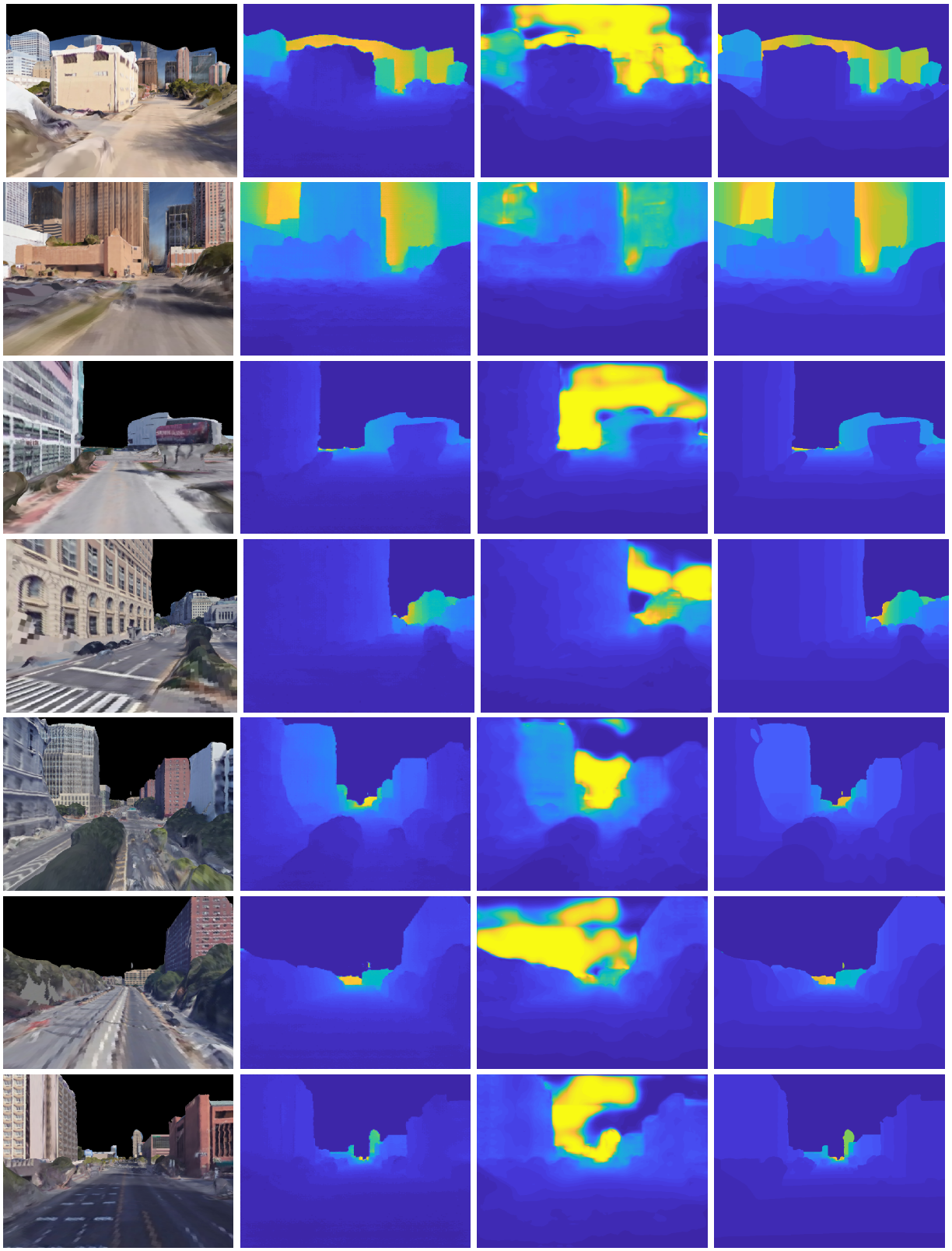


Fig. 6: Qualitative results for the depth prediction on the random-split experiment as reported in Table II. From left to right we show the input RGB image, the prediction from GAN model, prediction from CNN model, and the ground truth depth (best viewed in color).

the range of possible depth values into 250 bins. In pixel location i for each bin j, the network predicts the depth bin probability $p(j|x_i)$ as follows:

$$p(j|x_i) = \frac{e^{r_{i,j}}}{\sum_{k=1}^{255} e^{r_{i,k}}} \tag{1}$$

x_i is the feature vector at pixel location i and $r_{i,j}$ is network's response at pixel location i and at bin location j .The depth value is $d(x_i^*)$ is computed as the weighted sum over all the bins where depth bin probabilities are used as weights as follows:

$$d(x_i^*) = \sum_{j=1}^{255} j \times p(j|x_i) \tag{2}$$

Similar to the method of [7], we compute the scale invariant depth loss to encourage the network to predict the correct relative depth rather than absolute depth. The scale invariant depth loss is estimated as follows:

$$L = \frac{1}{n^2} \sum_{i,k} ((\log d(x_i) - \log d(x_k)) - (\log d(x_i^*) - \log d(x_k^*)))^2 \tag{3}$$

where $d(x_i^*)$ is the predicted depth and $d(x_i)$ is the ground truth depth at pixel location i. The architecture is fine-tuned using the pre-trained models from DeepLab [20]. The models are trained for 20000 iterations with a batch size of 1 and base learning rate of 10^{-4} in each experiment. The momentum parameter is set to 0.99.

Generative Adversarial Network (GAN): We formulate the depth prediction task as a domain translation in the framework of Generative Adversarial Networks (GAN). Depth prediction can be thought of as an image-to-image translation problem, where the goal is to translate an RGB image (source domain) to a depth image (target domain). A standard GAN has two components: a generator G and a discriminator D. The combined network is adversarially trained where the goal of the generator G is to produce images that fool the discriminator D into identifying 'fake' images as 'real' ones. The D is trained as well to discriminate 'fake' images from the 'real' ones. We use the conditional-GAN architecture from the work of Isola et al. [18]. In this variation of GAN, in addition to fooling the D network, G is asked to generate the 'fake' image that is as close to a given input in an $L1$-loss sense. We used the U-Net [19] architecture for the G network. The models are trained for 200 epochs with a batch size of 1 for each experiment. The weight of the $L1$-loss is set to 100.

A. Depth Evaluation Metrics

We evaluated our DNN architectures using three different metrics. Consider d_i and d_i^* as ground truth depth and predicted depth, respectively, at pixel location i in an image, and assume that there are N pixels in the image. We compute the following three metrics to measure the performance.

a) Absolute Relative Error: $M_a = \frac{1}{N} \sum_{i=1}^{N} \frac{|d_i - d_i^*|}{d_i}$

b) Linear RMSE: $M_b = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (d_i - d_i^*)^2}$

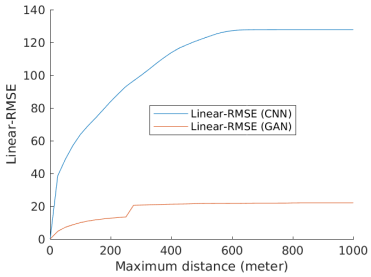


Fig. 7: Visualization of the *linear RMSE* as a function of depth (meters) for the experiment in Table II

Metric	CNN	GAN	Liu et al.[6]
Absolute Relative Error	2.39	**0.23**	0.67
Linear RMSE	126.42	**21.74**	92.72
Scale Invariant MSE	**0.07**	0.09	0.46

TABLE II: Depth prediction accuracy comparison for the random split experiment in Section IV-B using three different metrics: *Absolute Relative Error*, *Linear RMSE*, and *Scale Invariant MSE*. Lower error is better for all three metrics.

c) Scale Invariant MSE: $M_c = \frac{1}{N} \sum_{i=1}^{N} (log(d_i^*) - log(d_i) + \alpha(d^*, d))^2$. whereas $\alpha(d^*, d) = \frac{1}{N} \sum_{i=1}^{N} (d_i - d_i^*)$.

B. Depth Estimation Results on Random Split

In this experiment, we followed a similar setup as was done in the work of Saxena et al. [1]. We combined all of the 900 rendered images from all the cities and made a random split of 540 training images and 360 test images. We trained both DNN networks using the quantized depth values as ground truths. The predicted depth is multiplied by the quantization factor 4 to convert the bin into meters. Figure 6 shows qualitative comparisons between the two methods on different city images. The evaluation is done on the non-zero ground truth regions The sky region in most rendered images contain depth values of zero. A visible portion of sky appears in our 3D reconstructed models only in the city of *Houston*. The mean values across all 360 images are reported in Table II. GAN baseline performed better than the CNN baseline on the first two metrics, and comparable in the third-*scale invariant MSE* metric. We also evaluated the 360 test images on the method of Liu et al. [6] using their publicly available code and model trained on Make3D [2] dataset. The results are reported in the third column of Table II. The maximum range of this training data is 81 meters [2]; hence, the model fails to accurately predict more distant depths. Figure 7 shows the *linear RMSE* error as a function of the true depth. This *linear RMSE* metric reports the error in meters. The error of the GAN model is much lower than that of the CNN model at all depths, and also levels off much quicker.

C. Depth Estimation Results on an Unseen City

We conducted two other experiments where we train our two networks with all images from three different cities and

Metric	CNN	GAN
Absolute Relative Error	3.23	**0.63**
Linear RMSE	176.28	**30.46**
Scale Invariant MSE	**0.07**	0.24

TABLE III: Depth prediction comparison for three different error metrics. The unseen city images were picked from *New York City*. Lower error is better.

Metric	CNN	GAN
Absolute Relative Error	6.30	**1.53**
Linear RMSE	272.75	**57.17**
Scale Invariant MSE	**0.19**	0.27

TABLE IV: Depth prediction accuracies comparison for three different metrics. The models were trained using images from *Miami, New York City,* and *Houston*. The models were evaluated on images from *Salt Lake City*. Lower error is better.

then evaluation is done on all images from the remaining city. In the first of these tests, the networks are trained using 600 images in total from *Miami*, *Houston*, and *Salt Lake City*. The trained models are evaluated on the 300 images from *New York City*. Table III shows the performance comparison between the two methods. In comparison to our previous experiment, the performance deteriorates since the evaluation city typically has significant differences in appearance and structure. In the next experiment, we trained the models using 800 images from *Miami, New York City,* and *Houston* in total. In Table IV we report the performance comparison where the models are evaluated on the 100 images from *Salt Lake City*. We observe a similar trend as in Table III.

V. CONCLUSION AND FUTURE DIRECTIONS

In conclusion, this work extends the outdoor depth estimation problem to a range that is an order of magnitude greater than previously proposed. In order to train and evaluate potential solutions, we proposed a novel strategy for generating large collections of RGB and their corresponding ground-truth depth images for ranges up to 1000m using 3D reconstructed models in urban environments. We formulate the problem of depth estimation as that of image-to-image translation [18] using a GAN method and found in our experiments that the GAN performed better than a CNN baseline in *Absolute Relative Error* and *Linear RMSE* metrics. In the future, we would like to extend our repository of 3D models and generate more renderings from them. We would also like to address some artifacts in the 3D reconstructed models such as the inconsistent appearance of sky regions from the 3D reconstruction. Another future direction would be exploring whether the rendered images can be effectively utilized to pre-train a deep neural network to improve depth prediction performance in real images.

REFERENCES

[1] A. Saxena, S. Chung, and A. Ng, Learning Depth from a Single Monocular Image, In Advances in Neural Information and Processing System (NIPS), 2005.

[2] A. Saxena, M. Sun, and A. Ng, Make3D: Learning 3D Scene Structure from a Single Still Image, In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2009.

[3] D. Eigen, C. Puhrsch, and R. Fergus, Depth Map Prediction from a Single Image using a Multi-Scale Deep Network, In Advances in Neural Information and Processing System (NIPS), 2014.

[4] D. Eigen, and R. Fergus, Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, In International Conference on Computer Vision (ICCV), 2015.

[5] C. Cadena, A. Dick, and I. Reid, Multi-modal Auto-Encoders as Joint Estimators for Robotics Scene Understanding, Robotics: Science and Systems (RSS), 2016.

[6] F. Liu, C. Shen, and G. Lin, Deep Convolutional Neural Fields for Depth Estimation from a Single Image, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[7] A. Mousavian, H. Pirsiavash, and J. Kosecka, Joint Semantic Segmentation and Depth Estimation with Deep Convolutional Networks, In International Conference on 3D Vision (3DV), 2016.

[8] B. Liu, S. Gould, and D. Koller, Single Image Depth Estimation from Predicted Semantic Labels, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010.

[9] L. Ladicky, J. Shi,and M. Pollefeys, Pulling Things Out of Perspective, In IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2014.

[10] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. Yuille, Towards Unified Depth and Semantic Prediction from a Single Image, In IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[11] A. Torralba and A. Oliva, Depth Estimation from Image Structure, In IEEE Transactions on Pattern Analysis& Machine Intelligence, 2002.

[12] D. Hoiem, A. Efros, and M. Hebert, Closing the Loop in Scene Interpretation, In IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[13] D. Scharstein and R. Szeliski, A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms, In International Journal of Computer Vision (IJCV), 2002.

[14] C. Wu, VisualSFM : A Visual Structure from Motion System.

[15] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. Berg, A Dataset for Developing and Benchmarking Active Vision, In IEEE International Conference on Robotics and Automation (ICRA)", 2017.

[16] J. Schonberger, and J.M. Frahm, Structure-from-Motion Revisited, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[17] J. Schonberger, E. Zheng, M. Pollefeys, and and J.M. Frahm, Pixelwise View Selection for Unstructured Multi-View Stereo, In European Conference on Computer Vision (ECCV), 2016.

[18] P. Isola, J. Zhu, T. Zhou, A. Efros, Image-to-Image Translation with Conditional Adversarial Networks, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[19] O. Ronneberger, P. Fischer, and T. Brox, U-net: Convolutional Networks for Biomedical Image Segmentation. In MICCAI, 2015.

[20] L. Chen, G. Papandreou, I. Kokkinos K. Murphy and A. Yuille, Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs, In International Conference on Learning Representation (ICLR), 2015.

[21] G. Ross, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez, The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes, In IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[22] A. Gaidon, Q. Wang, Y. Cabon, and Elenora Vig, Virtual Worlds as Proxy for Multi-Object Tracking Analysis, In IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[23] A. Geiger, P. Lenz, and R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[24] S. Richter, V. Vineet, S. Roth, and V. Koltun, Playing for Data: Ground Truth from Computer Games, European Conference on Computer Vision (ECCV),2016.

[25] A. Handa, V. Patraucean, S. Stent, R. Cipolla, SceneNet: an Annotated Model Generator for Indoor Scene Understanding, In International Conference on Robotics and Automation (ICRA), 2016.