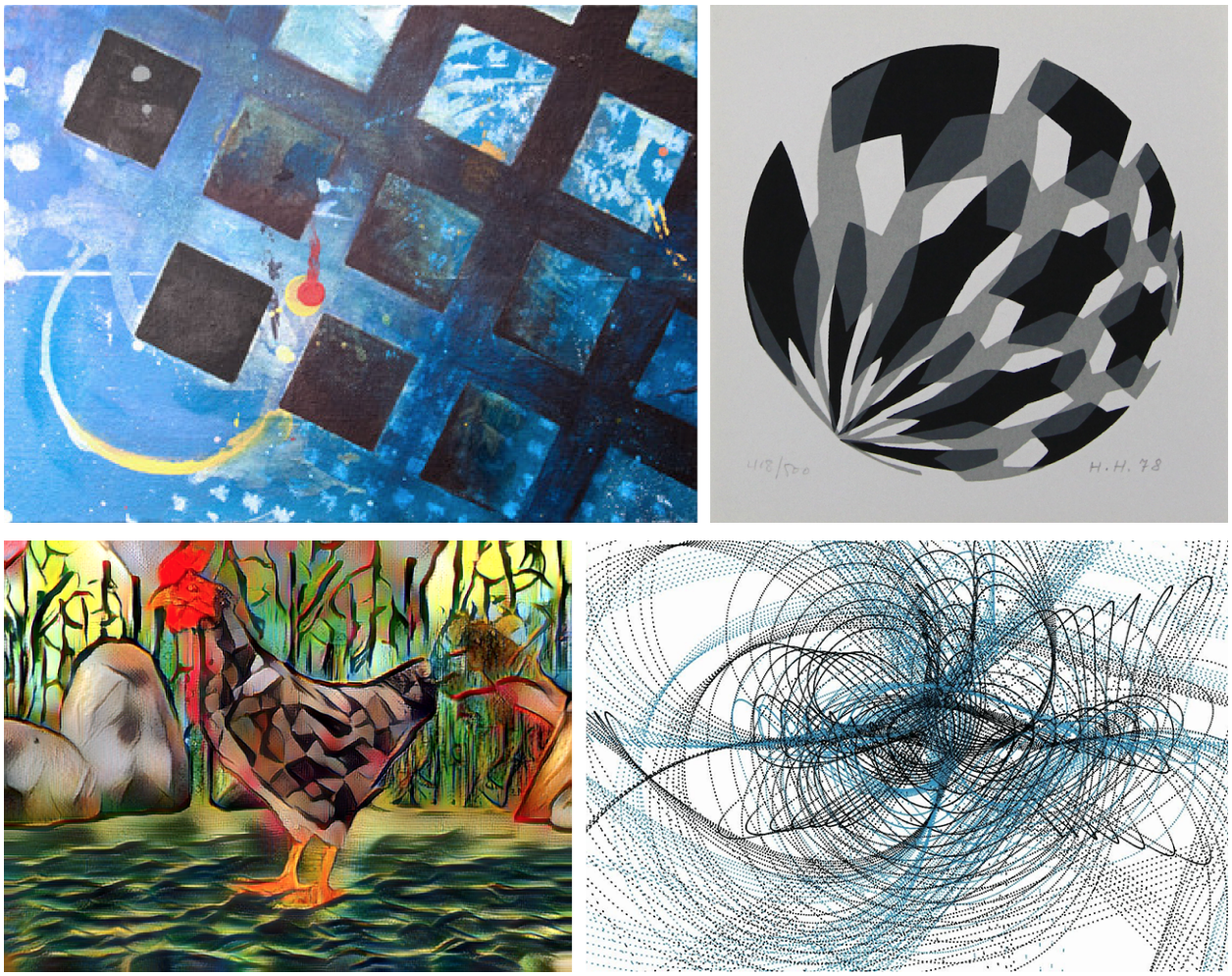
Machine Learning for the Arts
UC San Diego | Fall 2019
Final Project

Full-Stack Interior Design

An Artistic Portrayal of Information Clutter



[1]

Owen Jow

Description:

I use generative models (VAEs and GANs) to synthesize meshes, textures, and layouts for population of a 3D scene. By "overpopulating" the scene with a limitless collection of mostly gaudy items, I intend to represent the growing "clutter" of virtual content over time as more and more people (and now machines) add their creations to the mix. The cohesive output of my work is a navigable scene that fills itself at an increasing rate with stylish yet artificial objects. Along the way, I have also inadvertently created a prototype for an ML interior design tool.

Concept:

Originally, I wanted to create and add LEGO objects to a 3D scene in a nice, structured way, and let the LEGOs saturate space until the program ran out of memory. In addition to the automatic construction process, the user would be able to manually put things into the scene, not realizing at first that the scene would only get more and more cluttered, and that nothing would ever really go away. My idea was to portray the "entropy" (in a "disorder" or "heat death" sense) of digital information, in that it is easy to create and to publish, and even to produce content that seems beautiful or meaningful, but that inevitably all of these things are buried in noise as the universe of virtual data grows and kippleizes. I thought that the LEGOs could represent something clearly artificial, something obviously created by a person or a machine. I was also planning to include style transfer in order to represent how we sometimes dress things up and label them aesthetically appealing, at times missing the content (the underlying geometry) for the form (the style). I figured I could add an interesting feature to each object, like a decal or a star polyhedron, which would not be very observable under stylization, and then I could gradually reduce the level of stylization so that users would see what they had missed. Lastly, as a tertiary theme, I thought I could reflect upon simplicity in an increasingly messy world by periodically reverting from the visual cacophony to a state when the palette was cleaner, i.e. by providing glimpses at the way the scene used to look from certain viewpoints. Yet the buildup of clutter would remain inexorable.

Figure 1: A very messy scene

Obviously I didn't do most of these things. I realized that the full vision would be difficult to achieve in a week, so I toned it down. However, many of the initial ideas have persisted in one form or another. For example, I turned LEGOs into voxels and image stylization into mesh texturization. I also write flashbacks to the offline animation, and intentionally make the object deletion button much less effective than the object addition button.

Midway through the week, after switching the scene from a forest to a room, I thought of another interpretation for this project which was more practical and less of a symbolic stretch: interior design. Given that (in one mode) it generates and places objects around a room based on user input, the program could serve as a reasonable tool for an interior designer. Imagine a human wanting to spice up her room with some crazy new designs. Using this kind of program, she could ask the ML system to generate a new lamp or chair or desk, and then use ML (again) to help adjust the object's material and transform in a simulated environment. I'm not an interior designer, but this sounds handy to me.

Technique:

a. **Mesh Generation**

As an initial step, I use 3D-GAN [2] to generate voxel objects. Each 3D-GAN model is trained on ShapeNet data for a single object category; for the purposes of this project the categories are sofas, desks, and chairs. Given a trained 3D-GAN generator, one can sample latent vectors, feed them to the generator, and obtain random objects of the class the generator was trained on. This is what I do in order to synthesize voxel sofas, desks, and chairs.

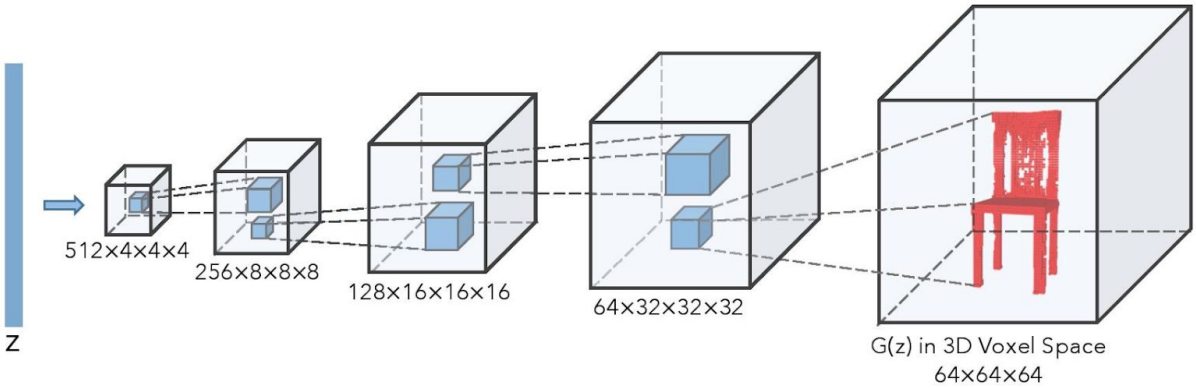


Figure 2: 3D-GAN generator. Figure is from the 3D-GAN project page [3].

I then convert the voxel objects to meshes using marching cubes. But the resulting meshes lack texture coordinates, meaning I still need to perform UV mapping if I want to later apply textures (which I do). I implemented a simple parameterization scheme which takes the largest connected component of the resulting mesh (there may be

multiple after marching cubes), extracts the largest manifold patch, cuts the patch into a topological disk, maps the boundary to a unit square, and then parametrizes the interior such that each vertex receives UV coordinates equal to a convex combination of the boundary vertices' UV coordinates, where the weights of the convex combination are based on the distances to the corresponding boundary vertices. I would like to have used geodesic distances here, but my meshes were too degenerate for geodesics to be computed, so I settled for L2 distances instead.

Alternatively, I can UV map via Blender's "Smart Project" operation. When I do this, I convert each mesh into a watertight manifold first. However, I don't really like the texturizations that arise from this method. I do, however, appreciate that it preserves the full geometry of each object.

b. Texture Generation

I train an MSG-GAN [4] for 700 epochs on ~3000 patches of Van Gogh paintings from The Met [5]. MSG-GAN is a GAN that synthesizes and discriminates against results at multiple scales. Quality propagates from coarser scales to finer scales, so that color and detail (albeit coarser detail) first manifest in the lower-resolution images, and extend to the higher-resolution images as the training process continues.

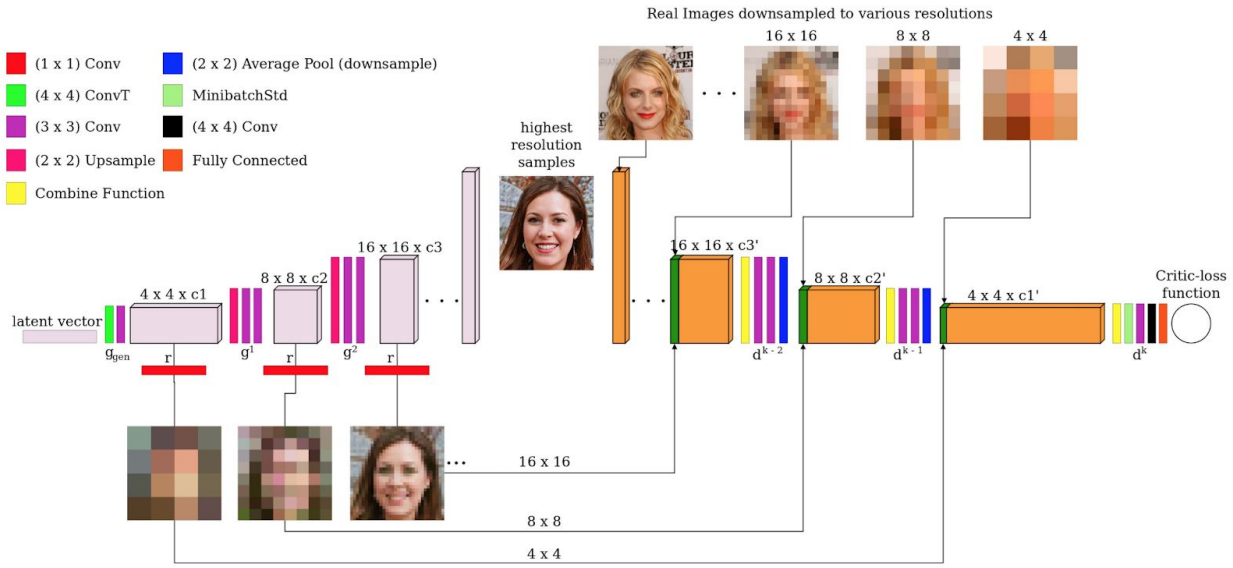


Figure 3: MSG-GAN architecture. Figure is from the MSG-GAN paper [4].

After using the MSG-GAN to unconditionally generate Van Gogh-inspired textures, I feed the textures to a second multi-scale texture synthesis algorithm [6] in order to obtain a larger texture set at a higher resolution (512x512).

c. Layout Generation

I train a convolutional denoising variational autoencoder (wow, that's a mouthful) [7] for 5000 epochs on a dataset of 700 256x256 satellite images [8]. The VAE has two convolutional layers and a fully-connected layer that take the image to a 512-D latent space, and a fully-connected layer and three transposed convolutional layers that take the 512-D latent vector back to a 256x256 image. At test time, I sample random latent vectors (or perturb encoded images) and send them through the generator in order to obtain new satellite-esque images to use as layout sampling maps.

d. Scene Composition and Rendering

At this stage, I combine my generated assets into one cohesive product by placing randomly sampled objects around a pre-made 3D room and rendering the scene using Panda3D [9]. First, I select a random generated layout. Then, if automatic spawning is on, I determine each object's location by sampling the layout (converted to PDF form). The top-down view of the room is discretized into a 256x256 grid so that I can easily position the objects according to this map. In each case, I choose a random mesh and texture to use, initialize it at some height, and let it fall with an acceleration of 9.8 meters per second squared until it hits the ground.

If automatic spawning is off, I let the user decide where to position each object. When he or she hits the object addition button, a textured mesh is randomly selected and then placed at a scaled distance along the forward vector of the camera frame.

If offline mode is selected, the camera will spin around in the center of the scene, taking screenshots as objects fall from the surrounding sky. (I designate a "no-object zone" in the immediate vicinity of the camera in order to avoid object-camera intersections.) When passing through a certain, predetermined angle for the second, third, or nineteenth time, the camera image will temporarily (for 10 frames) switch to the way the scene looked from that angle on the previous revolution of the camera. All of these frames are written to an output video upon termination of the program.

Process:

I began with nothing but a simple walkthrough rendering system from my previous work. In the context of this project, that system was hungry for 3D models to render. So I made it some – or rather, I got 3D-GAN to make it some. Since the 3D-GAN output was in the form of a voxel grid, I considered writing or adapting a simple voxel engine so that I could stack the voxels one by one (my original intent was to drop every LEGO block as an individual entity). I managed to find a Panda3D voxel engine project [10] that I could potentially extend, but ultimately decided that

it would be simpler and safer to convert the voxel objects to meshes, and that as long as the objects still looked somewhat cuboid and ML-generated it would be a similar effect.

The next hurdle to overcome was UV mapping. I spent a significant amount of time – too much, I think – trying to implement a reasonable mesh parameterization. The main challenge was that the marching cubes meshes had a lot of structural issues, meaning my code crashed when I tried to compute geodesics, extract manifold patches, or compute a harmonic map parameterization of the interior. I tried using some existing mesh cleanup software to fix my meshes (e.g. [11], [12], [13]), but to little avail. Eventually I settled for the simple parameterization mentioned in the "Technique" section. There's definitely still a lot of room for improvement.

Now that I had texture coordinates, it was time to generate textures. (Up to this point, I had been testing texturization with an image of Picasso's "La Muse.") I knew that I wanted the textures to look like paintings, and I decided that the source data would be Van Gogh paintings because they were relatively colorful and distinctively (well, to my somewhat untrained eye) non-photorealistic. Also importantly, Van Gogh paintings were available as public-domain images from The Met. After scraping these images, and splitting them into 256x256 patches (this solved the initial problem of having only 24 paintings; also, I chose 256x256 because it was the highest resolution I was planning to use with MSG-GAN), I fed them to an MSG-GAN for training, which was straightforward except for the time it took – about two days.

Figure 4: Examples of patches from Van Gogh paintings.

On top of MSG-GAN, I wanted a quick and easy way to synthesize more textures at arbitrary resolutions. So I applied the multi-scale neural synthesis technique from [6].

For layout generation, I originally wanted the network to place individual objects according to its understanding of typical scene composition, perhaps training on voxelized 3D scene spaces and predicting object placement probabilities in (again) a voxelized 3D space. But then I realized I didn't have the temporal wherewithal to collect all of this data and architect a model and go through a probably lengthy training process. So I switched to my current sampling idea, initially imagining I would discretize the space by the maximum size of each object and color the objects according to the original RGB layout image, so that I could show a top-down view and the scene

would look roughly like the sampling source. (I didn't end up doing either of these things, because I am lazy I guess.) I considered reusing MSG-GAN for layout generation, but I didn't want to endure another two-day training process; this time, I wanted a lightweight model. Thus I adapted an old autoencoder project of mine to use. (Incidentally, this is why the VAE was trained in a denoising fashion; I was investigating denoising autoencoders in that old project.) I hooked up a small VAE encoder-decoder architecture and chose to use satellite images as training data (I could have used anything, but I chose satellite data because at this point I was still using an outdoor scene and satellite data seemed appropriate). Then I started training, and happily it didn't take very long (less than an hour, probably) for the autoencoder to be creating decent reconstructions.

At this point, I had all of my assets: meshes, textures, and layouts. The only thing left was to put them all together. This is when I made all of the "experience" tweaks. I experimented with a bunch of different environments from the Alice Gallery [14], and settled on the empty room because its browns and beiges provided a decent contrast for the many green objects in my full texture set (unlike the green forest scene I had been using). Also, I thought it could be seen as a sort of "art museum," and the room setting fit with the furniture object categories. I added a sunset view through the window because sunsets make for attractive scenery (also, without an explicit image, the background would "smear" with foreground objects on the laptop version of the program). At this time, I also made many other small adjustments, like object start and end heights, scene scale, scene boundaries, assigning a "flashback" yaw during offline animation writing, setting the distance for object addition, hampering the deletion functionality, adding camera pitch controls, and tuning object addition rates when automatic spawn was on.

Results:

Figure 5 depicts some objects that I generated using 3D-GAN. The voxel output has been postprocessed (binarized via thresholding and reduced to the largest connected component) and converted to a mesh using marching cubes.

Figure 5: 3D-GAN output.

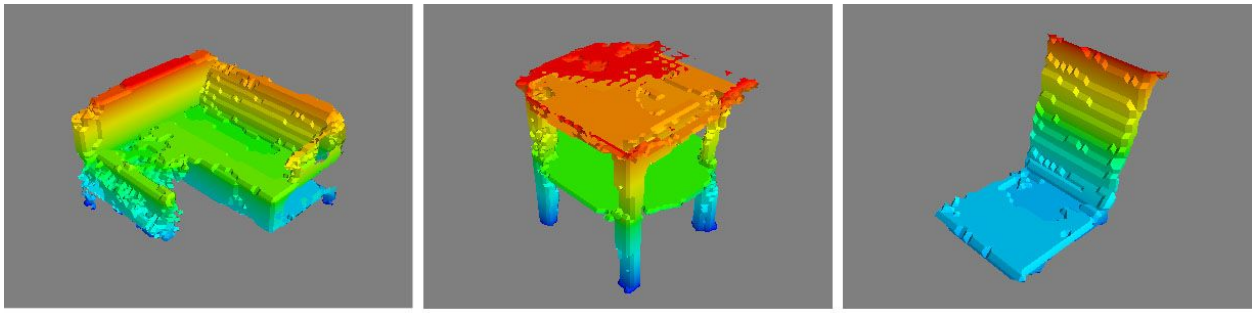
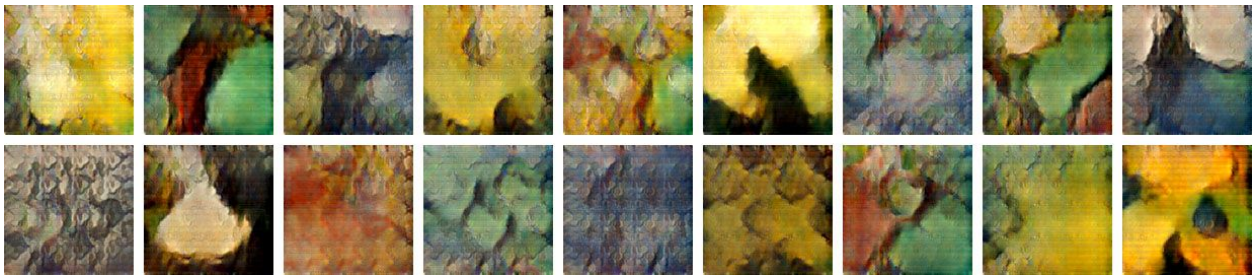


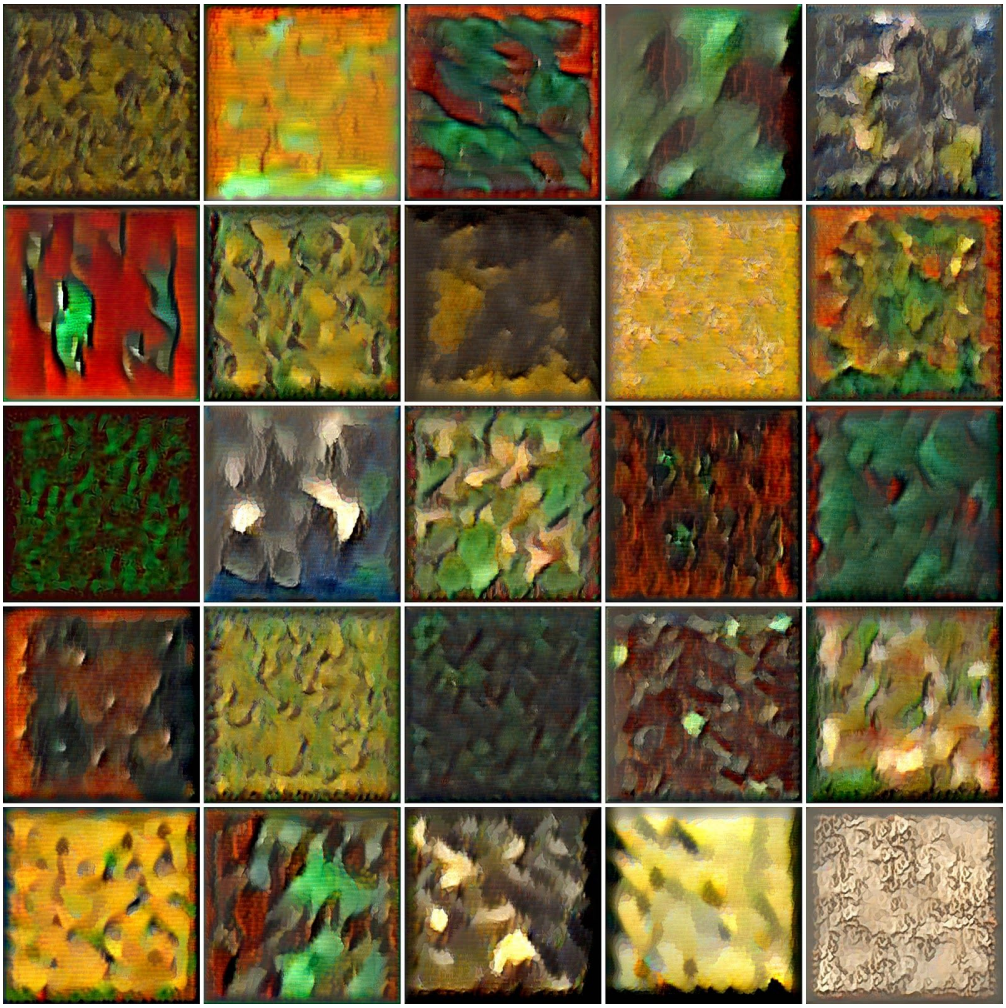
Figure 6 depicts some 128x128 textures that I generated using an MSG-GAN which was (under)trained on Van Gogh paintings.

Figure 6: MSG-GAN textures.



After generating the MSG-GAN textures, I passed them through a second synthesis stage in order to produce more textures at a larger (512x512) resolution.

Figure 7: Second-stage textures.



Next (in Figure 8) are examples of layouts generated using a convolutional VAE trained on satellite imagery. They were created by interpolating between encodings for actual inputs.

Figure 8: Synthesized layouts.

I consider the main result of my project to be the interactive program. Here are some snapshots of that program during execution (i.e. of the room scene populated by ML-generated objects).

Figure 9: Usage screenshots.

Reflection:

Looking back on this project (or more literally at the screenshots above), I am filled with one main regret: that I didn't do more to fix the meshes and maintain the full structure of the objects. In retrospect, it wouldn't have been much more work to at least retain the entire mesh throughout parameterization (by just processing each connected component and manifold patch separately, which I already have the code for, and then merging all of these things into one UV-mapped mesh at the end).

Other than that, I am pretty satisfied with how things went, considering it was essentially a week-long project in the midst of all my other end-of-quarter stuff. I was able to incorporate a lot of different components in order to put together a scene, which is a useful prospect in the visual computing world that I am currently enmeshed in. For example, recently there has been a void left by the retraction of the SUNCG dataset (a collection of 3D house scenes which was being used in a lot of research projects before the original source of the data filed a lawsuit and SUNCG had to be taken down). With the type of technology in my project, one could imagine using ML to create a new version of the SUNCG dataset.

I like that the project "evolved" in my mind from an artistic portrayal of clutter (which, now that I think of it, is a bit unappealing because people wouldn't usually want to look at clutter) to something that could actually be practical (given an enhanced version of everything I did here) as an interior design tool. This could be one future direction of my work.

Another direction would be to push the original "entropy" idea further, for example by creating a virtual set piece containing a series of 3D objects with increasingly disordered structure and texture. It could even be justified mathematically, for example according to Weyl curvature or with dynamical systems of Kasner metrics as per [1].

References:

[1] Matilde Marcolli. "Entropy and Art: The View Beyond Arnheim." http://www.its.caltech.edu/~matilde/EntropyArtChapter.pdf

[2] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In *NIPS*, 2016.

[3] http://3dgan.csail.mit.edu

[4] Animesh Karnewar, Oliver Wang, and Raghu Sesha Iyengar. MSG-GAN: Multi-Scale Gradient GAN for Stable Image Synthesis. *arXiv preprint arXiv:1903.06048*, 2019.

[5] https://www.metmuseum.org/art/collection

[6] Xavier Snelgrove. High-Resolution Multi-Scale Neural Texture Synthesis. In *SIGGRAPH ASIA Technical Briefs*, 2017.

[7] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.
[8] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote Sensing Image Scene Classification: Benchmark and State of the Art. *arXiv preprint arXiv:1703.00121*, 2017.
[9] https://www.panda3d.org
[10] https://github.com/anossov/dorfdelf
[11] https://github.com/achicac/BiharmonicRepair
[12] https://gfx.cs.princeton.edu/proj/trimesh2
[13] https://github.com/alextsui05/remesher
[14] http://alice.org/pandagallery

Code:
https://github.com/ohjay/inexorable

Links to Results:
I've generated and released some sample meshes, textures, and layouts on Google Drive.