Legonization of images

Rasmus Hogslätt

March 6, 2023

Abstract

The report describes how the CIELAB color space can be utilized to reproduce an arbitrary image using Lego bricks of a limited amount of random or relevant colors. The bricks can be of various sizes and are shaded using the L-value of the original image. Shaded effects appear to only yield a slight impact on the final image's perceived quality from afar.

1 Introduction

Image reproduction is nothing new or very special. In its simplest form, it is just a method to depict an image on some type of medium. Simply using a knife to carve what you see into a tree can be seen as a reproduction of an image. A more modern approach is to use a printer which uses halftoning, a method utilizing dots of varying colors, sizes and spacings, to reproduce an image. This produces results, that when seen from afar, look very much like the original image. There are also more artistic means of reproducing images. One such method is mosaic which instead of dots, uses small elements of some kind, for example bricks of Lego to reproduce an image. These images can look very similar when viewed from afar but when looking closely, each individual tile of the mosaic is clearly visible.

This report describes methods, results and conclusions of an image reproduction project implemented in *Matlab*, where an arbitrary image is to be reconstructed using bricks of Lego.

1.1 Aims

The aims of the project are

- Reproduce arbitrary image with bricks of Lego
- Bricks can have various sizes
- Use only a given number of relevant colors
- Explore how the reproduced image is perceived
- Compare signal-to-noise ratio and S-CIELAB differences of reproduced image
- Explore if the original image's L-values can be used to yield more accurate reproductions

1.2 Definition of Lego units

For ease of readability, a *unit* of Lego is defined as a 1x1 Lego brick. One *unit* is independent of the number of pixels the *unit* contains, as the implementation in this project allows the user control of the pixel resolution of the Lego bricks.

2 Method

2.1 Creating Lego bricks

The Lego bricks that were used to reproduce the original image were generated for each new image reproduction. Initially, a 1x1 Lego brick was constructed with a user defined size of pixels, referred to as one *unit* in size. This was reused to generate larger Lego bricks of *unit* sizes 1x2, 2x1, 2x2, 2x4 and 4x2, as these are some of the most iconic Lego bricks. Each brick was given border shadows and shadows for the studs by using Matlab's *brighten* function. The relative size of each stud was computed according to Stefan Gustavson's findings of Lego measurements[Gus05]. These were then stored as PNG images that could be reused. Some examples of these images can be seen in Figure 1.

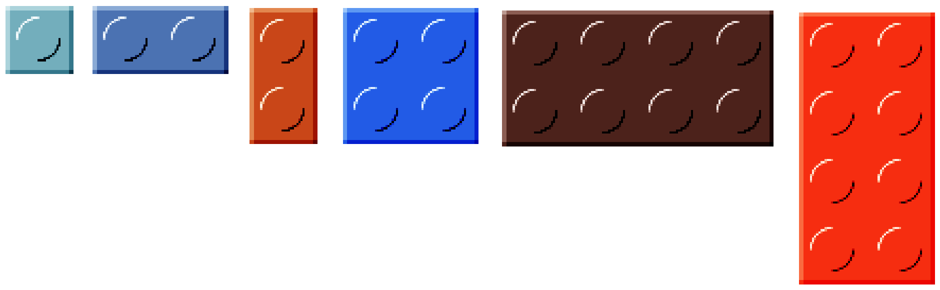

Figure 1: One of each supported brick type of various colors.

2.2 Color selection

In order to find relevant colors for the reproduction, the image was first converted into the CIELAB color space. The user was prompted to enter a number of colors to use, which was then passed into Matlab's *imsegkmeans* function, which computes the given amount of clusters using the K-means algorithm. This resulted in a list of colors in CIELAB space that could be used to generate the Lego bricks.

The K-means algorithm had to check all color values of every pixel in the image, which made the computational cost very heavy. Thus, the user was also prompted to enter a scaling factor which was used to resize the original image using nearest-neighbour interpolation. Hence, the speed of the color clustering could be significantly decreased by scaling the image.

The CIELAB values for each generated color was stored in a list and kept for later reconstruction of the final image.

An option to choose a number of random colors instead was also given to the user.

2.3 Final image resolution

If the original image was of low resolution and a reproduction with high resolution Lego *units* was sought after, the final image would contain very few Lego bricks, causing a lot of details from the original image to be lost. This is seen in Figure 2. Thus, the user was also prompted to enter a scaling factor so that the original image could be upscaled and thus output a higher resolution reproduced image.

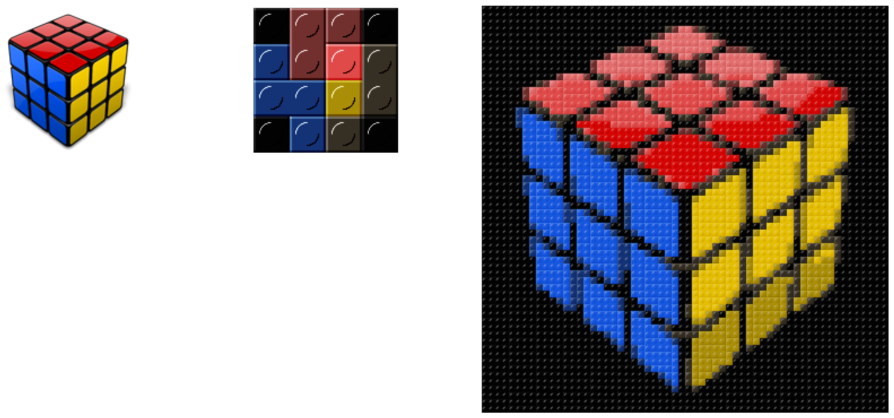

Figure 2: Left: Original image. Middle: Reproduced without scaling. Right: Reproduced with 15x scaling. Each *unit* was set to 32 pixels and 10 colors were used.

2.4 Reconstructing the image

When bricks for the reconstruction had been created for each color determined by the clustering, an image was constructed from 1x1 Lego bricks. The color of each brick was determined by the shortest euclidean distance between each brick's position in CIELAB color space and the mean CIELAB color in the original image's corresponding region. The index of each brick's color was also used to set a separate matrix's corresponding single color channel. This would later be used as a utility look-up matrix to determine where bricks of various colors had previously been placed.

The largest bricks were then placed wherever a region, the size of the brick to be placed, in the look-up matrix contained only one unique value. If that was true, the brick was added to the reproduced image, overwriting any previously placed bricks. An arbitrary element within each 1x1 *unit* was also set to a negative value, ensuring that any smaller bricks would not be placed on the larger brick later. The remainder of the bricks were placed in the image in similar structure order of descending brick size.

2.5 Brightness adjustment

The original image's L-values were used to apply shading effects on the Lego pieces by using the *brighten* function and a chosen brightness factor. The factor computed in three different ways. One by using the mean L-value of the original image and comparing it to the pixel currently being processed.

A second way by comparing the currently processed pixel's L-value in the reproduced image to the corresponding pixel's L-value in the original image. The third way instead compared each 1x1 *unit's* L-value rather than comparing only per individual pixels.

2.6 Quality metrics

Three quality metrics, signal-to-nose-ratio, S-CIELAB from 1 meters distance and mean-square-error were used to explore the objective quality of the reproductions.

3 Result

Table 1 contains image quality metrics using SNR and S-CIELAB for images shown in Figure 3. Note that not all images that were measured with metrics are not included in Figure 3.

Table 1: SNR, MSE and SCIELAB metrics for 6 different images reconstructions with different brightness applied. The SCIELAB metric was measured as if the images were seen from 1 meter away.

Grogu, 15 cluster colors, 24 pixels/unit	No brightness	Pixel brightness	Mean brightness	Unit brightness
SNR	5.8150	5.3801	5.2144	5.2380
S-CIELAB mean	0.7754	0.8519	0.9071	0.9633
MSE	13.3280	14.5132	15.4994	16.5591
Grogu, 15 random colors, 24 pixels/unit	No brightness	Pixel brightness	Mean brightness	Unit brightness
SNR	0.3820	-0.3270	-0.5707	-0.7839
S-CIELAB mean	3.4585	3.7282	3.7885	3.8674
MSE	45.7079	48.3367	49.1724	50.0000
Avatar, 30 cluster colors, 24 pixels/unit	No brightness	Pixel brightness	Mean brightness	Unit brightness
SNR	6.5679	6.1889	6.0417	6.0430
S-CIELAB mean	0.9125	0.9900	1.0578	1.1426
MSE	15.8518	17.1854	18.2012	19.1116
Avatar, 30 random colors, 24 pixels/unit	No brightness	Pixel brightness	Mean brightness	Unit brightness
SNR	5.1174	4.4966	4.4443	4.5283
S-CIELAB mean	2.0368	2.2046	2.1549	2.0954
MSE	26.8660	28.9508	28.8233	28.3329
Gyro, 10 cluster colors, 24 pixels/unit	No brightness	Pixel brightness	Mean brightness	Unit brightness
SNR	5.4880	5.2127	4.9642	4.7731
S-CIELAB mean	1.1794	1.2465	1.3215	1.4510
MSE	17.5034	18.5804	19.3941	20.3829
Gyro, 10 random colors, 24 pixels/unit	No brightness	Pixel brightness	Mean brightness	Unit brightness
SNR	6.5589	6.0429	5.9762	5.9803
S-CIELAB mean	3.4254	3.6095	3.6605	3.7148
MSE	35.5435	37.7184	38.0646	38.3679

Figure 4 shows a reproduction with no additional brightness applied. Figure 5, 6 and 7 show the same reproduction with additional brightness added with per-pixel-, mean-value- and per-block-method respectively.

Figure 8 shows a cropped region of Figure 4 in order to more clearly visualize the individual Lego bricks.

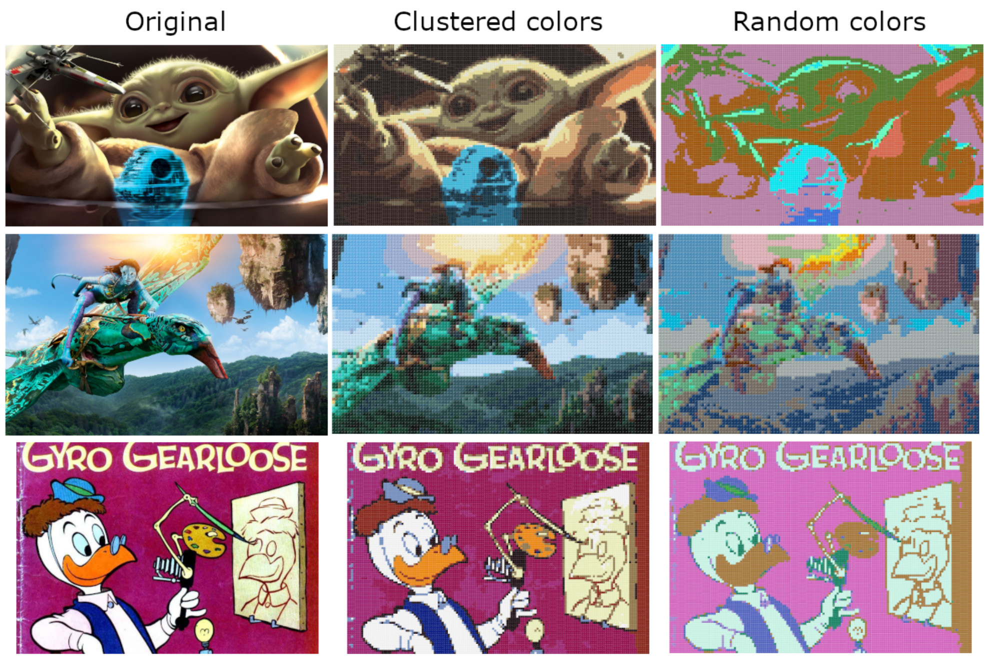

Figure 3: Example of original images (left column), with reproduction using k-means (middle column) and random colors (right column).

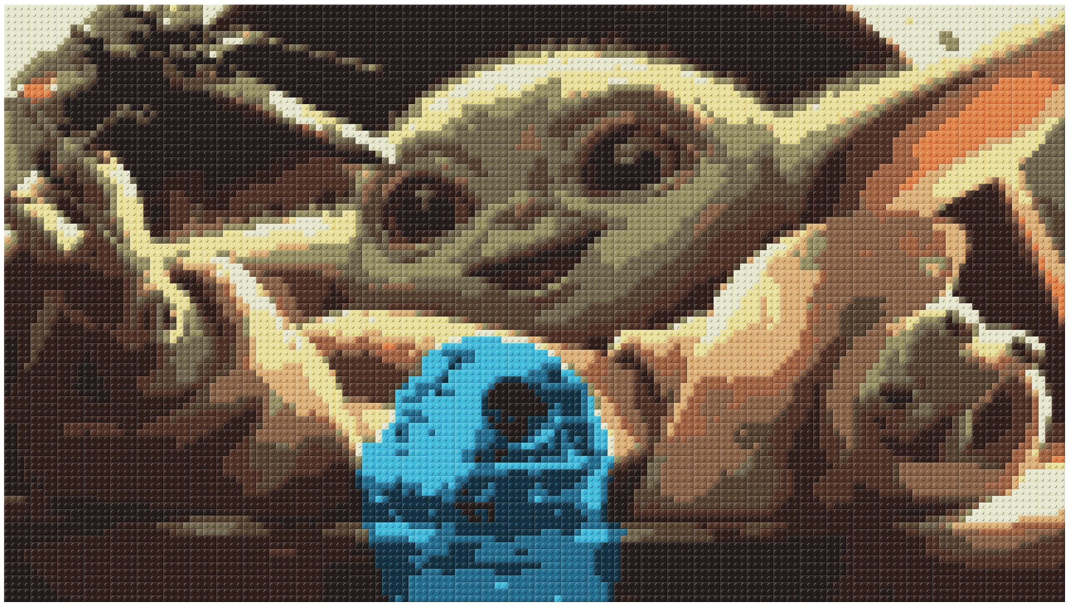

Figure 4: Regular reproduction of Grogu from the Mandalorian. Each *unit* is 24 pixels and 15 colors were selected by K-means clustering. SNR = 5.8150.

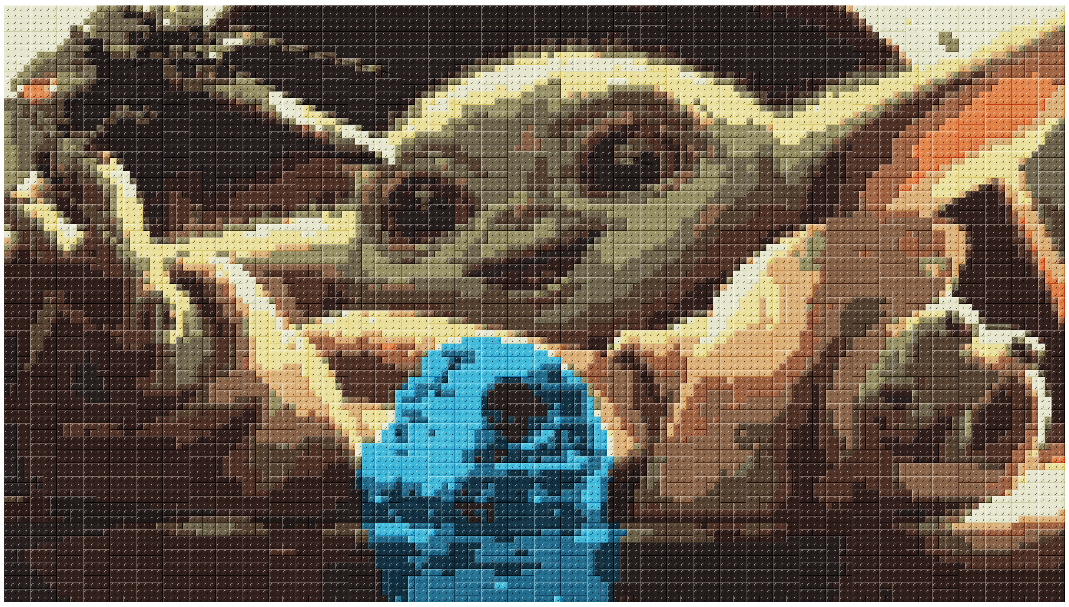

Figure 5: Per-pixel brightness applied to Figure 4. SNR = 5.3801.

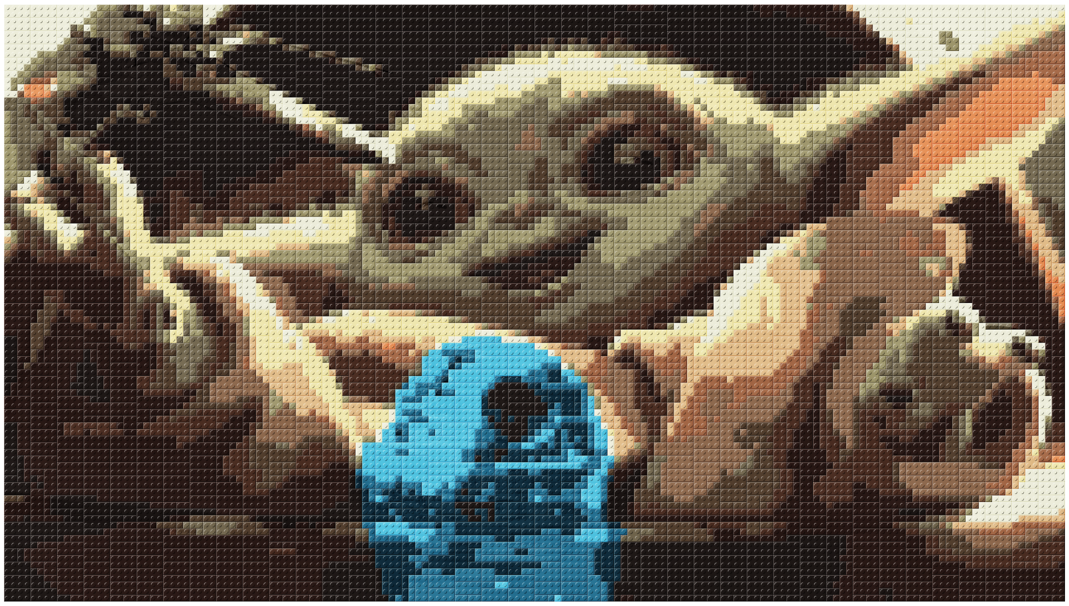

Figure 6: Mean L-value brightness applied to Figure 4. SNR = 5.2144.

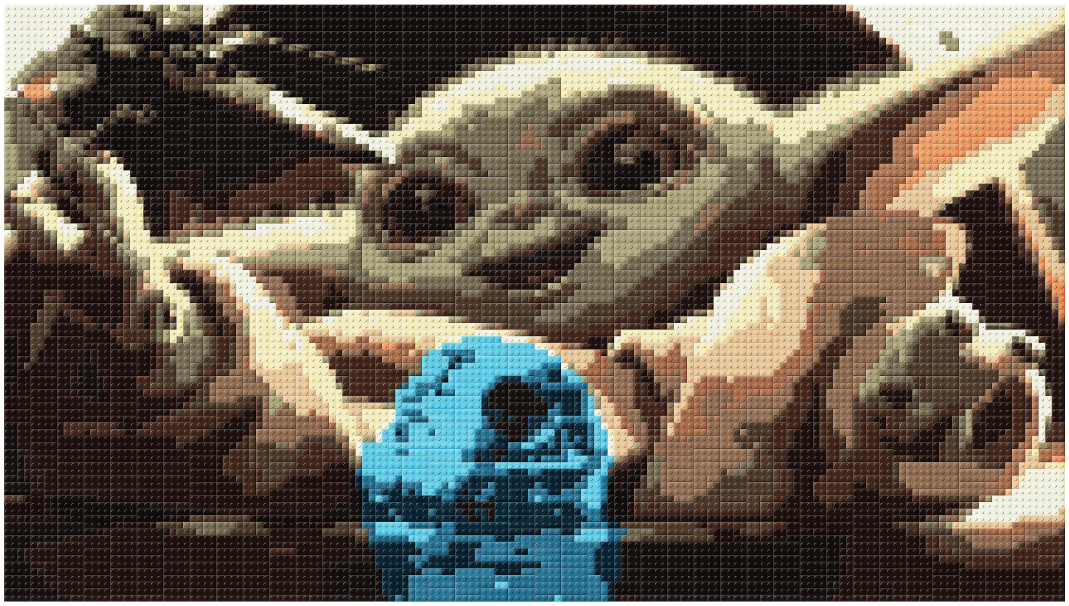

Figure 7: Per-block brightness applied to Figure 4. SNR = 5.2380.

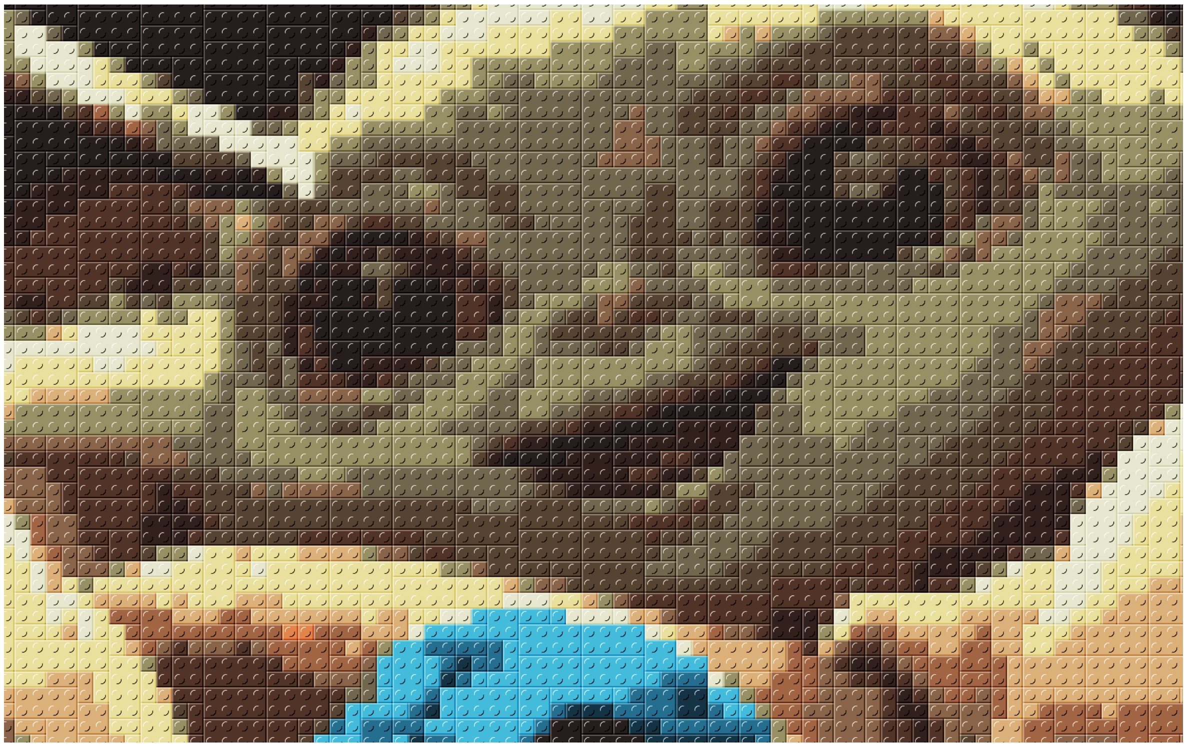

Figure 8: Cropped version of Figure 4 for more visible Lego bricks.

4 Discussion

The resulting implementation of the project fulfills the aims of the report. It takes a given image and reproduces it with Lego bricks as well as applies various methods for simulating the brightness of the original image.

As seen from the Table 1, the SNR for all images were observed to be the highest for the normal reproductions, i.e. the ones with no additional brightness. They were also seen to have overall lower mean S-CIELAB values and MSE, indicating that this version of the image was the best reproduction both objectively and subjectively. This makes sense as the normal images were solely constructed based on the closest values. As seen in Figure 4 to Figure 7, the added brightness is not very significant when seen from afar. However, a closer look shows that contrasts from the original image are visible in the reconstructions with brightness added. This may make the image appear slightly better from afar to the human eye, but spoils some of the innocent simplicity of the traditional Lego bricks.

Figure 3 shows a montage of several reproductions and it is seen that the reproduced images do depict the original images rather well from far distances. Using K-means clustering for the color selection produces much more accurate depictions but also take much longer to compute, depending on the size of the original image. This was efficiently reduced by allowing very high resolution images to be downscaled for the clustering. The random color selection required less computation but yielded significantly worse results using objective metrics.

The sizes of the bricks also seem to be well positioned with larger pieces wherever possible, however, the positioning was computationally expensive, as each type of brick required looping over all pixels in the image. Perhaps this could have been improved by other methods of positioning the pieces. Machine learning algorithms may prove useful, but would likely introduce more uncertainty, as guessing rather than using measures in the CIELAB color space would likely be more suitable then.

The sizes of the Lego bricks are of some relevance too. Using too low resolution per *unit* would cause the pieces to not resemble a typical Lego brick. A too high resolution, however, could cause low resolution images to be fully constructed by only a few bricks. Thus, low resolution images were preferably upscaled significantly. The image of Gyro Gearloose in Figure 3 for example, was upscaled six times, causing the final resolution to be drastically higher than the original image. This may or may not be suitable for some applications. There is likely no perfect solution to this, as the *unit* sizes must be large enough to depict a Lego brick.

5 Conclusion

Utilizing the euclidean nature of the CIELAB color space seem to be an efficient method to reproduce images. The results showed that brightness can be applied to the reproductions in various ways. Reproductions using Lego bricks is a rather artistic implementation, and thus, the perceived value of the reproduction may still be the highest when not adjusting brightness. That, however, is up to personal preference.

References

[Gus05] Stefan Gustavson. Empirical findings regarding the length of 1 ldu. `https://weber.itn.liu.se/~stegu/lego/LDUlength.pdf`, 2005.