3D Object Reconstruction

Rutvikk Kharod Harshith Nanda Sudharsan Sundaravenkataraman Bhargav Pulipati

Abstract

In this paper, we attempted to reproduce a simpler version of a recurrent neural network architecture called the 3D Recurrent Reconstruction Neural Network (3D-R2N2) [1]. We also attempted custom data collection by automating the process of getting 2D images from 3D models. Our approach was to apply a simpler version of the architecture on multiple datasets and compare the model's performance. Ultimately we demonstrated that even a crude re-implementation of the same model in PyTorch with roughly the same layers and pooling techniques can result in a reasonably high performance. This suggests that the 3D-R2N2 architecture can be considered as a standardized approach to 3D modeling using 2D images without the need for image annotations and object class labeling.

1. Introduction

The ability to quickly and automatically create 3D objects has changed the game in a variety of fields, including architecture, medicine, and visualization. For example, in the domain of medicine 3D modeling can help produce patient-specific prosthetics or implants resulting in more personalized treatment. In architecture, it can provide ease in regards to modifying a particular model or tailoring it to certain environmental constraints. This contribution is partly attributable to the fact that 3D printing has become widely available and 3D acquisition techniques are precise and effective.

In the domain of 3D modeling, many approaches have been undertaken. These approaches pertain to addressing a variety of challenges. Some of those early challenges included dealing with reflective surfaces, camera calibration-related challenges, and occlusions in feature correspondences. These led to further developments where certain assumptions were made. Namely, that cameras were calibrated or that objects were sufficiently segmented from the environment. Still more assumptions were made where object priors (information about the object's shape and appearance) was used to circumvent the need to perform feature correspondence across a variety of images. A few perspectives were sufficient. This paper developed an architecture and did not utilize the assumption that the object was sufficiently segmented from the background. It additionally did not require object class labels or image annotations (segmentation, keypoints etc.) This was due to the fact that this specific technique was independent of the establishment of any feature correspondences across multiple perspectives. The techniques leveraged in the original paper presenting the 3D-R2N2 model were propelled by the successes of LSTM networks and Convolutional Neural Networks in the domain of single-view 3D reconstruction [3] [5], i.e. taking a single view of an object and being able to effectively reproduce a 3D model.

2. Related Work

The 3D Recurrent Reconstruction Neural Network, also known as (3D-R2N2), is built on a large body of research in computer vision and deep learning. It's an amalgamation of multiple technologies, from 2D CNN encoding layers, 3D LSTM convolutional layers, and a whole new architecture.

2.1. 3D Convolutional Long Short-Term Memory Unit

Traditional LSTMs have been widely used for sequence modeling tasks due to their ability to capture long-term dependencies within sequential data. However, classic LSTMs operate on vectors, lacking explicit mechanisms for handling spatial information. The following equations describe a traditional LSTM:

1. $i_t = \sigma(W_i T(x_t) + U_i * h_{t-1} + b_i)$
2. $f_t = \sigma(W_f T(x_t) + U_f * h_{t-1} + b_f)$
3. $o_t = \sigma(W_o T(x_t) + U_o * h_{t-1} + b_o)$
4. $s_t = f_t \odot s_{t-1} + i_t \odot \tanh(W_s T(x_t) + U_s * h_{t-1} + b_s)$
5. $h_t = o_t \odot \tanh(s_t)$

The introduction of 3D convolutions to LSTMs paved the way for processing spatiotemporal data, such as video

sequences. In these applications, 3D LSTMs analyze a sequence of 3D image frames, capturing both the temporal evolution of the scene and the spatial relationships within each frame. This has led to advancements in tasks like action recognition, video prediction, and motion tracking.

Despite their success in handling spatiotemporal data, prior applications of 3D LSTMs often treat each spatial location independently, without explicit modeling of interactions between neighboring locations. This can limit their ability to capture local geometric patterns and context, which are crucial for accurate 3D reconstruction. Additionally, existing 3D LSTMs typically focus on predicting the next frame in a sequence or classifying the entire sequence. They are not specifically designed for the task of 3D reconstruction, where the goal is to progressively build a 3D representation of an object from multiple viewpoints.

The 3D-LSTM module in 3D-R2N2 improves upon pre-existing LSTMs by introducing several key innovations that address the limitations of prior work and tailor the architecture specifically for 3D object reconstruction.
First, the 3D-LSTMs in this implementation are spatially structured, meaning the 3D-LSTM units are arranged in a 3D grid structure and connected to their spatial neighbors through 3D convolutions, with each unit responsible for reconstructing a particular region of the 3D space. This creates a sense of locality and allows the network to focus on refining specific parts of the object based on the information from different viewpoints. The connection via 3D convolutions enables them to share information and learn local geometric patterns, leading to more accurate and contextually aware reconstructions. 3D-R2N2 also improves on the gating mechanism in its LSTMs. The gating mechanism of the LSTM allows the network to selectively update its internal state based on the input information. This is crucial for handling occlusions, as the network can retain information about previously observed object parts while incorporating new details revealed in subsequent views. Lastly, instead of predicting a single output, the 3D-LSTM in 3D-R2N2 progressively refines its 3D representation of the object as it processes a sequence of input images. This allows the network to take advantage of information from multiple viewpoints to refine its reconstruction of the object.

2.2. 2D-CNN Encoder

The 3D Recurrent Reconstruction Neural Network (3D-R2N2) leverages the power of 2D Convolutional Neural Networks (CNNs) to encode visual information from input images. While the use of CNNs for image feature extraction is well-established in computer vision, 3D-R2N2 introduces specific improvements that distinguish its approach from prior work and contribute to its success in 3D object reconstruction. The success of CNNs in 2D image analysis naturally led to their exploration for 3D understanding tasks. Early attempts involved using CNNs to extract features from 2D images and then employing additional modules, such as recurrent networks or fully connected layers, to infer 3D information. However, these approaches often struggled to capture the complex spatial relationships inherent in 3D data. The introduction of 3D CNNs, which extend the principles of 2D convolutions to 3D data, enabled more direct processing of 3D information, such as point clouds or voxel grids. This led to advancements in tasks like 3D object classification, segmentation, and pose estimation.

While 3D CNNs offer a direct way to process 3D data, 3D-R2N2 opts for a different approach, employing 2D CNNs as encoders for extracting features from individual input images for the sake of computational efficiency, which is based on this paper [4]. 2D CNNs generally require less computational resources than their 3D counterparts, making the training and inference process more efficient. This is particularly important for 3D-R2N2, which deals with sequences of images from multiple viewpoints. By processing each input image independently, the 2D CNNs can capture view-specific details and nuances that might be lost when directly processing 3D data. This information is then effectively integrated by the subsequent 3D-LSTM module to construct a comprehensive 3D representation

This paper explores two main variations of 2D CNN encoders: Standard Feed-Forward CNN and the Deep Residual CNN. The Standard Feed-Forward CNN acts as the primary encoder, taking an input image and progressively transforming it into a lower-dimensional feature vector. This vector captures the essence of the object's appearance and shape from that specific viewpoint. The Deep Residual CNN, while also an encoder like the standard CNN, also serves another purpose, which is to mitigate the vanishing gradient problems in deeper networks, allowing the network to be larger and learn more complex features.

2.3. 3D Deconvolutional Neural Network

This paper employs a 3D Deconvolutional Neural Network (3D DCNN) as its decoder, responsible for transforming the encoded representation of an object into a 3D voxel-based reconstruction. While 3D DCNNs have been explored in various prior works, 3D-R2N2 introduces specific design choices and adaptations that enhance their effectiveness for the task of 3D object reconstruction. 3D DCNNs are often used in conjunction with 3D CNNs,

which extract features from 3D input data like point clouds or voxel grids, and are most often used for 3D object generation, 3D shape completion, and Medical Image Segmentation. However, traditional 3D DCNNS suffer from major limitations: the loss of fine details due to repreated upsampling and deconvolution of the data, limited contextual reasoning that leads to inconsistencies or inaccuracies in the reconstruction, and the relative lack of sequential integration with LSTMs, resulting in suboptimal reconstructions.

3D-R2N2 addresses these limitations through specific design choices and adaptations in its 3D DCNN decoder, based on this paper [2]. For one, Instead of simple interpolation techniques for upsampling, the network employs 3D unpooling, which restores spatial resolution while preserving the locations of activated features from the previous layers. This helps to retain fine details and improve the overall accuracy of the reconstruction. Additionally, the 3D DCNN decoder takes the hidden state from the 3D-LSTM module as its input. This allows it to benefit from the contextual information and sequential reasoning capabilities of the 3D-LSTM, leading to more consistent and accurate reconstructions. The decoder also implements the Deep Residual architecture, which allows for deeper networks with improved feature propagation and potentially better reconstruction quality.

In all, the 3D DCNN decoder plays a crucial role in the final stage of the 3D reconstruction process, given it is the step that takes the output of the 3D-LSTM and turns it into the object itself. It receives the hidden state from the 3D-LSTM module, which represents a compressed and encoded representation of the object learned from the sequence of input images. Through a series of 3D deconvolutional layers and unpooling operations, the network gradually upsamples the feature maps, restoring spatial resolution and refining the details of the reconstruction. The final layer of the decoder produces a 3D voxel grid with each voxel representing the probability of occupancy. This probabilistic output allows for a more nuanced and informative representation of the object's shape.

2.4. 3D Voxel-wise Softmax

This paper employs a 3D Voxel-wise Softmax loss function to guide its learning process and evaluate the quality of its 3D reconstructions. This choice of loss function aligns well with the voxel-based representation of 3D objects and offers several advantages over alternative loss functions used in prior works.

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

In the past, L1/L2, Cross-Entropy, and Chamfer Distance were most commonly used for 3D deep learning tasks. However, each of those loss functions came with their own set of drawbacks, such as L1/L2's sensitivity to outliers & inability to effectively capture the spatial relationships between voxels, Cross-Entropy's treatment of voxel's as independent, which neglects spatial context, and Chamfer distance generally being more suitable for point generation tasks over voxel-based tasks.

Compared to the previous loss functions, 3D Voxel-wise Softmax provides several distinct advantages that make it uniquely suited to this task. In addition to the benefit of Softmax providing a probability distribution, which serves as a better representation of the voxel grid, the 3D Voxel-wise Softmax loss function takes into account the spatial relationships between neighboring voxels, encouraging the network to learn consistent and spatially coherent reconstructions. The probabilistic nature of the loss function makes it more robust to noise and ambiguity in the input data or ground truth labels, leading to more stable training and improved generalization.

2.5. Usage of SLAM

When it comes to the specific task of 3D object reconstruction, SLAM presents certain limitations that necessitate a departure from traditional approaches. The 3D-R2N2 architecture, proposed for robust 3D object reconstruction, marks a significant shift away from SLAM-based methods, offering a more versatile and effective solution for capturing the 3D structure of objects from visual data.

SLAM algorithms heavily rely on establishing accurate feature correspondences between consecutive frames or observations. As discussed earlier, this process can be susceptible to challenges such as reflective surfaces, textureless regions, occlusions, and large baselines. These factors can lead to errors in the estimated camera pose and the constructed map, ultimately affecting the quality of the 3D reconstruction.

The 3D-R2N2 architecture offers a departure from SLAM-based methods, leveraging deep learning for direct 3D object reconstruction from image data. By learning a mapping from images to 3D voxel representations, 3D-R2N2 bypasses the need for explicit feature correspondence, overcoming the limitations of traditional approaches. The network's convolutional neural networks

(CNNs) extract meaningful features from individual images, while a novel 3D convolutional LSTM (3D-LSTM) module integrates information from multiple viewpoints and progressively refines the 3D reconstruction.

This deep learning approach allows 3D-R2N2 to handle challenging scenarios effectively. The network learns to extract features robust to variations in appearance, integrates information from diverse viewpoints to handle occlusions and large baselines, and is less susceptible to camera calibration errors. Moreover, offline processing enables 3D-R2N2 to utilize information from multiple images to generate more accurate and detailed 3D reconstructions compared to online SLAM algorithms.

3. Method

Our method consisted of two key steps. We wanted to handle data collection uniquely, while still leveraging the ShapeNet dataset as well as the ObjaverseXL dataset. We wanted to try to generate voxel information from a 3D model for different categories of objects from the ObjaverseXL dataset as well and try to automate the process of extracting 2D images from those 3D models.

The second step was to reproduce the model using a simpler Python framework like Pytorch. We were successfully able to do so. In our code we utilized the default Pytorch layers while following the specific architecture detailed in the paper. Specifically for the encoder we utilized the provided PyTorch 2D Convolutional, 2D Pooling, and LeakyReLu layers. For the 3D LSTM layer we created our own custom layer based off of what the paper detailed and tried to emulate what was done in the linked repository. For the Decoder we utilized the provided PyTorch 3D Convolutional, LogSoftMax and LeakyReLu layers. We also used PyTorch to upsample the input tensor input by a factor of 2 along each dimension (D, H, W) using nearest-neighbor interpolation.

4. Experiments

4.1. Limitations

Before we attempted to implement the architecture, we attempted data collection unsuccessfully. As mentioned in the methods section we attempted to leverage Python libraries such as pyglet, matplotlib and various others in order to visualize some new 3D models and then automate the process of producing 2D images from those models.

For this project, we implemented the 3D-R2N2 neural network architecture outlined in the paper from scratch in PyTorch. However, given the massive size of the neural network, consisting of twenty-six primary encoding, decoding, and convolution layers and nine pooling layers, the neural network takes an exorbitant amount of time to train and generate new images. As a result, the experiments we could perform were rather limited. Reducing the size of the neural network resulted in huge losses in accuracy which simply were not acceptable.

Examples of this include Figure 1 and 2 and Figures 7 and 8. Reducing the size of the neural network led to inaccurate reconstructions and omissions of features such as wheels and windows.

5. Conclusion

In this study, we explored the feasibility of a simplified adaptation of the 3D Recurrent Reconstruction Neural Network (3D-R2N2) by employing a PyTorch-based implementation. Our approach involved an unsuccessful attempt at automating the collection of 2D images from 3D models and simplifying the network architecture while maintaining essential elements of the original design, such as convolutional layers and pooling operations.

These results not only affirm the efficiency of the simplified model but also highlight its practicality for applications requiring 3D reconstructions from 2D inputs without extensive annotations or object class labeling. The fidelity of our model's performance to that of the original implementation underscores the 3D-R2N2 architecture's potential as a standardized approach in the field of computer vision, particularly for tasks involving 3D object reconstruction from multi-view 2D images.

Further research could focus on optimizing the network's architecture to enhance its computational efficiency or expanding its applicability to a broader range of objects and textures. Particularly in augmenting the model's capability of being able to differentiate between different intra-categorical objects. The 3D models being produced are bare-bones. Our findings pave the way for broader adoption and adaptation of the 3D-R2N2 model, offering significant implications for advancements in areas such as virtual reality, augmented reality, and automated system design where rapid and reliable 3D modeling is crucial.

References

[1] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction, 2016. 1

[2] Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, and Thomas Brox. Learning to generate chairs, tables and cars with convolutional networks, 2017. 3

[3] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network, 2014. 1

Figure 1. Input image for a bus

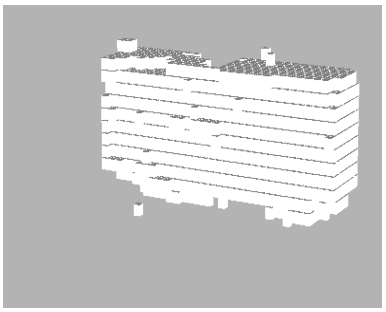

Figure 2. Model Reconstruction of a bus

Figure 3. Input image for a chair

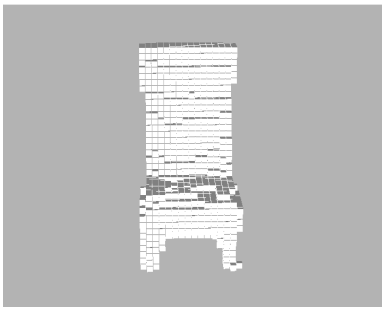

Figure 4. Model Reconstruction of a chair

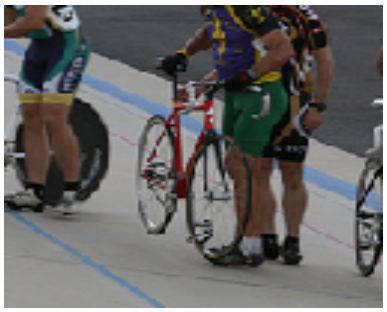

Figure 5. Input image for a bicycle

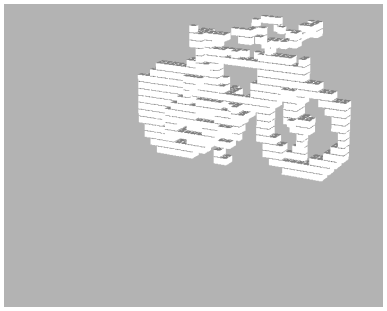

Figure 6. Model Reconstruction of a bicycle

Figure 7. Lower Accuracy Model Reconstruction of a chair

Figure 8. Lower Accuracy Model Reconstruction of a chair

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[5] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image,

2014. 1