Generative AI Art Internship Report

November 12, 2023

Tom Eijkelenkamp
s1889338

Supervisors:
Michael Wilkinson
Almila Akdag Salah

1 Text to Image Generation

Lately there has been lots of popularity for algorithms that could compute artistic or real-looking pictures from a text prompts. As the diffusion algorithm outplayed the earlier variational encoders and generative adversarial networks on the task of image synthesis, many came to know DALL-E, Midjourney or Stable Diffusion. The generated images were much more convincing to the audience, closely compared to realism.

The goal of these algorithms is not only to create images that look realistic, many styles of visuals could be generated as ones that look like paintings or drawings, designs for posters or websites. There are many more applications for image generation.

Image generation algorithms can be measured on their quality in many ways, Borji (2018) and Borji (2021) show a list of all the metrics out there. For example you can look at the variety of possible images that can be generated or you can evaluate how accurate a certain category of visual is displayed. You can qualify the realism of a generated image. On the other hand you can look at if the neural network is overfitting, underfitting or generalizing really well.

For the research of this internship we will be looking into evaluating realism of generated images. We will particularly study one aspect of realism, light illumination and shading. Our aim is to develop a technique for quantifying the interaction of light in these images. We conduct a variety of experiments and suggest several initial approaches for evaluation methodologies.

We start out by showing how the diffusion algorithm operates to synthesize image. Then we discuss related studies regarding computer image rendering and evaluating image realism. We conduct various experiments in search of finding a light qualifying method. We finalize by proposing future directions, possible light evaluation metrics to be explored.

2 Diffusion

In this first section we look into the main elements that make up the architecture for the text to image syntheses algorithm. We explain the basic concept that is used to learn a distribution behind a set of images. Then we go into the logical parts that make up the neural network.

The idea behind using diffusion for image synthesis is that it gives good control to capture the distribution. For generating images that look like one that is from the training data set, you need to know what does belong to the data set and what does not. We are trying to find is the distribution that describes the data set. A mathematical function that could say, this image is one we want to generate and another is not, giving us the whole high dimensional landscape of possible to generate images.

2.1 Concept

In order to find this distribution of the images from the training data set, we diffuse all these images stepwise with noise.

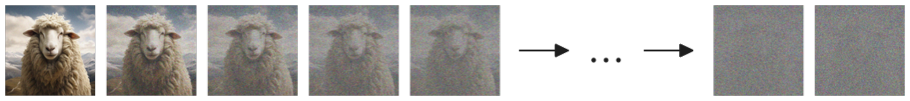


Figure 1: Incrementally adding noise to one training image

This noise is clearly not part of the distribution, so this can help us find the true distribution. We make a neural network able to learn the path back from a noised image to one that we want to generate. To do this we train the network to learn to subtract noise in an image. We have the training images that are incrementally noised until complete noise. For every step in this process, we make the neural network learn to reverse this step. By doing so we find the gradient in the distribution landscape. A vector that points in the direction in which the distribution gets more dense.

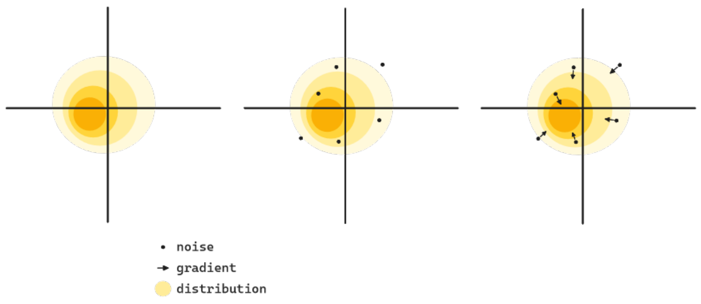


Figure 2: 2D visualization of a distribution with its gradient.

We not only want to generate an image that looks realistic or synthesized with a certain style, we want the generated image to be in line with the text that a user gave as input. All the training data is labeled with a textual description of what is represented in the picture. This is used throughout the algorithm to guide the synthesis towards a desired composition. The algorithm learns what visual aspects are related to which text descriptors.

In the end we can start from random noise and calculate using the gradient towards a picture from the target distribution. We guide this trajectory with a textual conditioning, to get a picture that is in line with the prompt.

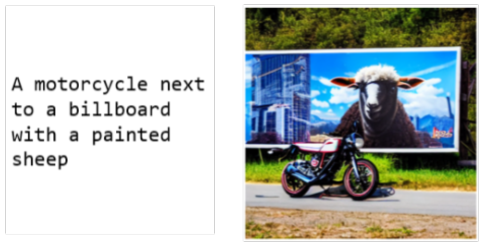


Figure 3: Generated image from a text prompt. Source: Mao et al. (2023)

2.2 Architecture

2.2.1 Training data set

The process is started with the training dataset. For the stable diffusion network a dataset of 200 million high detailed images were used that was collected from the internet. For each image a textual description label is required to train the network. This specific dataset is using the alt-text attribute from the html in which the image was found.

2.2.2 Noise training images in steps to complete noise

Every image of the dataset is diffused over time. This is done by incrementally adding guassian noise to every picture for a number of steps.

2.2.3 U-Net

A neural network is used to learn how to detect and remove noise from images. It is trained to do this using the diffused training images. Every step of diffusion is used as an training example. The network must predict for an image at time step t what the image at time step t-1 was. So only one step of guassian noise is removed. This is repeated for all the steps of the diffusion process.

The network must learn to segment the noise and replace it with expected content. The input image has the same dimensionality as the output image. A u-net structure is used to translate noisy patterns in the data to patterns that

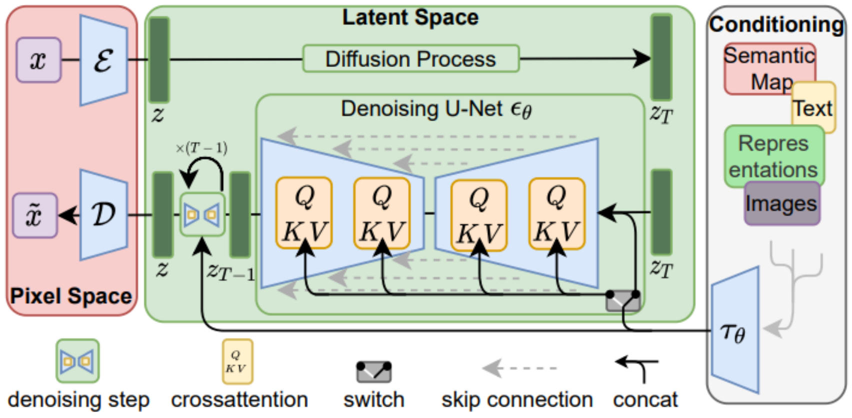


Figure 4: Architecture of the whole algorithm. Source: Rombach et al. (2022)

belong to the distribution. The network can be seen as a function that points in the direction of the desired distribution. Somehow there is a relation between both shallow and complex patterns, to go from a noisy picture to a less noisy one. The u-net learns this translation, by iterively showing it a picture with noise and the same picture with less noise. The u-net tries to predict the less noisy image. The network is updated according to the amount of error that it makes on this prediction.

2.2.4 Convolution layers

The u-net is made up by many convolutional layers. The idea of such a layer is that a kernel is used to detect the location of a certain pattern in a picture. This is done by sliding the kernel over the image and compute how much the pattern correlate to the picture at that location. If the pattern correlates the values are high, and on the opposite when the kernel pattern has nothing to do with the local pattern in the picture then the resulting value is low. Pictured in figure 5.

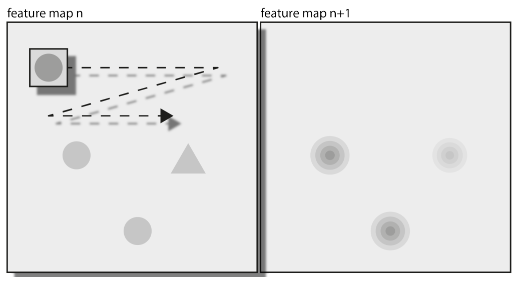


Figure 5: Kernel moving across a feature map.

The convolution layers are followed up by eachother. Each layer is detecting the locations of the patterns of the earlier layer. So the following layer is finding how patterns are combined with eachother. The deeper layers represent more complex patterns in the sense that they detect, patterns that are made out of patterns, that are made out of patterns, and so on.

2.2.5 Skip layers

For deeper networks it helps to not only have this specific follow up order of layers. When adding connections between layers that skip one or more layers in between, more layers can be added to increase learning potential. For the u-net this is done even more than normal convolution networks. Here shallow layers are connected to shallow output layers, but also complex layers are connected to complex layers. In this manner the network is able to learn the correlation between patterns on every level. How to replace noisy patterns for patterns that belong to the distribution is learned on shallow level, but also in higher complexity.

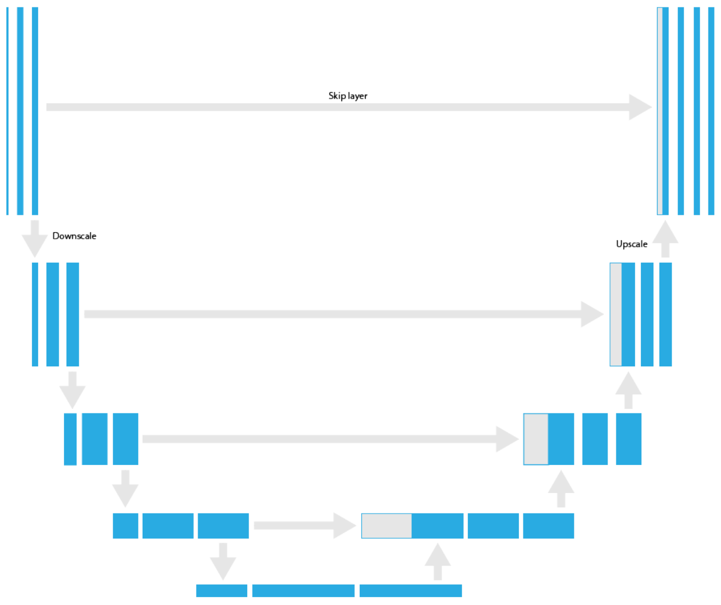


Figure 6: The u-net uses skip layers to create connections on every level. Ronneberger et al. (2015)

2.2.6 Cross attention

Cross attention is used to guide the translation from noisy patterns into pictural patterns with a text prompt. Not only is the noisy pattern replaced by a pattern from the distribution, also it's directed by the text. The attention mechanism is a way to learn what kind of visual representation the text is related to. Then the denoising process is using this as a guidance to create a picture.

2.2.7 Text labels

As described earlier, every image comes with a textual description of what is represented in the picture. The network learns the relation between the text and the visual representation, using the cross attention mechanism.

2.2.8 Image to latent encoder and decoder

The training data consists of high detailed images. It is computationally very expensive to do the whole training process using images in this detail. In studies it is show that it is not necessary to do this using this high dimensional representation. You don't get worse results by running the algorithm on a compressed version of the images. To improve computational efficiency, an encoder and decoder are put at the start and end of the algorithm. First an image is compressed to latent space, and at the end it is translated again back to pixel space.

3 Related literature

For this next section we look at existing methods for qualifying light illumination in pictures. We look at how light is simulated for image rendering using computer graphics. Additionally we show various related studies about what makes up a realistic image regarding light or other aspects.

A realistic photo is created from light. Waves of light move through the world, bounce on objects following particular laws. A camera can catch light and translate this to a picture. When you do this in the real world, you will get a realistic picture. There you have the realistic laws of nature.

But now we create images with an ai system. This computes what an image will look like from looking at example pictures. The system learns what the world looks like, from images. Trying to capture this into a mathematically formalized distribution. A set of rules from which we create new images. These pictures can be anything, but a subset of it will be realistic photos. But what defines this realism. For our research we will look into how the laws of nature regarding light movement is expressed in ai generated images.

In the world of computer graphics, there has been profound research on how to render realistic-looking scenes to a computer screen. Many studies on how to approximate how light flows through the world before it possibly hits the camera plane. Simplistically shown in figure 7, which sets the basis for an algorithm to compute how light rays scatter through the scene from a light source. This abstracts how in computer graphics images can be synthesized from a 3D data point composition.

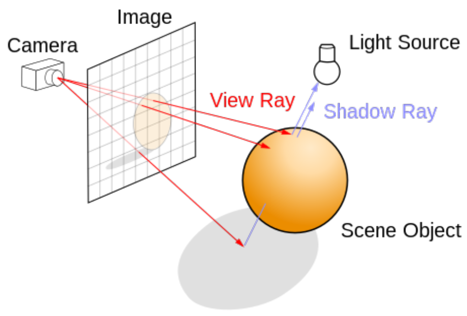


Figure 7: Basis of the ray tracing algorithm. Source: `https://en.wikipedia.org/wiki/Ray_tracing_(graphics)`

As diffusion models share the same goal of creating digital images, we can question how these methods compare. Ray tracing methods work very well for visualizing realistic reflections on object surfaces and create shadows nicely aligned to how the objects intersect the light. More advanced concepts as the refraction of light within glasslike objects are simulated well. For this research we look at how these concepts, as illumination, shadows, and refraction, are simulated in generative AI.

The rendering pipeline for computer graphics can become very complicated. For simulating all kinds of materials, shapes, any type of fluid or chemical effect. For simplicity we will focus on some basic illumination principles.

The basic Phong shading algorithm simulates the reflection of a light wave on a surface. This is simplified to three kinds of light interactions: ambient, diffuse and specular. The last two have to do with how rough or mirror-like the material is that the light bounces off, shown in figure 8.

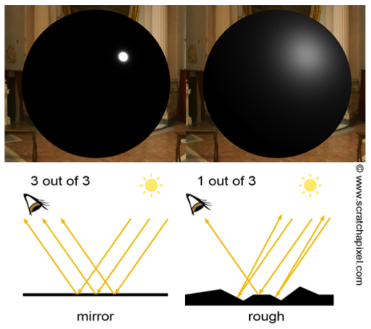


Figure 8: Diffuse and specular components of phong shading

There is always a layer of light particles endlessly scattering on the objects illuminating the whole composition, but taking too much time to compute. In the phong equation this is simplified by adding a small light intensity on every object. This creates an overall more illuminated scene. The three parts are

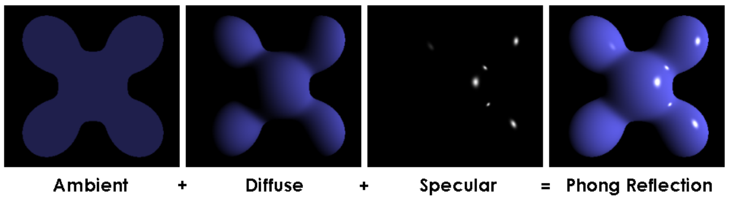
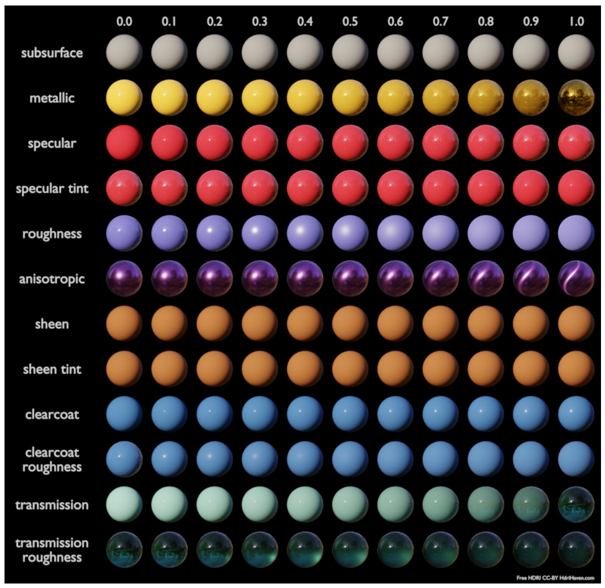


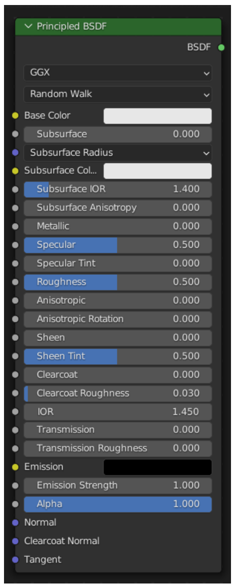
Figure 9: Phong equation

computed separately and added together, shown in figure 9.

The blender engine can render many kinds of light interactions, to simulate any type of material. Figure 10 shows a visualization of the possibilities.



(a)



Principled BSDF

BSDF

GGX

Random Walk

Base Color	
Subsurface	0.000
Subsurface Radius	
Subsurface Col...	
Subsurface IOR	1.400
Subsurface Anisotropy	0.000
Metallic	0.000
Specular	0.500
Specular Tint	0.000
Roughness	0.500
Anisotropic	0.000
Anisotropic Rotation	0.000
Sheen	0.000
Sheen Tint	0.500
Clearcoat	0.000
Clearcoat Roughness	0.030
IOR	1.450
Transmission	0.000
Transmission Roughness	0.000
Emission	
Emission Strength	1.000
Alpha	1.000
Normal	
Clearcoat Normal	
Tangent	

(b)

Figure 10: Blender material spectrum and settings

3.1 Color compatibility and realism

In Wong et al. (2012) they evaluate how the compatibility of colors on different objects in a picture can make a photo look unrealistic. When a part of one image is inserted into another picture it can look unrealistic.

They say the colors do not match from both pictures. I think this is because the lights are different. The lights in both images can have different colors, intensities and can come from other angles. I believe this will influence the color appearance. When both light compositions do not match, it will make

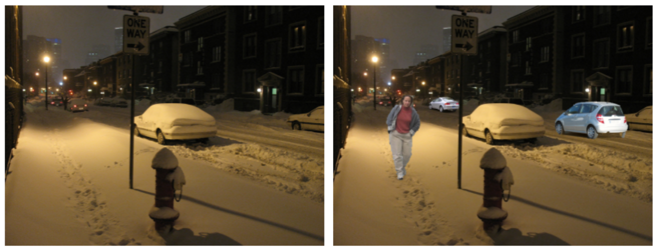


Figure 11: Unrealistic image composite with a white car and a man in gray pants inserted into a street under yellow lights Lalonde et al. (2007)

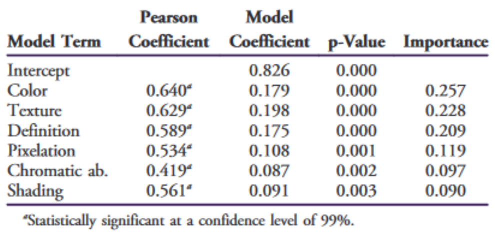
the picture unrealistic. In their study they develop an algorithm to transform the image, by changing the color distribution. When the color spectrum of the objects matches better, it makes the image look more realistic.

3.2 Predict realism using machine learning

In Fan et al. (2018) they use an annotated dataset of images with realism scores to predict the perceived realism for generated images. For our study we could use this as an inspiration for developing a shading qualifier. If we could predict the realism of images regarding illumination of light on surfaces, this could be a possible metric for our qualification objective. Their method does require a large annotated dataset which is a downside, compared to a complete computational method.

3.3 How do properties of an image correlate to perceived realism

According to Pardo et al. (2018) shading is not contributing as much as other aspects to how real an image looks. They have done an human evaluation study on two sets of images, one set of images of real world photographs and another set of images with an identical composition rendered by a computer graphics engine, Unreal.



Model Term	Pearson Coefficient	Model Coefficient	p-Value	Importance
Intercept		0.826	0.000	
Color	0.640[a]	0.179	0.000	0.257
Texture	0.629[a]	0.198	0.000	0.228
Definition	0.589[a]	0.175	0.000	0.209
Pixelation	0.534[a]	0.108	0.001	0.119
Chromatic ab.	0.419[a]	0.087	0.002	0.097
Shading	0.561[a]	0.091	0.003	0.090

[a]Statistically significant at a confidence level of 99%.

Figure 12: Correlation Coefficients between Perceived Realism. Source: Pardo et al. (2018)

They find that the properties color and texture of an object contribute most to perceived realism. Figure 12 shows their statistical results in a table.

3.4 Shadows in art

As we are looking at an art generation system we can look at how important it is to have realistic shadows in visual art. In Cavanagh (2005) they find that errors in shadow depiction are not so easily noticed. They say that the mind might interpret shadows in a more simplistic way than the complicated physics of reality. Artists use this in their works, creating more simplistic-looking shadows. Experiments show that unrealistic depictions often go unnoticed. Example show in figure 13.

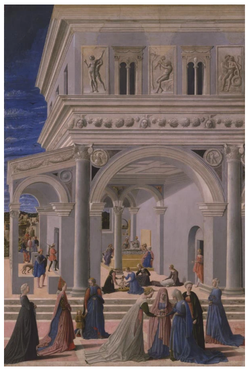
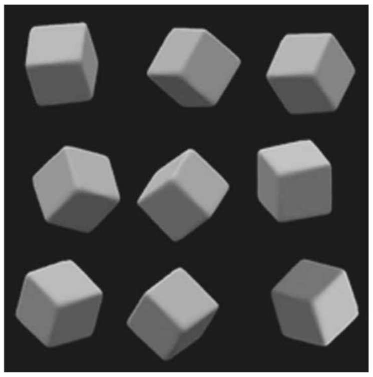


Figure 13: By 1467, artists such as Fra Carnevale had mastered consistent perspective but not consistent lighting. Source: Cavanagh (2005)

3.5 Illumination inconsistencies

The study of Ostrovsky et al. (2005) looks into shading and illumination where they also find that unrealistic visualizations are not so easily noticed. They perform a number of experiments on shading depiction where they look at what aspects are noticed and which are not. Figure 14 show pictures of the experiments.

For example, they do an experiment where they show an experiment participant a picture of a number of cubes all illuminated from one side. For only one of them, the light comes from the other side. The participants have to tell which cube is the odd one out. Another experiment they do is on images which show illumination inconsistencies. Here they use images where one part of the image is deliberately showing a shadow direction towards the other side, compared the the shadow direction of the rest of the image.


(a) Experiment image where participants have to find the odd one out Ostrovsky et al. (2005). All objects are illuminated from one side except one.

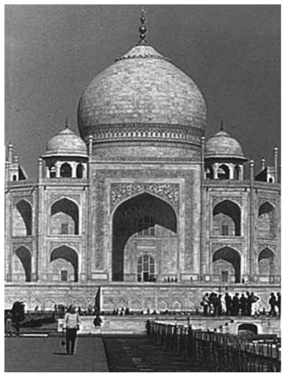

(b) Image in where participants have to say if they notice inconsistency in the illumination Ostrovsky et al. (2005).

Figure 14

4 Method

In the coming section we perform a number of experiments to find direction in our search for a suitable metric. We use an existing realism qualifier as a basis point. From there we finetune this towards a method we could use for qualifying light in images. We identify problems with the studied method and experiment with possible solutions to overcome these.

We want to create a good metric to measure the quality of shading in an image. To start out, we look at already existing methods out there that are used to evaluate the quality of image generation algorithms. One widely used metric is Fréchet Inception Score Heusel et al. (2018). This shows the distance between the feature distributions for two sets of images. Using the Inception classification network Szegedy et al. (2015), the present features in the images are detected. All the features in the two sets of images are counted. Then the distributions are compared for the two sets, resulting in a distance value.

This method can be used to qualify the realism of generated images. You can use a set of realistic photos, and compute the Fréchet distance to generated images. From this you can say something about how truly the generative model synthesizes realism.

For our research we want to say something about how well the model performs for shading properties. To get an idea, we rendered a set of images that contain very clear aspects about shading and see how closely the generative algorithm relates to these patterns for shading. Blender does a good job for creating realistic looking shading properties in images. It is not as exact as real life photos, but it makes it easy to make images that show clear patterns for shading and leave out all other patterns that are not important for this research. We can create images that only contain a simple shape against a background with one light source. This image shows in clear detail how the shadow aligns to the shape that casts it, also it shows how the gradient moves across the shape according to the angle of how the light hits the object. You can see an example of this in figure 15.

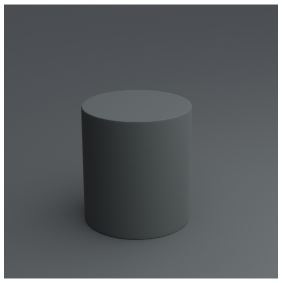


Figure 15: A cylinder shaped object against a background, illuminated by one light source. Created with Blender. Here you can see clear characteristics of shading. The position of light influences how bright each part of the object is displayed. The intensity gradient is shaped according to the normal along the surface.

We rendered many of these kinds of images using different shapes and materials, to synthesize a set of images containing simple details about shading. Using the Fréchet Inception Score we can study how stable diffusion generates similar images.

The problem of course is that stable diffusion can generate much more complicated scenes than those created with blender. You can have images of complete cities, containing many light sources and objects making the light scatter in complicated ways through the scene. Also you can create images that are not representing a 3D world, where shading properties are not important. As a starting point, we focus on how well the generative network is performing for very simplistic 3D scenes in terms of shading.

Using stable diffusion we generate a set of images, similar to what we can create with blender. Just one object on a surface, both made from some material and the scene is illuminated by a light. Then we create this set of images with blender as well. The results can be compared using Fréchet Inception Score.

For generating using stable diffusion, we use the xl-base-1.0 and xl-refiner-1.0 parameter sets [1]. We want to create a set of images containing a number of shapes: cubes, pyramids, cones, cylinders, and toruses. Also, we vary the materials (metallic, plaster, wood, pvc, cloth, stone, cardboard) for what the shapes are made out of, as for the background. We created a prompt "even flat {1} diffuse ground surface, one solid {2} made out of an even flat {1} diffuse

[1] `https://huggingface.co/stabilityai/stable-diffusion-xl-base-1.0`

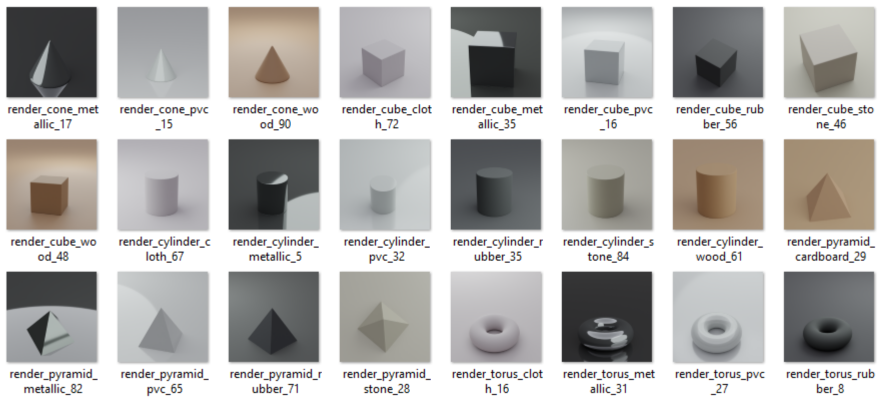


Figure 16: A small fraction of images rendered with blender

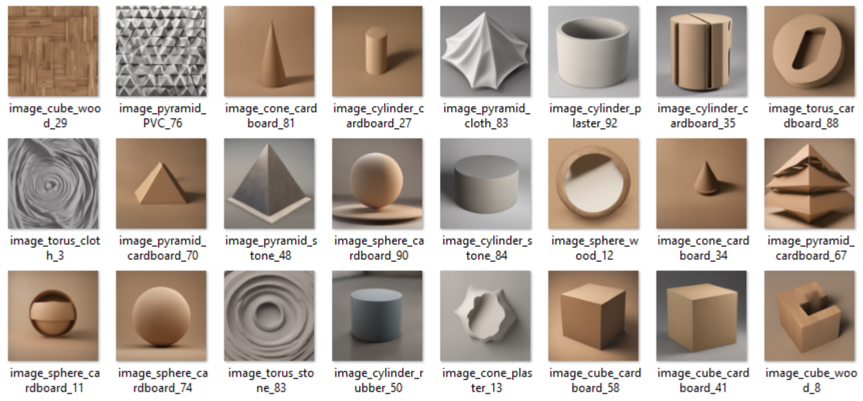


Figure 17: A small fraction of images generated with stable diffusion

material". Here we replaced {1} with the material and {2} with the shape[2].

For the set we create with blender, we create these simple shapes as well. The materials we simplify by just varying the color, specular, metallicness and roughness properties. We don't create woodlike structures or other variations in the appearance. We use a script to vary light and camera position, looping through a random number of compositions. The number of shape-material variations are equal and replicate the number created with SD.

Doing a simple unfiltered FID test Seitzer (2020) on both sets results in a score of 115.036

This relative high score might be due to the high detail of the images. The resolution here is 1024x1024 pixels. We do the test again scaling the images down to a resolution of 64x64 to test this hypothesis. This results in a FID

[2]`https://github.com/TomEijkelenkamp/stable_diffusion_generator/`

score of 130.313 which is higher than the earlier obtained result, showing that the high value is not due to the size of the images.

The generated images from stable diffusion also contain many images that do not look like the composition we created with blender. For our study we do not test how well the generated image is in line with the text prompt, we just want to see how realistic the shading looks. For a next test we filter out all the images from SD that do not contain a similar composition as the ones from blender. We are not cherry picking on shading quality, we filter on composition.

We use a similar metric to filter all the images. We only want the images that have a certain composition. We use a couple of example images that show this composition. For each shape and material we have one example picture. Again we use the inception network to calculate the features within those images. Then we calculate the features within the whole collection of generated images. Then we only use the images that are within a threshold similarity distance from the example pictures. We use the angle between the feature vectors as a distance measure[3].

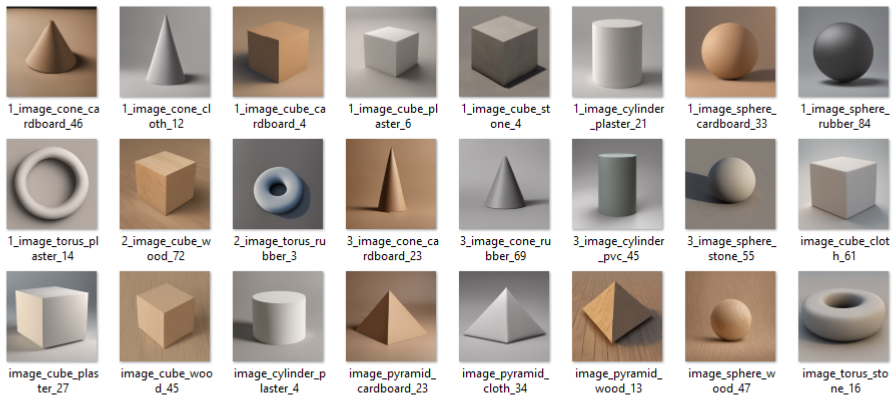


Figure 18: A small fraction of filtered set of images generated with stable diffusion

Now we obtained a new set of images and compute the FID score again. This time it results in a score of 100.903 being lower than our first result.

To make the score even more accurate we would have to make the two sets completely identical on all variables other then shading. We want to know if the light interaction in an ai generated image is following the laws of the real world. If we create images with stable diffusion and identical images with blender, we could compare the light interaction. Identical meaning, the composition in the image is the same. The same shape is displayed at the same location. The material and the color of the object is in both images the same. The textures

[3]https://github.com/TomEijkelenkamp/image_similarity_clustering/blob/main/similar_to_examples.py

are identical. The only variable is how stable diffusion displays shading and how blender does this. If we would create two large sets, and do the FID score, to only measure shading properties, we should eliminate all variation on other features.

We tried to eliminate part of the variation by filtering out compositions that do not show only one object of a particular shape displayed against a background. We could continue to eliminate variation by improving the material and color properties. If we could create a better replicate for materials using blender, we could improve our shading score.

On another experiment we calculated the main colors used in the set of images created with stable diffusion. We calculated the histogram for the used colors in all the images. Then we excecuted the k-means algorithm to compute 32 mostly used colors. Those colors we used to generate a new set of images in blender. The goal was to create a set of images that is more closely related to the ones from stable diffusion. We try to eliminate the variation of used colors. We want to make the two sets of images be mainly of the same color distribution[4].

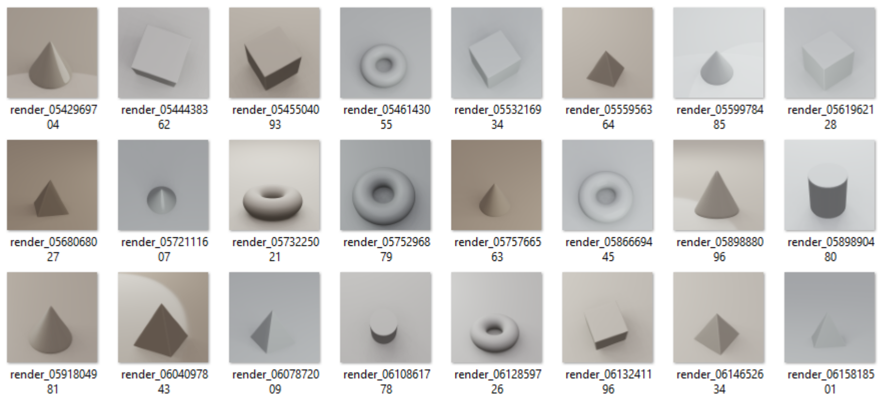


Figure 19: Snapshot of the set created with blender, using the colors obtained through the k-means algorithm.

We failed to create a set that is more closely related then the previous experiment. The resulting FID score here is 123.440.

4.1 Problems with FID score

- **FID score uses all features in the two sets of images, not only shading features.** The FID score is not designed to specifically measure shading quality of images. It measures the performance for generated images comparing general features that are present to a set of realistic images. You can do a FID score with image sets as CIFAR-10 or ImageNet.

[4]`https://github.com/TomEijkelenkamp/image_color_distribution`

For our shading study it's only important to look at a particular proportion of the features captured by the inception network. We want to see if features for shading are in line with what you would expect in reality. It would be better if we would have a network that can detect shading features for images, and then find the Fréchet distance between a generated set and a realistic set.

The inception model is designed to categorize images. There are 1,000 different classes it can classify images into. It does this by detecting all kinds of features in images, that can direct a prediction for which class the image belongs to. For example, when you got an image of a car, it will have wheels, a number of doors and windows, a certain shape. When these elements are there within an image, the change goes up that it categorizes as a car.

Features that have to do with shading are only a small proportion influencing the classification. You would expect the change of a good categorization to go down, when the shading realism is completely off. When there is a cube in the image, but the shadow is round, the image is less likely to classify as displaying a cube. It might as well display a sphere, if you look at the shadow.

Also the color gradient on a shape indicates the form of that shape. When this property of shading is not correct, the classification also might be less confident.

We do not know to what proportion these features for shading are influencing the categorization. Because of this we do not know how the FID score is influenced by these features.

- **InceptionV3 is trained on a particular set of images.** Similar to the problem described above, the inception network is trained to classify a particular set of classes. For these types of images the network can detect features very well. For images that contain other properties, many important features will not be detected, simply because the network was not trained to detect such features. For our problem, we want to detect very smooth flat and clear surfaces. The images used for training inception network are rich in complexity and intricate details. The network will perform less at classifying the features we are lookinig for. It would be better to use a classifier that is particularly good at these simple images we use.

- **InceptionV3 requires images to be of a certain size.** The inception network needs images of the dimensions 299x299. Our generated and rendered images have the dimensions 1024x1024. For this, the images need to be scaled down, losing detail for feature extraction. High detailed aspects of shading will not be captured by the network and so the FID score is not able to take this into account. The score will more be an approximate blurred measure of shading properties. A sharp shadow outlining the edge of a box cannot be perfectly matched as realistic or not realistic. The FID score will more be a rough estimate how shading features are compared to shading features that blender simulates.

4.2 Finding features that relate to shading in InceptionV3

We want to modify the FID score for our research specifically qualify shading properties. We are going to look at which of the features that the InceptionV3 network captures, relate to shading. Then we can segment these features, and only use these to find a shading pattern distribution for a set of images. Then we can compare the shading properties of two sets of images and see how much they differ. Simply put, we do a FID score only using shading features.

To find out what features relate to shading, we are going to do an experiment. We create images in which we vary components that display all characteristics of shading. Because it's easy to simulate these kind of images using blender, we use this to render the shading depictions. We vary the following apsects:

- Object to be displayed. Similar to what we did in an earlier experiment we create a scene showing one object against a background. We vary the scene using the same type of shapes: cube, pyramid, sphere, cone, torus.
- Color of the object. Here we use loop through a set of hue values in the HSV spectrum.
- Material characteristics. We again use different values for metallicness and roughness.
- Position of the light illuminating the object. We use nine positions, creating a three by three grid of options, not varying the height. So the light is casted from the back, front, the sides and corners.

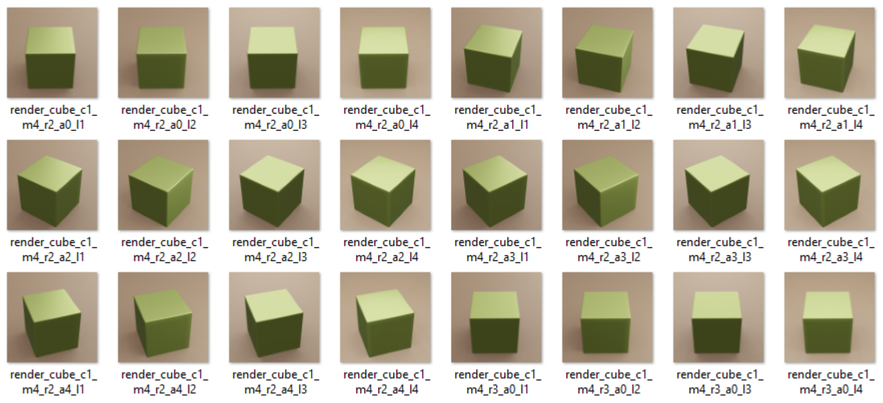

render_cube_c1_ m4_r2_a0_l1
render_cube_c1_ m4_r2_a0_l2
render_cube_c1_ m4_r2_a0_l3
render_cube_c1_ m4_r2_a0_l4
render_cube_c1_ m4_r2_a1_l1
render_cube_c1_ m4_r2_a1_l2
render_cube_c1_ m4_r2_a1_l3
render_cube_c1_ m4_r2_a1_l4
render_cube_c1_ m4_r2_a2_l1
render_cube_c1_ m4_r2_a2_l2
render_cube_c1_ m4_r2_a2_l3
render_cube_c1_ m4_r2_a2_l4
render_cube_c1_ m4_r2_a3_l1
render_cube_c1_ m4_r2_a3_l2
render_cube_c1_ m4_r2_a3_l3
render_cube_c1_ m4_r2_a3_l4
render_cube_c1_ m4_r2_a4_l1
render_cube_c1_ m4_r2_a4_l2
render_cube_c1_ m4_r2_a4_l3
render_cube_c1_ m4_r2_a4_l4
render_cube_c1_ m4_r3_a0_l1
render_cube_c1_ m4_r3_a0_l2
render_cube_c1_ m4_r3_a0_l3
render_cube_c1_ m4_r3_a0_l4

Figure 20: A small fraction of images simulating shading characteristics, created with blender

With all these images created, we find out how they activate features in the inception network. We look at the last layer, just as done with FID score. We expect that shading features will be located in earlier layers of the network. The last layer stores more complete objects, complex combinations of forms. Shading relates more to how color gradients go across surfaces. Intuition guides us to look more into shallower layers as well, as they store color patches and gradients.

First we look at the activations in the last layer. We use pearson correlation to find out what features correlate to our list of shading characteristics.

Image categorization should not be influenced by the way an most of these shading properties. The direction where the light comes from, or the rotation of the object does not make it another object. The network should be invariant to these variables. This casts doubt on finding features in the inception network for these properties. The material and color should be detected, because this indicates what kind of object it is.

After running the pearson correlation we get a large matrix of numbers. Every number represents how this one feature correlates linearly to one characteristic, such as metallicness. There are 2048 features we are looking at. To make more sense out of this, we only look at features that really correlate. For every characteristic, we order the features on how much they correlate. We look at the most correlating features.

From the table we find that the different shapes are influencing the activations the most. There is a large correlation between the shape (cube, pyramid, etc) in the picture and the feature activations. Also for the different colors, there correlation is high, but lower then for shapes. What is interesting to see is that also the light position is captured by the network. The feature activations relate to the light position of the presented picture. There might be a way to make use of this, to detect any error in light direction. At last we looked at the matellicness and roughness, this is also detected by the network, but the feature activations are less influenced by this[5].

From these numbers it is still hard to get an intuition for what these features are representing. Is this really the feature that represents the direction of the light shining on the object. Is it really capturing the color of the object displayed. To get a better intuition where the shading features are detected in the inception network, we are going to do a visualization.

A possible way to visualize what is happening inside the inception network is to show the activations of the feature maps. We can input an example picture and see how the network reacts. As an experiment we show an image of a cube to the network. The whole network consist of many layers, including 93 convolution layers. Each convolution layer contains ranging from 32 to 384 kernels. In figure 21 the output of the convolution layers 30, 50 and 70 are show, when the network is shown a picture of a cube[6].

These feature maps are difficult to interpret. We wanted to visualize the

[5]`https://github.com/TomEijkelenkamp/shading_features_inception`
[6]`https://github.com/TomEijkelenkamp/feature_visual_inception`

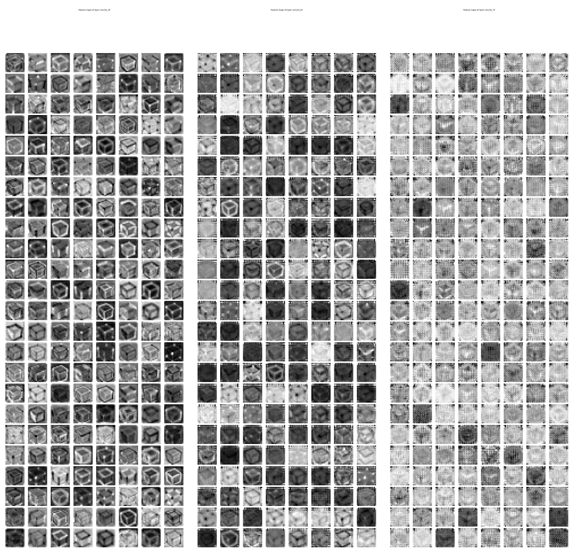


Figure 21: Feature map activations of convolution layer 30, 50 and 70 respectivily on an input image showing a cube.

corresponding features from the experiment before. There we checked for 2048 detected features, which relates to the shading characteristics. The problem here is that there are many layers in between the last convolution layer and the activation layer we use to detect features. Because of these difficulties we should find another way to find out what features correlate with shading characteristics.

5 Future work

5.1 Three dimensional representation translation

To get an accurate scoring mechanism regarding shading, it would be good to have a ground truth per tested image. We want to see how accurate the shading features are in image. We could create an algorithm that computes the 3D representation of an image. Then using this representation, we could compute a realistic render using theory from computer graphics. We could compute for a whole set of images made with stable diffusion, a whole set of images that are the ground truths. Then the distance between the two sets is the quality of shading. Still we compare to a computer simulation, not the real world.

Due to time reasons we were not able to complete the following proposals. We started sketching out some ideas to compute the 3D representation for cube and sphere images.

For images displaying a cube, you could use the canny edge detection algorithm to find all the edges of the cube. Then using this we could compute this to a set of lines with hough transform[7].

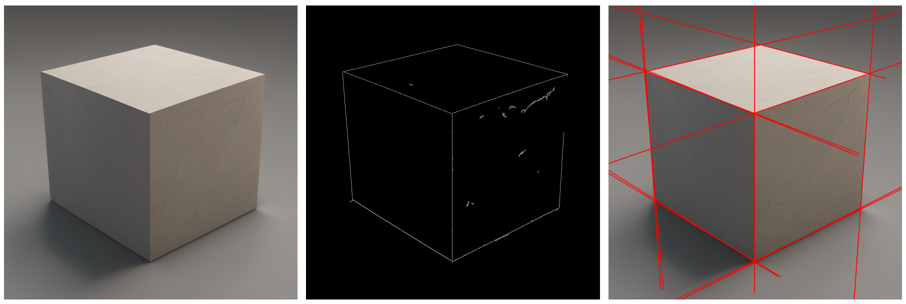


Figure 22: Edge detection on an image of a cube, and compute it to lines.

If you extend these lines, the vanishing points on the horizon can be found on the intersection point. Find the focal length using the equation: $(u_1, v_1, f) \cdot (u_2, v_2, f) = 0$. If this results in similar focal lengths for all the sides of the cube, it's really a cube and not a irregular shape. Using all the lines you could compute this to the 3D representation using geometry and perspective theory. It might be possible to compute the light position using the light intensity levels on the cube surfaces and the normals. This is an ill posed problem, since we do not know what type of lights are used and how many there are.

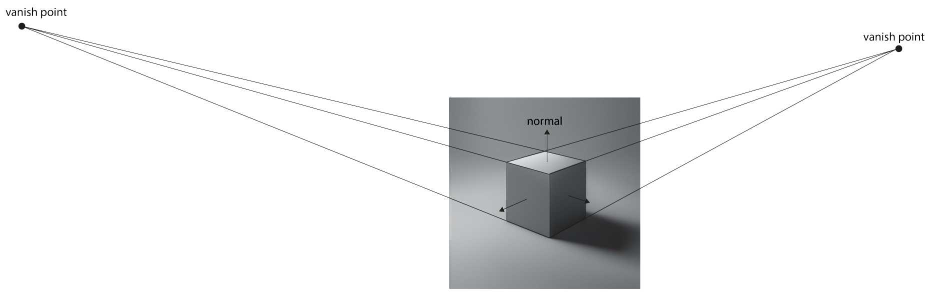


Figure 23: Possible calculation metric for 3D representation of a cube

For an image with a sphere you can find out a light position measuring the distance of the highlight from the center of the sphere[8]. Shown in figures 24 and 25. Using the measured radius relative to distance from the center to the highlight, you can obtain an angle. This is the angle between two vectors. One vector goes from the center of the sphere to the camera, and the other one goes from the center to the highlight.

[7]`https://github.com/TomEijkelenkamp/cube_3d_computation`
[8]`https://github.com/TomEijkelenkamp/sphere_3d_computation`

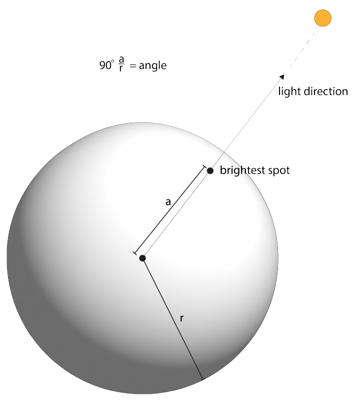


Figure 24: Possible metric to compute light position on an image displaying a sphere.

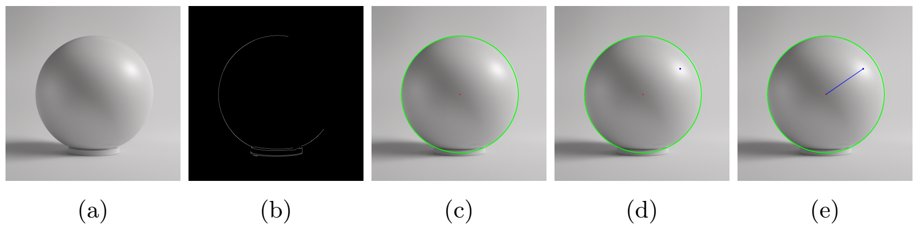


Figure 25: Using canny edge detection and hough transform to detect the sphere position. The find the brightest spot within the sphere region. Angle between the vector that goes from the sphere center to the highlight and the vector that goes from the sphere center towards camera: 70 degrees.

6 Conclusion

After all our experiments we find that it we could not create a computational method yet for analyzing shading realism. The problem is more complex to come to a complete solution. We proposed directions in which further research could be useful. A neural network could be used to detect shading feature in images. Similar to FID score, you could use this network to find the frechet distance between two distributions of shading features. Another method could be to compute a three dimensional representation out of an image. Then you could use this to render this with realistic lighting properties. The starting image and computed image can then be compared, where the difference is the score.

References

Borji, A. (2018). Pros and cons of gan evaluation measures.

Borji, A. (2021). Pros and cons of gan evaluation measures: New developments.

Cavanagh, P. (2005). The artist as neuroscientist. *Nature 434*(7031), 301–307.

Fan, S., T.-T. Ng, B. L. Koenig, J. S. Herberg, M. Jiang, Z. Shen, and Q. Zhao (2018). Image visual realism: From human perception to machine computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 40*(9), 2180–2193.

Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter (2018). Gans trained by a two time-scale update rule converge to a local nash equilibrium.

Lalonde, J.-F., D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi (2007, jul). Photo clip art. *ACM Trans. Graph. 26*(3), 3–es.

Mao, J., X. Wang, and K. Aizawa (2023). Guided image synthesis via initial image editing in diffusion model.

Ostrovsky, Y., P. Cavanagh, and P. Sinha (2005). Perceiving illumination inconsistencies in scenes. *Perception 34*(11), 1301–1314. PMID: 16358419.

Pardo, P. J., M. I. Suero, and Ángel Luis Pérez (2018, Apr). Correlation between perception of color, shadows, and surface textures and the realism of a scene in virtual reality. *J. Opt. Soc. Am. A 35*(4), B130–B135.

Rombach, R., A. Blattmann, D. Lorenz, P. Esser, and B. Ommer (2022). High-resolution image synthesis with latent diffusion models.

Ronneberger, O., P. Fischer, and T. Brox (2015). U-net: Convolutional networks for biomedical image segmentation.

Seitzer, M. (2020, August). pytorch-fid: FID Score for PyTorch. `https://github.com/mseitzer/pytorch-fid`. Version 0.3.0.

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna (2015). Rethinking the inception architecture for computer vision.

Wong, B.-Y., K.-T. Shih, C.-K. Liang, and H. H. Chen (2012). Single image realism assessment and recoloring by color compatibility. *IEEE Transactions on Multimedia 14*(3), 760–769.