3D Point Estimation Using A Recursive Neural Network

Hanna K. Winter*
Student, Stanford University

Abstract

3D reconstruction and pose estimation have been huge areas of research in recent years. Success in these two areas have allowed enormous strides in augmented reality, 3D scanning, and interactive gaming. However, utilizing machine learning in this realm has been a difficulty since creating relevant datasets is extremely time consuming. Additionally, 3D annotated video datasets do not exist yet. By rendering 3D objects, we create a dataset of image sequences to emulate video frame data. Using a VGG net [Simonyan and Zisserman 2014] and a RNN, we introduce a model that can predict a object's 3D bounding box based on synthetic video-like image sequences. Our model can be extended to predict 3D feature points to create point cloud data for 3D reconstruction.

1 Introduction

Creating 3D geometry from 2D image data has been a very active field of research for some time. However, machine learning, especially deep learning, techniques for these tasks has only been taken advantage of as of late. With the huge successes seen in deep learning for computer vision tasks such as object classification, detection, and segmentation, it is highly possible these techniques can be successfully applied to 3D reconstruction and recognition. In fact, strides have already been made as seen in [Su et al. 2015]. In this paper, we explore a new deep learning model architecture to perform preliminary 3D object reconstruction.

Currently, one of the inherent problems with 3D pose estimation or 3D reconstruction from a 2D RGBA image is dealing with perspective ambiguity. Because of this, RGBA image data taken from a variety of camera angles is required for proper reconstruction. As a result, videos from phones or drone cameras that capture an object from many angles is the main data used in 3D reconstruction. Thus, it is reasonable to utilize multi-frame RGBA data of a given object to perform 3D pose estimations. In order to processes multi-frame image data as a sequence rather than as separate entities, we propose a deep learning architecture using a Recursive Neural Network (RNN).

Because of the lack of 3D annotated video datasets, we utilize the idea of rendering synthetic 3D annotated data [Su et al. 2015]. We create a dataset of rendered image sequences to emulate video frames and use a a VGG Net [Simonyan and Zisserman 2014] attached to a RNN to predict an objects 3D bounding box for a given image sequence. Extending this technique to predict feature points for point cloud creation would be the most useful application.

2 Related Works

The work from Render for CNN [Su et al. 2015] makes great strides to use deep learning for camera pose estimation. Not only does the paper render single image data from ShapeNet [Chang et al. 2015] models but it also uses a convolutional neural network (CNN) to predict the azimuth, elevation, and tilt angles of the camera pose.

Render for CNN [Su et al. 2015] utilizes 3D rendering to create a massive dataset for viewpoint estimation using a CNN. This state of the art approach renders a massive dataset with pose estimation labels and trains a CNN on the synthetic data. Then, the CNN is tested on the Pascal3D+ [Xiang et al. 2014] dataset and performs very well on single RGB images. However, the model is extremely limited and can only be tested on object categories it has previously been trained on. This is because after the last convolutional layer of the model, the CNN is split into 12 different fully connected layers, one layer for each category. As a result, the model also scales linearly with the number of object categories it can properly evaluate. Additionally, the model cannot predict the distance of the camera from the object in question because the input data is only single images. We attempt to combat these issues with a general model that trains on sequence data containing images taken from different camera angles.

Recurrent neural networks (RNNs) are commonly used for word sequence predictions because the structure innately uses an ordered sequence of data to make predictions. Because of this structure, we decided to use an RNN as part of our model to operate on image sequence data.

3 Data

We created a dataset of synthetic images by rendering ShapeNet models [Chang et al. 2015] from a variety of different viewpoints. We use the rendering pipeline from the Render For CNN paper [Su et al. 2015] with a few adjustments. Specifically, we changed the functionality to render image sequences rather than single images and to output both the 2D and 3D bounding boxes for each image. As is done in Render For CNN and the PASCAL3D+ dataset [Xiang et al. 2014], we create a dataset containing 12 categories of objects.

3.1 Rendering Pipeline

We first render each image sequence of a 3D model using Blender. The initial viewpoint is drawn from a distribution of viewpoints taken from a real image dataset [Su et al. 2015]. We then semi-randomly interpolate from the original viewpoint for subsequent renderings in a given image sequence. This is done by picking a direction of movement for all degrees of freedom (azimuth, elevation, tilt, distance) and adjusting each value by a random amount in the specified direction. In this manner, we create image sequences that mimic various timestep samples taken from a set of video frames. Note that we chose to do timesteps of video frames. This is done because otherwise our sequences would need to be at least about 120 in length in order to represent just 5 seconds of 24 fps video and this would require a large amount of memory usage during training and testing since the RNN would save all states and outputs for every iteration. Additionally, emulating exact video data can create redundant data since there is little change in viewpoint from frame to frame. During the rendering process, we extract the 3D bounding box for each 3D model to use as labels. This is described as a 6 item vector containing the minimum x, y, and z values and the width, height, and depth of the bounding box.

The images are then cropped according to [Su et al. 2015] with the exception that we keep the same cropping scheme for a given sequence. This results with minimal randomization of cropping within a given sequence as seen in Figure1.

*e-mail:hannawii@stanford.edu

After cropping, the background of the rendered image is filled with an image randomly sampled from the SUN397 dataset [Xiao et al. 2016]. We use the same background image for the entire image sequence and again allow for randomized positioning.

Finally, we preprocess the overlaid images to feed into our VGG+RNN model. All images are resized and/or padded to be 224x224x3. Additionally, instead of having a out-of-box object detection model locate the bounding box for each image, as was done in [Su et al. 2015], we instead output the 2d bounding box for each rendered image and utilize this ground truth for our learning. We apply a 4th channel indicating the 2D bounding box around the object by placing 1's at the pixel locations where the 2D bounding box would be and 0's elsewhere. Note that because of this, we assume ground truth object detection which could have an effect on our results.

3.2 DataSet

We created a dataset of 50 image sequences of length 20 for each of the 12 classes from PASCAL 3D+ [Xiang et al. 2014] for a total of 600 image sequences as our training examples. 10% of this dataset was designated as test data giving 60 testing examples and 530 training examples. Our labels are the ground truth 3D bounding boxes, (x, y, z, width, height, depth), as stated earlier. Note that we use our synthetic renderings as test data. This is generally undesirable as testing on real video or image sequences gives a more accurate assessment of the model's performance. However, as was mentioned before, annotated 3D video datasets with 3D bounding box labels do not exist.

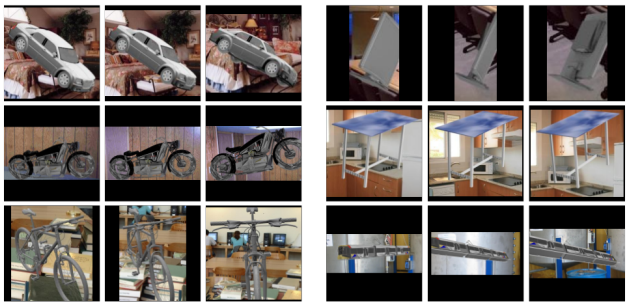


Figure 1: *Dataset Examples. 6 different categories of the 12 total. Each row displays a segment of an image sequence training example.*

4 3D Point Estimation Model

4.1 Network Architecture

We build a deep learning architecture with Tensorflow [Abadi et al. 2015] using an out-of-box VGG19 network [Simonyan and Zisserman 2014] and replace all but the first fully connected layer with an RNN. The output of the first fully connected layer of the VGG19 is a 1x4096 feature vector for each image. The image sequences are fed through the VGG as a large batch of images and reshaped back into a sequence of 1x4096 vectors before going through the RNN. Each of the feature vectors are fed one-by-one in time order to the LSTM cell of the RNN. The hidden size of the LSTM cell is 2048. The final fully connected layer converts the output of the network to a size 6 vector representing the 3D bounding box of the object. An illustration of this can be seen in Figure2.

Note that unlike [Su et al. 2015], we use the same RNN and fully connected layers for all of the 12 object classes.

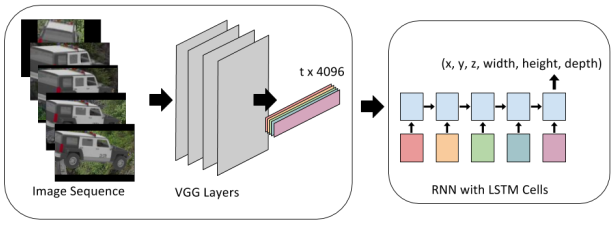


Figure 2: *Network architecture. The VGG19 receives as input a set of t 4-channel images and outputs a $t \times 4096$ matrix. Here, t indicates the number of images for a given timestep. These vectors are fed to the RNN so that each 1×4096 vector is the input at the corresponding timestep.*

5 Experiments

Experimentation and evaluation was all done with our rendered image data. The model ended up not being hyper-tuned due to time constraints. Our batch size of 5 and RNN hidden size of 2048 were chosen to be as large as feasible given the limited computational power. Additionally, because of lengthy training time, we limited the number of RNN layers to 1 and the image sequence length to 15. We used 0.1 for our learning rate, mean squared error (MSE) to compute loss, and the momentum update for our optimizer.

6 Results

Our results come from our model run on our rendered dataset. We use three different metrics to measure performance: mean squared error (MSE), intersection over union (IOU), and accuracy. The MSE is computed as the mean of the squared difference between the ground truth bounding box and our models output. The IOU metric is used in the recent YOLO paper [Redmon et al. 2015] for evaluation of their 2D object detection model. The IOU is calculated by dividing the area of the intersection of the ground truth 2D bounding box with the predicted bounding box by the union of the areas of the two boxes. The closer this ratio is to 1, the better the prediction. For our purposes, we compute the IOU using the volume of the 3D bounding boxes. We say our model predicted correctly if the IOU > 0.3. This is a more lenient bound compared to 2D object detection, the YOLO paper has an IOU > 0.5 as a correct prediction, but we take into account that we are considering another dimension. Table 1 describes the loss, average IOU, and average accuracy of our model on our dataset.

Table 1: *VGG+RNN Results*

	Loss (MSE)	IOU	Accuracy
Train	0.33	0.385	0.66
Test	0.392	0.356	0.60

Results from our VGG+RNN. An 3D bounding box is decided as correctly predicted if the IOU > 0.3.

A visualization of some of our model's output compared to the ground truths is displayed in Figure 3.

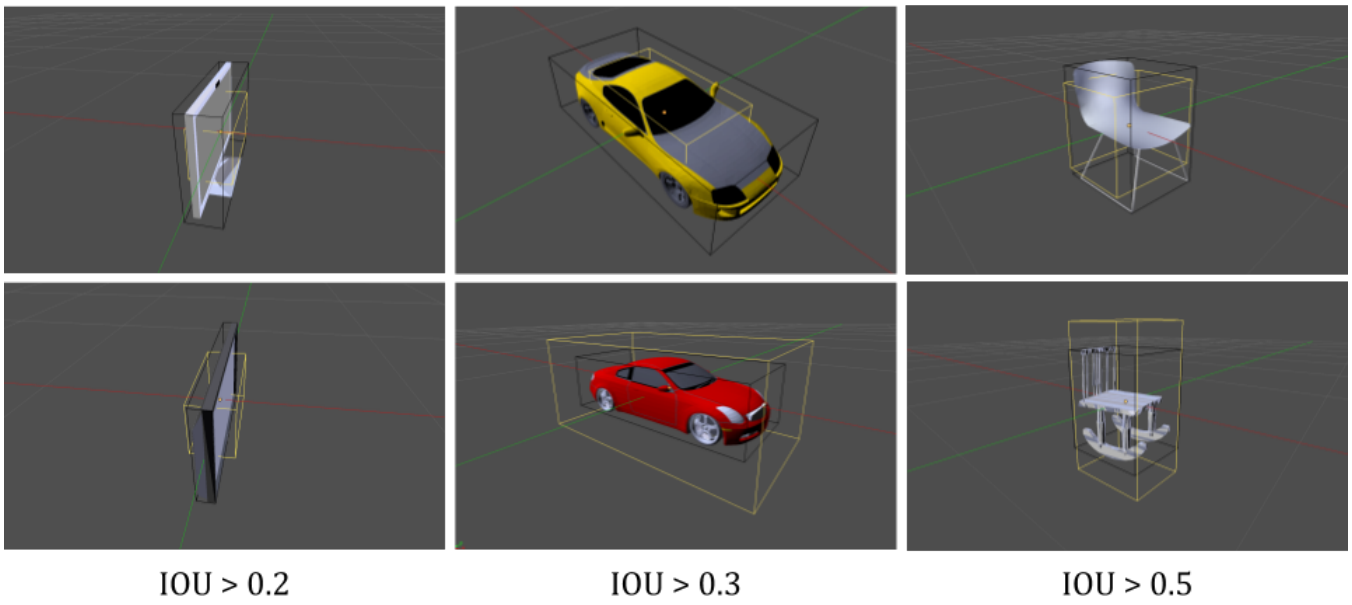
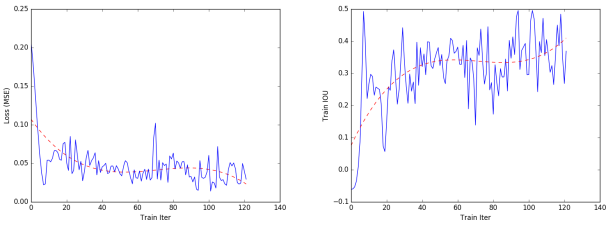


Figure 3: *Results from our model compared with ground truth for 3 different object categories. The yellow bounding boxes are our model's output and the black are the ground truth bounding boxes. The images are taken in Blender.*

7 Discussion

The model was able to produce semi-reasonable results with minimal training of 120 training iterations and batch size of 5. The loss continually decreases over these training iterations, as seen in Figure4, showing that with more training and hypertuning, the model should be able to improve. Additionally, training on image sequences longer than 15 should produce even better results.



(a) *Training Loss (MSE)* **(b)** *Training IOU*

Figure 4: *The training loss and training IOU over 120 training iterations. The red line represents a degree 3 polynomial fitted to the data to better display the trend of the data over time. After 120 iterations (of batch size 5), the loss seems to continue decreasing and the IOU seems to continue increasing.*

Currently testing is done on the rendered image sequences. It would produce a more accurate evaluation of the model if testing were done on real video data or image sequences. 3D annotated datasets with real single images are available for testing but do not have the 3D bounding box labels we require. Additionally, there does not seem to exist an annotated 3D dataset with image sequences or video frame data. Therefore, we chose to evaluate our model on our rendered data.

Image results from our model are shown in Figure3. Our model performed the worst on the TV object category. We believe this to be because the shape of the television monitors are fairly uneven cubes where one dimension is much smaller than the others. Any image sequence that does not completely display the object from all sides fails to give our model all the data required to properly predict the bounding box. For example, an image sequence like the one displayed in Figure1 for the television makes it seem as if the television is not as wide as it actually is. We believe training and testing with longer image sequences can give our model the information it needs for improved predictions for these less uniformly shaped models.

8 Future Work

Extending this method to perform 3D feature point prediction would be the logical next step. This would allow for the model to predict a point cloud rather than a simple bounding box. In this manner, 3D reconstruction of an object would be fast and straightforward. To add this, a bit of manual work is required to mark 3D feature points on the ShapeNet models so the render pipeline can output the proper labels.

Additionally, the model probably can be improved by training on larger datasets with longer image sequences. Because of the rendered data, acquiring a larger dataset is as simple as rendering a larger amount of data. It is required, however, to train and test on a more computationally powerful machine or multiple machines.

Finally, aspects of the dataset can be improved. The current cropping scheme is too randomized to mimic video data accurately and the shorter image sequences are not guaranteed to cover 90-degrees or more along a given camera angle axis. Because of this, currently some data does not give enough information for the model to correctly predict distance or the bounding box of uneven shapes. Additionally, some of the 3D models do not load in correctly in Blender. A simple weeding out of bad models or a fix in the Blender obj file loader can correct this issue.

References

ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., GOODFELLOW, I., HARP, A., IRVING, G., ISARD, M., JIA, Y., JOZEFOWICZ, R., KAISER, L., KUDLUR, M., LEVENBERG, J., MANÉ, D., MONGA, R., MOORE, S., MURRAY, D., OLAH, C., SCHUSTER, M., SHLENS, J., STEINER, B., SUTSKEVER, I., TALWAR, K., TUCKER, P., VANHOUCKE, V., VASUDEVAN, V., VIÉGAS, F., VINYALS, O., WARDEN, P., WATTENBERG, M., WICKE, M., YU, Y., AND ZHENG, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

CHANG, A. X., FUNKHOUSER, T., GUIBAS, L., HANRAHAN, P., HUANG, Q., LI, Z., SAVARESE, S., SAVVA, M., SONG, S., SU, H., XIAO, J., YI, L., AND YU, F. 2015. ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

REDMON, J., DIVVALA, S. K., GIRSHICK, R. B., AND FARHADI, A. 2015. You only look once: Unified, real-time object detection. *CoRR abs/1506.02640*.

SIMONYAN, K., AND ZISSERMAN, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556*.

SU, H., QI, C. R., LI, Y., AND GUIBAS, L. J. 2015. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *The IEEE International Conference on Computer Vision (ICCV)*.

XIANG, Y., MOTTAGHI, R., AND SAVARESE, S. 2014. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

XIANG, Y., KIM, W., CHEN, W., JI, J., CHOY, C., SU, H., MOTTAGHI, R., GUIBAS, L., AND SAVARESE, S. 2016. Objectnet3d: A large scale database for 3d object recognition. In *European Conference Computer Vision (ECCV)*.

XIAO, J., EHINGER, K. A., HAYS, J., TORRALBA, A., AND OLIVA, A. 2016. Sun database: Exploring a large collection of scene categories. *Int. J. Comput. Vision 119*, 1 (Aug.), 3–22.