Single and Multi-view 3D Object Reconstruction

Divyanshu Sharma
Northeastern University
Boston, MA
sharma.divya@northeastern.edu

Grania Machado
Northeastern University
Boston, MA
machado.g@northeastern.edu

Abstract

Inspired by methods that employ shape priors to achieve robust 3D reconstructions, in this project we implement a recurrent neural network architecture called the 3D Recurrent Reconstruction Neural Network. The network learns a mapping from images of objects to their underlying 3D shapes from a large collection of synthetic 3D models. The network takes in one or more images of an object instance from arbitrary viewpoints and outputs a reconstruction of the object in the form of a 3D occupancy voxel grid. The network does not require any image annotations or object class labels for training or testing. The reconstruction framework enables 3D reconstruction of objects in situations when traditional SFM/SLAM methods fail because of lack of texture and/or wide baseline.

1. Introduction

The problem of recovering a geometric representation of the 3D scene given a set of 2D projection images is classically defined as multi-view 3D reconstruction in computer vision. Traditional pipelines such as Structure from Motion (SfM) (Ozyesil et al., 2017)[1] and visual Simultaneous Localization and Mapping (vSLAM) (Cadena et al., 2016) [2] without deep networks typically rely on hand-crafted feature extraction and matching across multiple views to reconstruct the 3D model. However, if the viewpoints are separated by relatively large baselines, or the reflectance functions of the objects are non-Lambertian, and textures are not homogeneous, it can be extremely challenging for the feature matching approach due to significant changes of appearance. The reconstructed 3D shape is usually a sparse point cloud without geometric details.

The 3D shape of an object is a combination of countless physical elements that range in scale from gross structure and topology to minute textures endowed by the material of each surface. Intelligent systems require representations capable of modeling this complex shape efficiently, to perceive and interact with the physical world in detail. Numerous recent papers have begun to propose high resolution 3D shape representations with better scaling, such as those based on meshes, point clouds or octrees but these often require more difficult training procedures and customized network architectures. Our 3D shape model is motivated by a foundational concept in 3D perception: that of canonical views. The shape of a 3D object can be completely captured by a set of 2D images from multiple viewpoints. Deep learning approaches for 2D image recognition and generation scale easily to high resolutions. We aim at studying the problem of recovering an underlying 3D structure from a set of images. Based on the success of methods that employ shape priors to achieve 3D reconstructions, we're trying to implement a recurrent neural network architecture. The network should learn a mapping from images of objects to their underlying 3D shapes from a large collection of synthetic data.

Most of the state-of-the-art methods for 3D object reconstruction, however, are subject to several restrictions. Some restrictions are that objects must be observed from a dense number of views; or equivalently, views must have a relatively small baseline. This is an issue when users wish to reconstruct the object from just a handful of views or ideally just one view. Another issue is objects' appearances are expected to be non-reflective and mostly comprised of non-homogeneous textures i.e., smooth textures.

Unlike many previously proposed methods, our philosophy is centered around the availability of prior knowledge of the appearance and shape of the object. This circumvents the necessity of finding feature correspondences across different views of the object, and makes this method work with fewer images and fewer reflectance assumptions.

Shape priors are typically encoded in the form of 3D primitives or learnt from 3D CAD models. For this project, instead of trying to match a 3D shape prior to the observed object, we use deep convolutional neural networks to learn a mapping from observations to their underlying 3D shapes of objects from a large collection of training data. Given one or more images of an object instance from arbitrary viewpoints, it should output a reconstruction of the object in the form of a 3D occupancy grid. During training, the network does not need labels, segmentations, or keypoints.

For this project, we try to implement a GRU (Gated Recurrent Units) network which is a variant of a LSTM (Long Short-Term Memory) Recurrent Neural Network, so we can leverage the memory retention to incrementally refine the output reconstruction as more observations become available. The input to this network come from a 2D CNN that encodes each input image to low dimensional features. The network then uses 3D LSTM units that selectively update their states. This should help the network handle self-occlusions as it updates the cell states as more images are input. Finally, the states are decoded by a 3D Deconvolutional Neural Network that outputs the occupancy probability of each voxel cell in the output using a softmax function.

2. Background and Related Work

3D reconstruction has been an important topic for decades due to its extensive usefulness in augmented reality, virtual reality, medical imaging, rapid prototyping, engineering, and design. The objective of image-based 3D reconstruction is to gather the 3D geometrical structure and construction of items and scenes from collection of 2D pictures. We reviewed methods for multi-view 3D object reconstruction by space carving [4][5] and its probabilistic perspectives [6], that assume that the object is accurately segmented from the background, or the cameras are calibrated. However, this is not a practical assumption for many applications. While most research focus on multi-view geometry such as SFM and SLAM, ideally, one expects that 3D can be reconstructed from the abundant single-view images. Under this setting, however, the problem is ill-posed, and priors must be incorporated. Early work such as *ShapeFromX* made strong assumptions over the shape, or the environment lighting conditions. pioneered the use of learning-based approach for simple geometric structures. Coarse correspondences in an image collection can also be used for rough 3D shape estimation. As commodity 3D sensors become popular, RGBD database has been built and used to train learning-based systems.

We further reviewed approaches that use 3D shape priors for object reconstruction [7][8] and deep network implementations using novel network architectures [3][9]. The shape priors are commonly encoded as straightforward 3D natives as shown by early spearheading works or gained from rich vaults of 3D CAD models, by which the idea of fitting 3D models to pictures of countenances was investigated to a lot bigger degree. Recently, large-scale repositories of 3D CAD models, such as ShapeNet [4], have been introduced. They have great potential for 3D reconstruction tasks. For example, proposed to deform and reassemble existing shapes into a new model to fit the observed image. These systems rely on high-quality image-shape correspondence.

To circumvent issues related to large baselines or non-smooth surfaces, 3D volumetric reconstruction methods such as space carving and their probabilistic extensions have become popular. These methods, however, assume that the objects are accurately segmented from the background or that the cameras are calibrated, which is not the case in many applications.

A different philosophy is to assume that prior knowledge about the object appearance and shape is available. The benefit of using priors is that the ensuing reconstruction method is less reliant on finding accurate feature correspondences across views. Thus, shape prior-based methods can work with fewer images and with fewer assumptions on the object reflectance function. The shape priors are typically encoded in the form of simple 3D primitives as demonstrated by early pioneering works or learned from rich repositories of 3D CAD models, whereby the concept of fitting 3D models to images of faces was explored to a much larger extent. Sophisticated mathematical formulations have also been introduced to adapt 3D shape models to observations with different degrees of supervision and different regularization strategies. Building on this work with specific modifications like using deep convolutional neural networks on large datasets of objects to map its underlying 3D structure.

2.1. Scene Representation

The geometry of an object or scene can be represented in numerous ways; most multi-view algorithms use voxels, level-sets, polygon meshes, or depth maps. While some algorithms adopt a single representation, others employ different representations for various steps in the reconstruction pipeline. Some methods represent the scene as a set of depth maps, one for each input view. This multi-depth map representation avoids resampling the geometry on a 3D domain, and the 2D representation is convenient particularly for smaller datasets.

2.2. Photo-consistency measure

Numerous measures have been proposed for evaluating the visual compatibility of a reconstruction with a set of input images. Most of these measures operate by comparing pixels in one image to pixels in other images to see how well they correlate. For this reason, they are often called photo-consistency measures. The choice of measure is not necessarily intrinsic to a particular algorithm—it is often possible to take a measure from one method and substitute it in another. We categorize photo-consistency measures based on whether they are defined in scene space or image space.

Comparing the predicted and measured images yields a photo-consistency measure known as prediction error. While prediction error is conceptually very similar to scene space measures, an important difference is the domain of integration. Scene space error functions are integrated over a surface and thus often tend to prefer smaller surfaces, whereas prediction error is integrated over the set of images of a scene and thus ascribe more weight to parts of the scene that appear frequently or occupy a large image area.

2.3. Visibility model

Visibility models specify which views to consider when evaluating photo-consistency measures. Because scene visibility can change dramatically with viewpoint, almost all modern multi-view stereo algorithms account for occlusions in some way or another. Geometric techniques seek to explicitly model the image formation process and the shape of the scene to determine which scene structures are visible in which images. A common approach in surface evolution approaches is to use the current estimate of the geometry to predict visibility for every point on that surface. Visibility computations can be simplified by constraining the allowable distribution of camera viewpoints. If the scene lies outside the convex hull of the camera centers, the occlusion ordering of points in the scene is same for all cameras, enabling several more efficient algorithms.

2.4. Shape Priors

Photo-consistency measures alone are not always sufficient to recover precise geometry, particularly in low textured scene regions. It can therefore be helpful to impose shape priors that bias the reconstruction to have desired characteristics. While priors are essential for binocular stereo, they play a less important role in multi-view stereo where the constraints from many views are stronger.

Techniques that minimize scene-based photo consistency measures naturally seek minimal surfaces with small overall surface area. This bias is what enables many level-set algorithms to converge from a gross initial shape. Many methods based on voxel coloring and space carving instead prefer maximal surfaces. Since these methods operate by removing voxels only when they are not photo-consistent, they produce the largest photo-consistent scene reconstruction, known as the “photo hull.” Because they do not assume that the surface is smooth, these techniques are good at reconstructing high curvature or thin structures. However, the surface tends to bulge out in regions of low surface texture.

2.5. Initialization Requirements

In addition to a set of calibrated images, all 3D reconstruction algorithms assume or require as input some information about the geometric extent of the object or scene being reconstructed. Providing some constraints on scene geometry is in fact necessary to rule out trivial shapes, such as a different postcard placed in front of each camera lens. Many algorithms require only a rough bounding box or volume. Some algorithms require a foreground/background segmentation (i.e., silhouette) for each input image and reconstruct a visual hull that serves as an initial estimate of scene geometry. Image-space algorithms typically enforce constraints on the allowable range of disparity or depth values, thereby constraining scene geometry to lie within a near and far depth plane for each camera viewpoint.

2.6. Voxel Representation

For the output representations, we use a 3D space. For 3D representations, we can use different geometry types like point clouds and triangle meshes. However, we pick a voxel grid as out output instead. A voxel grid is a type of 3D geometry wherein a voxel can be thought of as a 3D counterpart to a pixel in 2D. We use an output of 32x32x32 voxel grid. This also allows the 3D LSTM network to localize its learning. During training, we use binary voxel data for ShapeNet shapes, and leverage helper utilities to read and write voxel models.

3. Datasets

Figure 1: ShapeNet contains multiple views of 3D object models

Datasets used -

- **ShapeNet** (https://shapenet.org/)
  ShapeNet dataset is a collection of 3D CAD models that are organized according to the WordNet hierarchy. We plan on using a subset of the ShapeNet dataset called ShapeNetCore, which covers 55 common object categories with about 51,300 unique 3D models.

- **PASCAL3D+** (https://cvgl.stanford.edu/projects/pascal3d.html)
  Dataset for 3D object detection and pose estimation, that contains 12 categories of rigid objects selected from the PASCAL VOC 2012 dataset. These objects are annotated with pose information (azimuth, elevation, and distance to camera). Pascal3D+ also adds pose annotated images of these 12 categories from the ImageNet dataset.

- **ModelNet40**
  (https://modelnet.cs.princeton.edu/)
  ModelNet40 consists of 12, 311 objects belonging to 40 categories. We can use the 10-class subset of the model that contains manually aligned and categorized CAD models.

4. Models

4.1. Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed or undirected graph along a temporal sequence. This allows it to exhibit temporally dynamic behavior.

Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. Recurrent neural networks are theoretically Turing complete and can run arbitrary programs to process arbitrary sequences of inputs.

Recurrent networks can have additional stored states, and the storage can be under direct control by the neural network. Such controlled states are referred to as gated state or gated memory and are part of long short-term memory networks (LSTMs) and gated recurrent units. This is also called Feedback Neural Network (FNN).

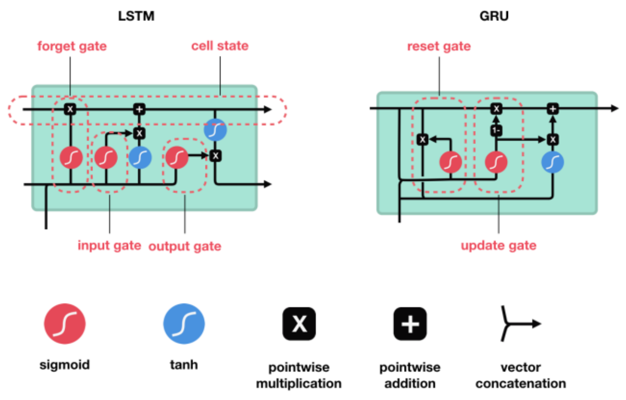


Figure 2: Different RNN hidden units (a) LSTM (b) GRU

4.1.1 Long Short-Term Memory Unit

One of the most successful implementations of the hidden states of an RNN is the Long Short-Term Memory (LSTM) unit. An LSTM unit explicitly controls the flow from input to output, allowing the network to overcome the vanishing gradient problem. Specifically, an LSTM unit consists of four components: memory units (a memory cell and a hidden state), and three gates which control the flow of information from the input to the hidden state (*input gate*), from the hidden state to the output (*output gate*), and from the previous hidden state to the current hidden state (*forget gate*). More formally, at time step *t* when a new input x_t is received, the operation of an LSTM unit can be expressed as:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$
$$s_t = f_t \times s_{t-1} + i_t \times tanh(W_s + U_s h_{t-1} + b_s)$$
$$h_t = o_t \times tanh(s_t)$$

Where i_t, f_t, o_t refer to the input, the output gate, and the forget gate respectively. *W*, *U* are matrices that transform the input and the hidden state, and *b* represents the biases.

4.1.2 Gated Recurrent Unit

In a GRU, an update gate controls both the input and forget gates, and a reset gate is applied before the nonlinear transformation. An advantage of GRU over a standard LSTM is that there are fewer computations. Formally,

$$u_t = \sigma(W_u \tau x_t + U_u * h_{t-1} + b_f)$$
$$r_t = \sigma(W_i \tau x_i + U_i * h_{t-1} + b_i)$$
$$h_t = (1 - u_t) \times h_{t-1} + u_t \times tanh(W_h x_t + U_h(r_t \times h_{t-1}) + b_h)$$

where u_t, r_t, h_t represent the update gate, the reset gate and the hidden state respectively. The other notations are the same as in the LSTM calculations.

4.2. Our model

We use a combination of a 2D Convolutional Neural Network, 3D LSTM and a 3D Deconvolutional Neural Network. Provided one or more images of an object from different viewpoints, the CNN first encodes each image input into low dimensional features. Then, given the encoded output, a set of 3D LSTM selectively update or retain their cell states. In the end, the 3D Deconvolutional Network decodes the hidden states of the LSTM units and generates a 3D voxel model.

The advantage of using an LSTM based network is that the network handles self-occlusions when multiple images are given to the network. It can selectively update the memory cells that correspond to the visible part of the object.

4.2.1 Encoder

We used ResNet18 as an encoder. The network has identity shortcut connections that improve the optimization process by providing skip connections across residual blocks, which help with the vanishing gradient problem. Input of each residual block is added to the output of the block after performing 1x1 convolutions to match the dimensions. A similar process is followed while backpropagation of gradients during training. This allows us to build deeper models without running into bottlenecks.

4.2.2 3D LSTM

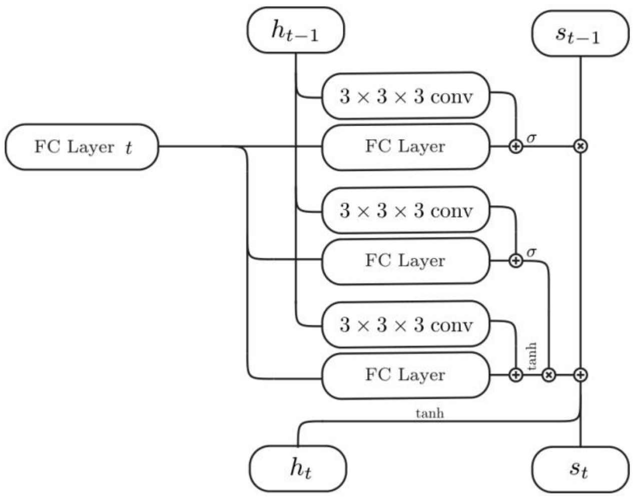


Figure 3: 3D-LSTM unit

We use convolutions to down sample the output of the encoder, along with 3D LSTM units that are spatially distributed such that each unit is responsible for reconstructing one part of the output. Inside the 3D grid there are $n \times n \times n$ 3D-LSTM units. All units have independent hidden states, that are indexed. For every LSTM unit,

$$f_t = \sigma\left(W_f \tau x_t + U_f * h_{t-1} + b_f\right)$$
$$i_t = \sigma(W_i \tau x_t + U_i * h_{t-1} + b_s)$$
$$s_t = f_t \times s_{t-1} + i_t \times tanh(W_s \tau x_t + U_s * h_{t-1} + b_s)$$
$$h_t = tanh(s_t)$$

This configuration forces a LSTM unit to handle the mismatch between a particular region of the predicted reconstruction and the ground truth model such that each unit learn to reconstruct one part of the voxel space instead of the entire space. This helps the network better handle occlusions as we can selectively update its prediction about the previously occluded part.

4.2.3 Decoder

We use a 3D Deconvolutional Neural Network to act as a decoder for the hidden state output by the 3D LSTM. It increases the hidden state resolution by applying 3D convolutions, ReLU and unpooling.

Like the encoder, we use a simple residual network with 4 residual connections, that feed into the final convolution layer. After the last layer, we convert the final activation to the occupancy probability of the voxel cell.

4.2.4 Loss

We used voxel-wise softmax loss, which is defined as the sum of voxel-wise cross entropy. For each voxel i, j, k in the output space, the loss can be defined as

$$L(x, y) = \sum_{i,j,k} log\left(p_{i,j,k}\right) + \left(1 - y_{i,j,k}\right) log\left(1 - p_{i,j,k}\right)$$

5. Implementation

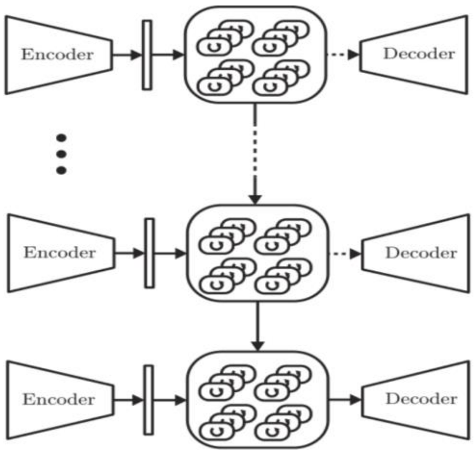


Figure 4: Network of 3D-LSTM units

We use a variable number of viewpoints for images of an object during training, to enable the network to perform both single and multi-image reconstructions. We used CAD model renderings, which were augmented with random images from the PASCAL3D+ dataset during training to generate input images and voxel occupancy maps.

We used an input image size of 127x127, and an output representation of 32x32x32. We used a batch size of 10 to fit the model on to the GPU. Our model has the advantage that we do not need to train and test per category for an object. The network trains and reconstructs without the knowledge of what category it belongs to.

We use PyTorch to implement our network, and leverage utilities for interacting with voxels written in Theano by the authors of 3DR2N2 network [9]. We used SGD with Adam for optimization. The network was trained with a batch size of 10 so it could fit on a mobile RTX3060 GPU. The training environment was created using CUDA running on a Docker container. The network was trained for 50,000 iterations, with a learning rate scheduler with a decay of 0.1. The steps were selected to be 1e-5 after 20,000 iterations, and 1e-6 after 60,000 iterations. IoU loss for our object examples was calculated using meshlab's intersection percentage.

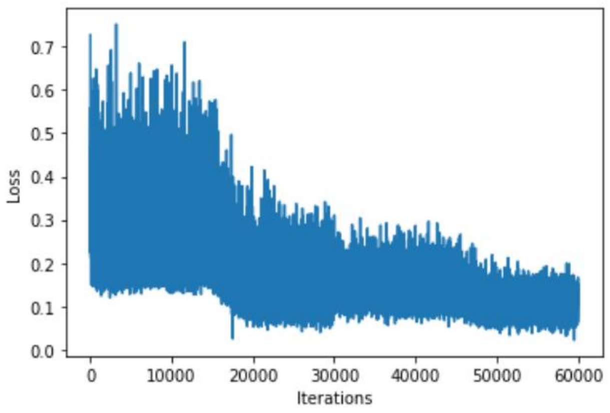


Figure 5 5: Network training

6. Experiments

6.1. Single View Reconstruction

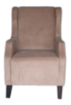
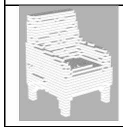
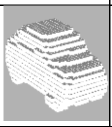
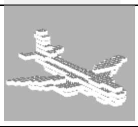
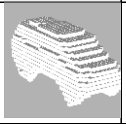
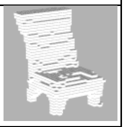
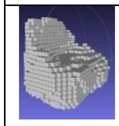
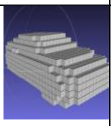
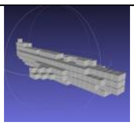
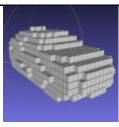
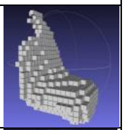
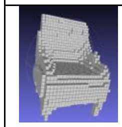
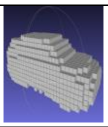
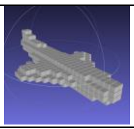
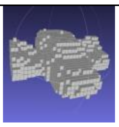
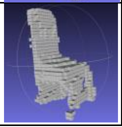
chair	car	aero	truck	tall_chair
				
				
				
				

Table 1: Single and Multi-view object reconstructions. The rows represent: Original object, 3DR2N2 outputs, single view output, multi-view output

We evaluated the performance of the model in single-view reconstruction using real world images. We used images from the PASCAL VOC dataset and its corresponding models from the PASCAL 3D+ dataset.

We trained a network on ShapeNet dataset and fine-tuned it using PASCAL 3D+. We see from the results that the outputs from our model are very close to the 3DR2N2[9] benchmark implementation. Our model has difficulties reconstructing minute details in objects. It also sometimes misrepresents an object based on a single viewpoint. Yet, our implementation provides competitive results. We use the voxel output's Jaccard index to quantify the percentage overlap between the target and out prediction.

Model	car	chair	aero	bike	tv
3DR2N2	0.699	0.280	0.544	0.499	0.574
Our implementation	0.548	0.250	0.368	0.349	0.517

6.2. Multi-view Reconstruction

We evaluated the network with the ShapeNet dataset. The testing set contained models from 13 major categories. We rendered 5 views of each model and optimized using softmax loss.

We can see that the model performs better reconstructions with increase in the number of views of the object. This is in line with the ability of the RNN to handle self-occlusions and additive information about parts of the output space. We also see that more views provide only marginal increase in IoU, as each new view provides less information since two random views are very likely to have partial overlap.

We also report the reconstruction IoU for 5 different categories in comparison to 3DR2N2 network. We can see that the network performs well on objects that have less shape variance, while objects with more intricacies perform worse. We can also see that the network improves with more views, as the initial reconstruction for the TV was a CRT, using just the front view, but it was corrected as more views became available.

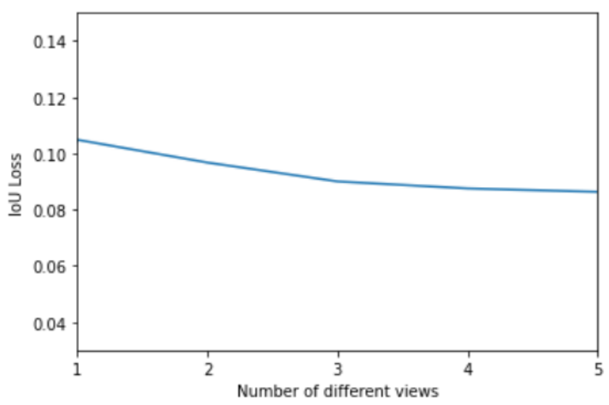


Figure 6: Decrease in loss with multiple views of the object

Views	Model	car	chair	aero	bench	tv
1	3DR2N2	0.798	0.466	0.513	0.421	0.468
1	Ours	0.758	0.422	0.482	0.382	0.375
2	3DR2N2	0.821	0.515	0.536	0.484	0.527
2	Ours	0.794	0.489	0.507	0.441	0.492
3	3DR2N2	0.829	0.533	0.549	0.502	0.545
3	Ours	0.806	0.497	0.518	0.476	0.519
4	3DR2N2	0.833	0.541	0.556	0.516	0.558
4	Ours	0.811	0.520	0.528	0.496	0.523
5	3DR2N2	0.836	0.550	0.561	0.527	0.565
5	Ours	0.821	0.538	0.53	0.507	0.537

Table 2: IoU metric for multi-view objects. We can see that as we add more views, the loss improvement rate decreases.

6.3. Real World Object Reconstruction

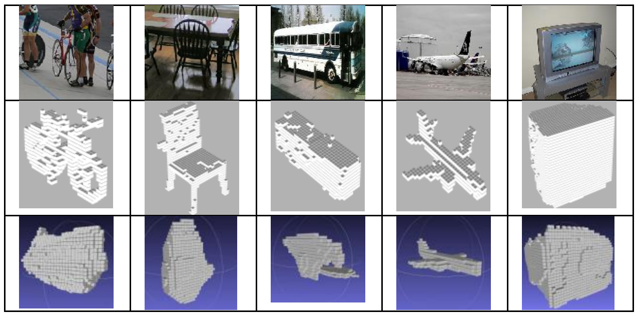


Table 3: Object reconstructions from real-life images

We tested our network on the Online Products Dataset. The results show that the network is capable of reconstructing real-world objects using only synthetic data from training samples. It also shows that the network improves the reconstruction with more views.

7. Conclusion

In this project, we implemented a network architecture that unifies single and multi-view 3D reconstruction into a single framework. We show that the ability of the PyTorch network implementation to reconstruct a model based on single view is comparable to the reconstructions of 3DR2N2. We further verify that the network performs multi-view reconstructions on the ShapeNet dataset, where the reconstruction loss decreases with an increase in the number of viewpoint images. Our network does not require a set number of images in order to produce a valid reconstruction, and overcomes the challenges faced by classical models, like inability to handle wide baseline viewpoints or self-occlusions.

8. Improvements/Future work

Recovering the 3D representation of an object from single-view or multi-view RGB images by deep neural networks has attracted increasing attention in the past few years. When given the same set of input images with different orders, RNN-based approaches are unable to produce consistent reconstruction results. Moreover, due to long-term memory loss, RNNs cannot fully exploit input images to refine reconstruction results. To solve these problems, we can further implement a framework for single-view and multi-view 3D reconstruction, named Pix2Vox [10]. By using a well-designed encoder-decoder, it generates a coarse 3D volume from each input image. Then, a context-aware fusion module is introduced to adaptively select high-quality reconstructions for each part from different coarse 3D volumes to obtain a fused 3D volume. Finally, a refiner further refines the fused 3D volume to generate the final output. This method would be faster than 3DR2N2 model in terms of backward inference time.

References

[1] Ozyesil O, Voroninski V, Basri R, Singer A, “A Survey of Structure from Motion” (2017) Acta Numerica 26:305–364

[2] Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, Reid ID, Leonard JJ, “Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age” 2016 IEEE Transactions on Robotics 32(6):1309–1332

[3] Bo Yang, Sen Wang, Andrew Markham, Niki Trigoni “Robust Attentional Aggregation of Deep Feature Sets for Multi-view 3D Reconstruction” 2018 CoRR, arXiv:1808.00758v2 [cs.CV]

[4] Kutulakos, K.N., Seitz, “A theory of shape by space carving” 2000 International Journal of Computer Vision 38(3):199–218

[5] Gregory G Slabaugh, W Bruce Culbertson, T.M., Stevens, M.R., Schafer, “Methods for volumetric reconstruction of visual scenes” International Journal of Computer Vision 2004 57(3):179–199

[6] Broadhurst, A., Drummond, T.W., Cipolla, “A probabilistic framework for space carving’ Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on. Volume 1. 388–393

[7] Dame, A., Prisacariu, V.A., Ren, C.Y., Reid, “Dense reconstruction using 3d object shape priors” Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference

[8] C. B. Choy, M. Stark, S. Corbett-Davies and S. Savarese, "Enriching object detection with 2D-3D registration and continuous viewpoint estimation," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 2512-2520, doi: 10.1109/CVPR.2015.7298866.

[9] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, Silvio Savarese, “3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction” arXiv:1604.00449v1 [cs.CV] 2 Apr 2016.

[10] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, Shengping Zhang, “Pix2Vox: Context-aware 3D Reconstruction from Single and Multi-view Images”, arXiv:1901.11153v2 [cs.CV] 29 Jul 2019.