Deep Rendering Graphics Pipeline

Mark Wesley Harris
University of Colorado, Colorado Springs
1420 Austin Bluffs Pkwy
80918 Colorado Springs, Colorado
wharris2@uccs.edu

Sudhanshu Semwal
University of Colorado, Colorado Springs
1420 Austin Bluffs Pkwy
80918 Colorado Springs, Colorado
ssemwal@uccs.edu

ABSTRACT

The graphics rendering pipeline is key to generating realistic images, and is a vital process of computational design, modeling, games, and animation. Perhaps the largest limiting factor of rendering is time; the processing required for each pixel inevitably slows down rendering and produces a bottleneck which limits the speed and potential of the rendering pipeline. We applied deep generative networks to the complex problem of rendering an animated 3D scene. Novel datasets of annotated image blocks were used to train an existing attentional generative adversarial network to output renders of a 3D environment. The annotated Caltech-UCSD Birds-200-2011 dataset served as a baseline for comparison of loss and image quality. While our work does not yet generate production quality renders, we show how our method of using existing machine learning architectures and novel text and image processing has the potential to produce a functioning deep rendering framework.

Keywords

Graphics pipeline, rendering technologies, machine learning, image processing, generative adversarial networks, text-to-image, semantic data processing

1 INTRODUCTION

Rendering a 3D environment often requires excessive amounts of time and computational resources. The cost of high quality rendering is multiplied by variable circumstances during production, such as changes to the scene, cast, or script. A determining factor in the speed of rendering is undoubtedly the amount of processing which occurs in the rendering pipeline. We propose the application of text-to-image deep learning networks to the rendering of a 3D environment, to help subvert the costs of traditional rendering processes. We trained the Attentional Generative Adversarial Network [Xu01a] using textual descriptions for small sections of pre-rendered frames. We then generated renders of the environment for a given frame by stitching together outputs generated from the frame's corresponding attributes.

We show how current research may be used in new ways to inject more relevance into generated images, and ultimately produce reasonable renderings of unknown scene data. Although our method does not yet produce images of high enough quality to replace current rendering technologies, we provide a proof of concept to support future advancements for deep learning approaches to the graphics pipeline.

2 BACKGROUND

Deep text-to-image synthesis was made possible through advancements in computer vision and machine learning. Although neural networks have been applied to animation and frame prediction, none so far have attempted to circumvent the rendering process as a whole [Sem01a]. We propose the application of deep generative networks to the rendering of a 3D scene using textual data alone. Two models we found crucial to solving this problem were generative adversarial learning, and attention mechanisms. We utilized an architecture combining both of these concepts, the Attentional Generative Adversarial Network developed by Microsoft Research [Xu01a], in the creation of a deep rendering framework capable of rendering images given textual descriptors of a 3D scene.

2.1 Generative Adversarial Networks

The Generative Adversarial Network (GAN) was first proposed as a method of producing realistic images from random noise [Goo01a]. The GAN is made up of two sub-architectures which are trained in tandem: a generator, *G*, and a discriminator, *D*. The generator is

trained to progressively synthesize images that closely resemble a target, while the discriminator is trained to determine if a given image is real or fake.

A non-conditional GAN model functions as a random image generator, where generated outputs share global and local qualities of images from the training dataset. The advent of the conditional GAN opened new avenues for research in adversarial learning, and has since led to applications in image-to-image and text-to-image translation.

2.2 Attention Mechanisms

Another deep learning concept applicable to text-to-image generation is called Attention. Attention is a type of sequence-to-sequence learning which uses attention functions to reference past data during each iteration of training. An attention function is a mapping of a query and a set of key-value pairs to an output. Attention may be accomplished using an encoder-decoder structure, where the encoder is trained to map feature space $(x_1,\ldots,x_n)$ to a continuous representation $\mathbf{z} = (z_1,\ldots,z_n)$, and the decoder is trained to transform $\mathbf{z}$ into an output sequence $(y_1,\ldots,y_m)$ [Vas01a]. Applied to text-to-image generation, feature space would consist of all input words and the output sequence would consist of 3-channel pixel data. The self-referential nature of attention enables the learning of complex relationships in long data sequences.

3 RELATED WORK

While they perform excellently as random generative models, GANs tend to be unstable, and their outputs often do not fully reflect the diversity in the training set. Over-training of a GAN network may result in preference of a small set of images rather than learning more detailed representations from the dataset. Researchers also found it difficult to train GANs to learn global structures; semantic meaning such as pose, composition, and color are impossible to control without modifying the GAN model. Higher quality images with more stability and semantic relevance were obtained by seeding the model with non-random data – such as a low-quality image or textual description – to generate from [Par01a]. Others found that variation inference and other types of embedding increased the quality of generated images [Luc01a]. These discoveries led to the creation of many variant GAN architectures, such as the Deep Convolutional GAN (DCGAN) [Rad01a].

The DCGAN was first proposed as a way to bridge the capabilities of unsupervised and supervised learning through adversarial training. The model architecture has a similar structure to the original GAN model, but uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively [Rad01a]. The addition of convolutional layers increases abstraction and allows the DCGAN to recognize more complex relationships. The text-conditional convolutional GAN architecture was built off of the successes from the DCGAN, and was one of the first GAN architectures to produce results in adversarial text-to-image synthesis [Ree01a].

Text-to-image generation was notably improved by Microsoft Research with the advent of their Attentional GAN (AttnGAN) model [Xu01a]. The AttnGAN architecture combines inherent inference obtained by the generator and discriminator within the GAN framework with the improvements in dependency correlation and stability from attention mechanisms. Instead of encoding the entire caption as a single input vector, each word in the sequence is allowed an opportunity to claim regional importance for the output image. This ability provides better recognition of conditional sub-regions, which improves overall semantic relevance and coherence of the output image. AttnGAN was partially based on the work of the StackGAN++ architecture, which found a multi-stage approach was necessary for stabilizing sensitivities in training GAN models [Zha01a].

Preceding adversarial training, the AttnGAN architecture loads all textual attributes in the text dataset, and creates a one-hot encoding representation of the data. A one-hot encoding assigns each symbol a bit value of a binary string of size n, where n is the size of the language of all attributes. This creates an easy way for the network to recognize words and ensures a minimal amount of resources are allocated to tracking them. The AttnGAN requires training two models: pre-training for the text and image encoders, and generative training for image synthesis [Xu01a]. Pre-training is contained in the Deep Attentional Multimodal Similarity Model (DAMSM), which initializes the text and image encoders preceding any adversarial training. This step is essential to providing the stability necessary for conditional image generation. Xu et al. used an LSTM model for their text encoder, and Google's Inception-v3 CNN as the image encoder. After pre-training is finished, the encoder paths are used for adversarially training text-to-image generation.

Another architecture was created from the advancements of AttnGAN, called LEarn, Imagine and CreAte GAN (LeicaGAN) [Qia01a]. Similar to other attention-based GANs, the LeicaGAN architecture is comprised of a text-embedding phase followed by coarse-to-fine image generation. The main improvement of LeicaGAN is a result of their use of textual-visual co-embedding (TVE) and multiple priors aggregation (MPA) to further extract semantic meaning from input images.

Finally, we review work from the related field of frame prediction. Mathieu et al. were of the first to apply adversarial learning to predict images in a seqence of

frames. They developed a recursive, multi-scale GAN framework to predict small portions of an input video sequence. Their mult-scale approach provided them the benefits of convolutional layers without lowering the quality of their output images [Mat01a]. The Spatial Transformer Predictor [Fin01a] and Geometry-Based Next Frame Prediction [Mah01a] models deployed a series of convolutional Long Short-Term Memory units to generate the next frame in a video sequence. The former predicted a series of transformations which were masked and then applied to source frames, while the latter predicted a depth map which was then converted into a final image. Liu et al. trained a convolutional encoder-decoder network to predict images between two frames. Their architecture, which they called Deep Voxel Flow [Liu01a], was trained to generate an in-between frame by using voxels borrowed from the two frames adjacent to it. Their approach notably did not require pre-processing, as was used in the prediction methods discussed previously.

4 APPROACH

We propose the creation of a machine learning framework to render frames of a 3D environment given semantic data input as text, a concept we refer to herein as a *deep rendering graphics pipeline*. We chose to use the AttnGAN as the generative architecture, to provide a well-defined baseline for current and future work. Inputs of the architecture were any number of textual attributes containing contextual meaning for the objects which were to be rendered. Once trained, the rendering system was then used to preview new views of the scene without re-rendering using the animation software or re-training the AttnGAN architecture.

The *deep rendering graphics pipeline*, as we define it, requires non-image seed data as input to produce a rendered output image. We note that frame prediction and deep rendering overlap in expectations for final outputs, however given the divergence between text-to-image and image-to-image networks, we classify the two as separate deep learning problem spaces. Although we require both text and image data during training, our final outputs were constructed from text attributes alone. This serves as a function of rendering directly from extracted scene data and does not involve the manipulation or processing of image inputs to generate the final frame, as is done in frame prediction.

4.1 Process

To measure the capabilities of our method, we created a 3D scene with many moving parts and complex lighting interactions. We applied RenderMan [Chr01a] materials with varying properties, including blue glass, soap bubble, gold, obsidian, thick glass, thin glass, beer glass, and murky water. Lighting was applied via a spherical HDRI image of a bright outdoor scene. We animated 4 axes of rotation and 1 telescopic projection over the course of 120 frames, and rendered the animation using RenderMan. These frames were then processed further to create sets of images to be used as training data.

Our datasets of training images were generated by extracting small portions of each frame, which we refer to as "frameblocks". It was clear that including insignificant or redundant information could result in over-training, so we ensured during extraction that elements of the dataset contained minimal similarity to one another. To generate frameblocks for frame F_i, we compared the visual changes between the two adjacent frames, F_{i-1} and F_{i+1}, using a bit-wise XOR and vector sum capped at 255. The result was a grey-scale image which highlighted changed components between frames F_{i-1} and F_{i+1}. While processing each frameblock in the frame, we compared the total amount of change in the block to the sum of all changes in the frame. If the value surpassed the 1.0 percentile of total change, it was included in the dataset, otherwise it was stored in a cache to be compounded in future calculations for the same frameblock. Thus, even small changes over many frames were represented in the image dataset over the course of image pre-processing.

We then extracted corresponding text attributes describing the objects present in each frameblock of the image dataset. Types of attributes we experimented with included screen distance, frame and block index, and vertex, edge, normal, and orientation information. In experimenting with what attributes produced the best outputs, we discovered the AttnGAN model found difficulty in recognizing relationships inherent to numerical data. We concluded that, due to the use of one-hot enocding, the AttnGAN network more easily recognized unique identifiers over numerical values. This is also supported by similar studies of other learning models [Zha02a].

To accommodate the loss in recognition, we attempted to more precisely describe an object's characteristics in relation to its rendered pixels. Each frameblock was divided into $m \times n$ regions, and each region was assigned a bit position. We used these bit positions as flags for marking the area a given 3D object was present inside of. Figure 1 demonstrates this effect for finding the precision attribute of an object in the lower right corner of the sample frameblock. Using the 3×3 grid shown, we see the object resides inside of regions 0x020, 0x080, and 0x100. The logical "OR" operator ($\oplus$) was used to combine these values and create the final precision attribute. Thus, for our example, the precision value was calculated from $0x020 \oplus 0x080 \oplus 0x100 = 0x416$, and resulted in the output attribute of "v416". This process was repeated for all objects present inside of each

frameblock. As we expected, the increase in precision improved recognition and output quality. We experimented with the resolution of precision, from 2×2 subdivisions to 8×8. In general, results did not improve visually past 4×4, so we decided to generate our final results with $m = 4$ and $n = 4$, for a total of 65,536 possible resolution cases.

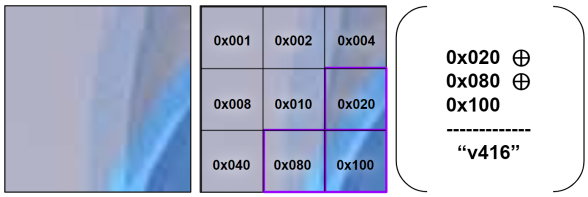


Figure 1: Example of the precision attribute for an object in the bottom right corner of the sample frameblock.

While the precision attribute was able to mask the presence of an object for a given frameblock, it was not able to provide information on what part of the object was in view. We decided to again employ unique identifiers, this time for indexing groups of faces visible in the frameblock. For each object, we assigned an index to the group of face indices exactly containing visible faces. A running list of all face groups was maintained, so that if a given group of faces was seen again, the same identifier was used. This strategy provided spatial relevance while maintaining uniqueness, and was found to be paramount to the visual quality of our output images.

We obtained the best results by prepending the mesh name to each attribute. Recognition also improved with the addition of identifying words; we used "and" to separate attributes connected to the same object, and "also" to identify the end of one attribute group and the start of the next. A sample from the 8×8 dimension dataset is shown in Figure 2, outlining the location of extraction from the source frame in white. The attributes we used included distance to the screen (d), face group identifier (i), precision value (v), and concatenated Euler rotation (r). The text required for this 8×8 pixel sample describing 4 objects was approximately 400 characters in length, while simpler samples with only one object were represented in as few as 30 characters in total.

4.2 Training

To begin training, we generated text attributes for all elements of the corresponding image dataset, as well as for any test frames used in evaluating our framework. We performed two experiments, one using 2 frames of known input, and another using 30 frames. For these experiments, we selected every other frame to be used as a test case. Once the AttnGAN architecture was fully trained, the model was used to generate the first four frames of animation, 2 of which were novel views.

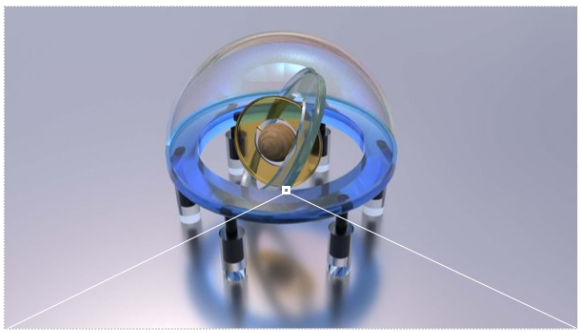


Figure 2: Example of attributes for an extracted 8×8 pixel frameblock of a known frame.

The final stages of our process involved frame generation and analysis. To create a final synthetic frame, we fed in the semantic attributes for all frameblocks of the frame, and combined the generated images in their corresponding block locations.

We experimented with varying frameblock sizes, to gain insight into the efficacy of our text attributes. We generated and tested with frameblocks of dimensions $2^k \times 2^k$ pixels, for $k = 2, 3, \ldots, 6$. We found that using large values of k resulted in smaller datasets which took less time to train, but also generated more visual anomalies and less coherence. Smaller block sizes resulted in very slowly training architectures with higher image quality. Table 1 shows a synopsis of statistics for various frameblock sizes, and Table 2 shows our generated dataset size compared to the total size if all frameblocks were included.

	64	**32**	**16**	**8**
PT (hrs)	0.364	1.216	8.477	17.726
GT (hrs)	1.109	2.727	17.617	34.874
RT (hrs)	0.050	0.167	0.605	2.289

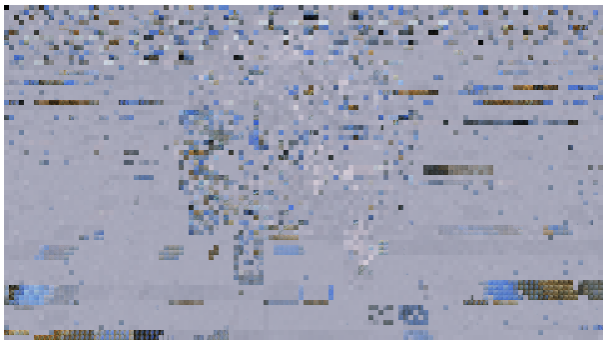
Table 1: Training and evaluation times in hours per dimension of frameblock. **PT** represents pre-training time, **GT** represents generation training time, and **RT** represents final render time for a single frame.

	64	**32**	**16**	**8**
Dataset Size	4,389	17,862	73,741	296,526
Dataset Max	14,400	118,800	241,200	9,720,000

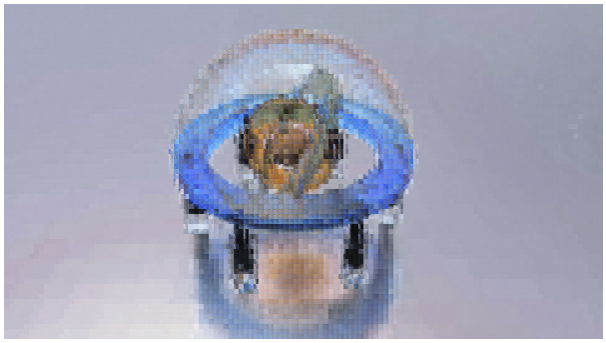
Table 2: Total number of image-text pairs for each dataset compared to the same dataset generated without our selective pre-processing.

In using our method of frameblock selection, we reduced dataset size to 30% of the maximum on

average, and achieved an improvement of 112.8% for SSIM and reduction of 143% in MSE. The model trained on all frameblocks in the frame – in this case a total of 241,200 images – over-trained on the background attributes, which were more abundant per frame than those of any other object in the scene. Our pre-processing phase yielded a dataset of 73,741 unique images and attributes, which resulted in better training and more visual cohesiveness, as seen by the improvements between Figures 3a and 3b. This shows the benefits of our novel use of frameblocks in adversarial text-to-image synthesis.



(a) The frame generated with all attributes present in the frame, including redundant information.



(b) The frame generated using our method of pre-processing.

Figure 3: Visual improvement accomplished using our method of frameblock pre-processing for datasets generated with 16×16 pixel resolution frameblocks.

To serve as a baseline for comparison, we also trained on images of the Caltech-UCSD Birds-200-2011 (CUBS) dataset. CUBS contained 200 categories of bird types with a total of 11,788 images [Wah01a]. The attribute data was represented in multiple variable-length captions for each image. An example caption for an image in CUBS is, "this bird has a bright yellow crown, a long straight bill, and white wingbars" – these captions are much like the attributes used for our datasets of frameblocks, however they are less specific concerning the make up of the image, such as in object placement, pose, and setting. Results from all of our datasets outperformed those of CUBS, which implies that our selection of attributes more accurately represented images compared to the CUBS captions.

5 RESULTS

Since GANs do not optimize any kind of objective function and operate instead on a learned latent space, they can be difficult to analyze analytically [Ric01a]. To evaluate our results, we measured the closeness of our generated images to their target frames via Mean Squared Error (MSE), Structural Similarity Index (SSIM), and Peak Signal-To-Noise Ratio (PSNR). Training losses of all block sizes are produced in Figure 4.

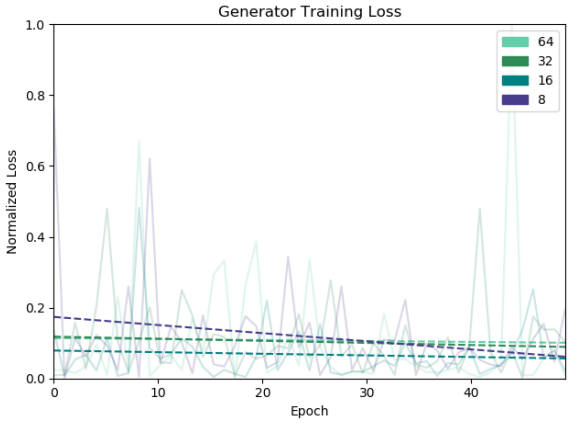


Figure 4: Final training losses for the AttnGAN model trained on frame data.

5.1 Image Analysis

The final generated frames for our 2-frame and 30-frame experiments are shown in Figures 5 and 6, respectively. In both cases, we trained and tested with 4 datasets of block sizes 64, 32, 16, and 8 pixels. We found that the smaller the resolution, the better quality of image was generated. The decreased size in image space improved the ability for the model to learn relationships between pixel value and textual attributes. The larger frameblock sizes of 64 and 32 did not have enough context to produce cohesive outputs. For the 2-frame experiment in particular, these sizes did not contribute enough data to produce arguably good results.

While the model trained on 8 pixel dimension frameblocks produced the best renders of those we generated, it did not generate enough detail or coherence between blocks to constitute a visually accurate rendering. The very center portion, which should display a rotating wooden texture, rendered very little texture in all of our outputs. We note this quality is true for most portions of the image with extremely fine textural details, such as the subsurface scattering on the clear dome and dark glass objects, and the reflection on the floor. This is likely due to the lack of texture and lighting information in the text attributes. Although we theorized that decreasing the block size would improve textural generation, frameblock dimension did not affect the level of

detail generated aside from improving object structure. Further experiments would be necessary before we may determine how to improve the accuracy for rendering textures and lighting using our method of data generation.

Our analysis of output images is produced in Table 3. The values of any given category were mostly uniform across dimensions, which does not trend with the dramatic change in visual quality between outputs. Our further analysis in Table 4, which compares results on a block-by-block basis, reflects greater differences between dimensions. This shows that generated frameblocks were more accurate individually, but scored lower when combined for the final frame. This premise is also supported by the significant improvement in loss over the CUBS baseline. Notably, the most accurate images generated from our datasets were close to or completely correct, while the most accurate images of CUBS exhibited larger amounts of error.

Frames 2 and 4, which were not included in the training or test datasets, still held high correspondence to their targets; the AttnGAN model was able to extract enough semantic meaning from our attributes to create plausible renderings of unknown views, and thus function as a true rendering system. Further, the high similarity between our 2-frame and 30-frame experiments demonstrates the abstraction power of our method. The 64 dimension case of the 2-frame experiment, however, did not contain enough data for learning the semantic descriptions, and thus we conclude our approach is limited by dataset size and training time. Optimizing the number of epochs to data and frames processed would create crisper images, however we found that no amount of training could entirely overcome cross-boundary incoherence or greatly improve the generation of textures. Taking the average of all frameblock sizes, our dataset yielded 156% better MSE, and 130% better SSIM over the CUBS dataset.

Blockdim 64			
Frame	**MSE**	**SSIM**	**PSNR**
1	45.443	0.857	31.556
2	46.304	0.863	31.475
3	45.607	0.859	31.540
4	46.533	0.863	31.453
Blockdim 32			
Frame	**MSE**	**SSIM**	**PSNR**
1	46.341	0.862	31.471
2	47.772	0.859	31.339
3	47.272	0.857	31.385
4	48.513	0.857	31.272
Blockdim 16			
Frame	**MSE**	**SSIM**	**PSNR**
1	42.997	0.874	31.796
2	45.049	0.867	31.594
3	45.273	0.863	31.572
4	45.778	0.864	31.524
Blockdim 8			
Frame	**MSE**	**SSIM**	**PSNR**
1	39.723	0.867	32.140
2	41.894	0.855	31.909
3	40.865	0.857	32.017
4	42.083	0.854	31.890

Table 3: Results of MSE, SSIM, and PSNR for all dimensions and frames analyzed.

6 CONCLUSION

Generative Adversarial Networks show promise for image generation, but can be highly unstable and at times generate undesirable results. Attention-based GAN networks are proven to synthesize higher-quality images with more semantic relevance and improved composition. Using the Attentional GAN model created by Microsoft Research, we successfully implemented a proof of concept for generating frames without a commercial renderer. Our method of frameblock selection reduced dataset size for long animated sequences, and yielded 156% improvement in MSE over the CUBS dataset. While the model trained on the CUBS data generated birds in the likeness of their input captions, fine control of the pose and background was not possible. Our semantic text attributes improved this capability for all block sizes, as we were able to dictate form, material, and structure. The smaller dimensions of 8 and 16 obtained the most accurate renderings, since attributes were more focused for each image. The larger sizes of 32 and 64 failed to provide enough textual detail to generate accurate outputs.

Our results present plausible basis for future work in this area. Replacing the rendering pipeline is a computationally very difficult problem, and would not be possible without the use of advanced deep neural networks. We assert that future advancements in a *deep rendering graphics pipeline* could decrease rendering costs for static and dynamic rendering systems, and ultimately provide an alternative to traditional rendering processes.

7 REFERENCES

[Aug01a] Augusteijn, M., F., Motion generation with a recurrent neural network, in Conf.proc. ICNN'94,

Averages Per Datatype			
Datatype	**MSE**	**SSIM**	**PSNR**
CUBS	105.495	0.392	27.925
Ours (64)	45.971	0.882	34.209
Ours (32)	47.475	0.887	33.310
Ours (16)	44.775	0.899	33.848
Ours (8)	**41.141**	**0.900**	**34.989**
Medians Per Datatype			
Datatype	**MSE**	**SSIM**	**PSNR**
CUBS	106.526	0.349	27.856
Ours (64)	27.068	0.993	33.807
Ours (32)	32.881	0.994	32.961
Ours (16)	26.673	0.996	33.871
Ours (8)	**23.334**	**0.999**	**34.472**
Best Output Per Datatype			
Datatype	**MSE**	**SSIM**	**PSNR**
CUBS	5.184	0.989	40.984
Ours (64)	0.948	0.998	48.366
Ours (32)	0.993	0.998	52.156
Ours (16)	0.245	0.999	54.243
Ours (8)	**0**	**1**	**100**

Table 4: Results per frameblock, with calculations for average and median. The best value for each column of every category is shown in bold.

Vol.5, pp. 2726-2731, 1994.

[Chi01a] Child, R., et al. Generating long sequences with sparse transformers, in CoRR abs/1904.10509, 2019.

[Chr01a] Christensen, P., et al. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering, in Conf.proc. ATG'37, Vol 3, No. 30, 2018.

[Fin01a] Finn, C., Goodfellow, I., and Levine, S. Unsupervised Learning for Physical Interaction through Video Prediction, in Conf.proc. NIPS'16, Barcelona, Spain, Curran Associates Inc., pp. 64-72, 2016.

[Goo01a] Goodfellow, I., et al. Generative Adversarial Nets, in Conf.proc. NIPS'14, Montreal, Canada, MIT Press, pp. 2672-2680, 2014.

[Hua01a] Huang, H., Yu, P.S., and Wang, C. An Introduction to Image Synthesis with Generative Adversarial Nets, in CoRR abs/1803.04469, 2018.

[Liu01a] Z. Liu, et al. Video Frame Synthesis Using Deep Voxel Flow, in Conf.proc. ICCV'2017, pp. 4473-4481, 2017.

[Luc01a] Lucic, M., et al. High-Fidelity Image Generation With Fewer Labels, in Conf.proc. PMLR'19, Long Beach, California, USA, No. 97, pp. 4138-4192, 2019, pp. 4183-4192.

[Mah01a] Mahjourian, R., Wicke, M., and Angelova, A. Geometry-based next frame prediction from monocular video, in Conf.proc. IV'17, IEEE, pp. 1700-1707, 2017.

[Mat01a] Mathieu, M., Couprie, C., and LeCun, Y. Deep multi-scale video prediction beyond mean square error, in CoRR abs/1511.05440, 2015.

[Par01a] Parmar, N., et al. Image Transformer, in Conf.proc. PMLR'18, No. 80, pp. 4055-4064, 2018.

[Qia01a] Qiao, T., et al. Learn, Imagine and Create: Text-to-Image Generation from Prior Knowledge, in Conf.proc. NIPS'19, Curran Associates, Inc., No. 32, pp. 887-897, 2019.

[Rad01a] Radford, A., Metz, L., and Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, in Conf.proc. ICLR'16, San Juan, Puerto Rico, 2016.

[Ree01a] Reed, S., et al. Generative Adversarial Text to Image Synthesis, in Conf.proc. ICML'16, New York, NY, USA, No. 48, pp. 1060-1069, 2016. Adversarial Networks, in CoRR abs/1710.10916, 2017.

[Ric01a] Richardson, E., and Weiss, Y. On GANs and GMMs, in Conf.proc. NIPS'18, Montreal, Canada, Curran Associates Inc., No. 32, pp. 5852-5863, 2018.

[Sem01a] Sudhanshu K., S., A Proposal for using ANNs for CG Animation, in CC-AI: The Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology, Vol. 10, No. 1-2, pp. 93-106, 1993.

[Vas01a] Ashish Vaswani et al. Attention is All you Need, in Conf.proc. NIPS'17, No. 30, Curran Associates, Inc., pp. 5998-6008, 2017.

[Wah01a] Wah, C., et al. The Caltech-UCSD Birds-200-2011 Dataset. Tech. rep. CNS-TR2011-001, California Institute of Technology, 2011.

[Xu01a] Xu, T., et al. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks, in Conf.proc. IEEE/CVF'18, pp. 1316-1324, 2018.

[Zha01a] Zhang, H., et al. StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks, in CoRR abs/1710.10916, 2017.

[Zha02a] Zhao, Y., et al. 3D Point Capsule Networks, in CoRR abs/1812.10775, 2018.

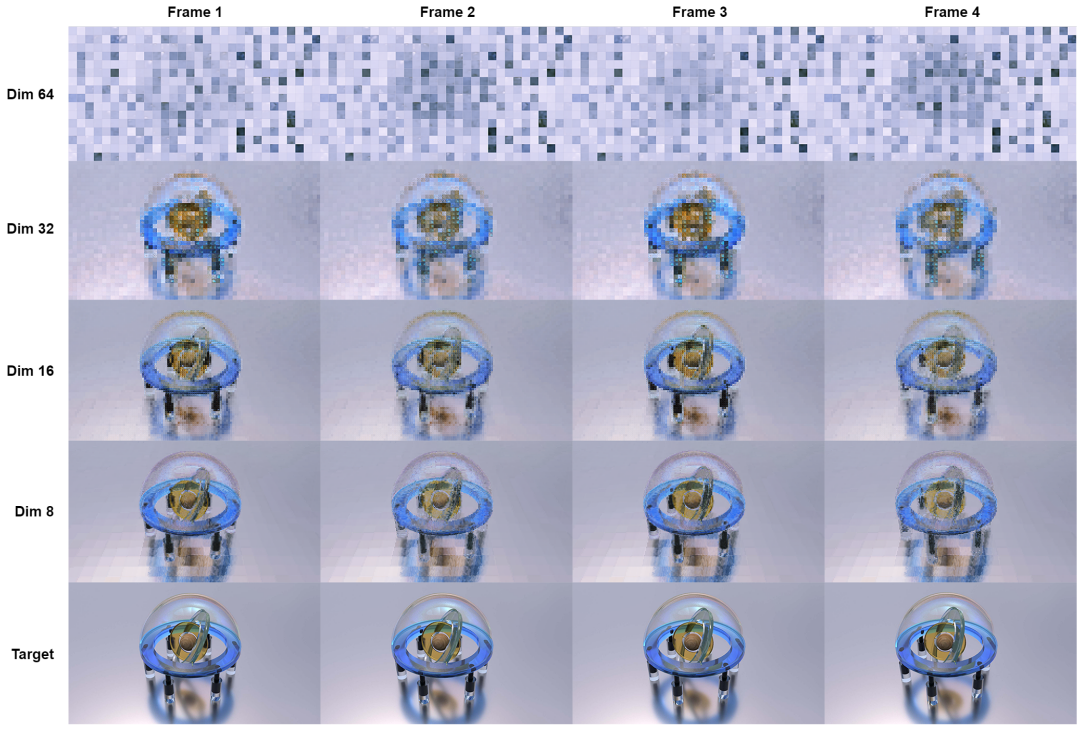


Figure 5: Results of our framework for 64, 32, 16, and 8 dimension frameblocks, using 50 epochs and the first and third frame of the 60 frame animation. The network did not have enough training data for the 64 and 32 dimension cases, and thus more epochs would be necessary to produce rendering given only 2 frames as inputs.

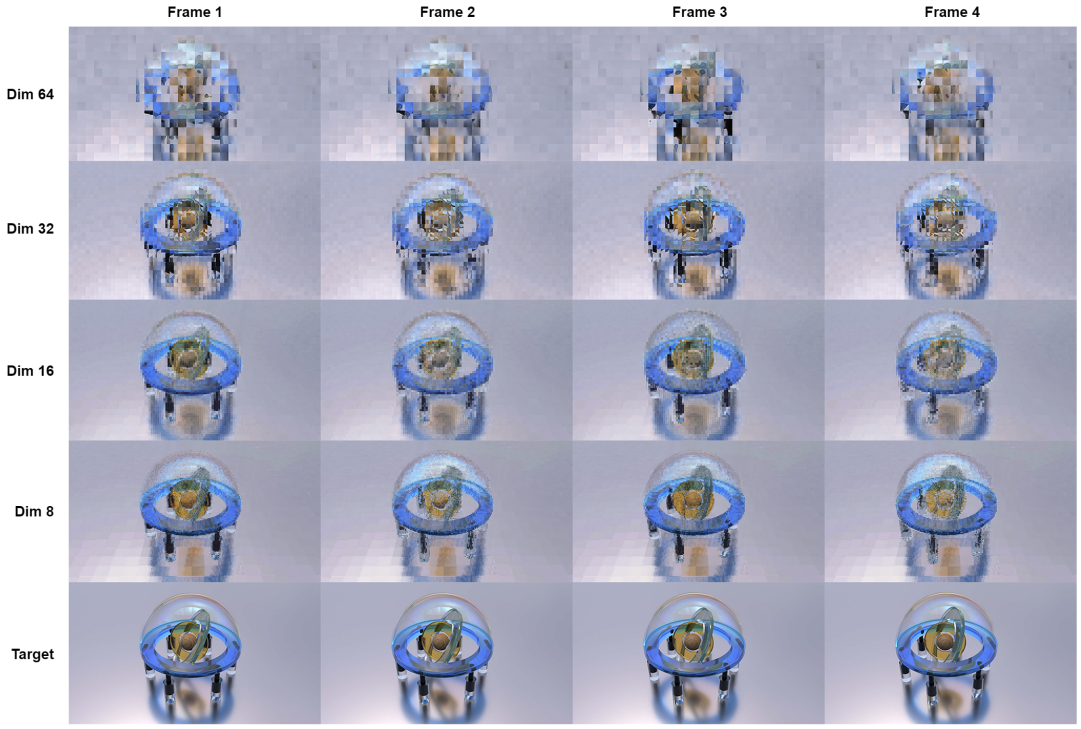


Figure 6: Results of our framework for 64, 32, 16 and 8 dimension frameblocks, using 50 epochs and every other frame of the source 60 frame animation. Frames 1 and 3 were included in the training set, while 2 and 4 were not.