Documentation: Rearranging Pixel **Nhu Quang Dang**

Summer Semester 2020

Bauhaus-Universität Weimar

Prof. Dr. Charles Wüthrich

1. Introduction

Rearranging Pixel is a long-term student project which following the project “Visualize my Picture” [WS17/18] **[1]** supervised by Charles Wüthrich at the Computer Graphics department.

The goal of the previous project is exploring the possibilities of sampling pictures not with a grid pixel distribution, but with “random” pixel distribution and proving that some sampling methods help us to gain a higher quality image than using the method with a grid-like in a photo sensor.

In this semester, I want to explore a pseudorandom number generator (PRNG), a new method for sampling pixel namely Correlated Multi-Jittered Sampling, other usage for Delaunay Triangulation and apply Alpha Compositing for Splatting method.

2. Research

After working through some papers about random sampling I understand that using random sampling might result in point clustering and will reduce the quality of the image. Therefore, a better PRNG or methods like Halton Sampling and Jittered Sampling help the image to contain pixels with a more balance distribution.

When the image is sampled, the pixels need to be splatted in order to cover the whole image. In previous work, they tested the Delaunay Triangulation algorithm but the results have visible triangle edges; it effects on the quality of the image just by the human's judge.

So I was deciding to use a normal method for splatting and use Delaunay Triangulation only for computing the biggest distance between two pixels in the image and use this distance as the size of the splatting. A splat at the sampled coordinate of the image is just a circle or square (which is bigger than 1px) with a transparent gradient and contains the color information from the sampled pixel.

But if two or more pixels are close to each other and the distances between them are smaller than the size of their splat, they will overlap on each other and cause the image to look rough. Therefore, an algorithm called Alpha Blending is applied for producing a new color blended between two or more pixels, making it impossible to distinguish between parts of the pixel's splat.

3. Techniques

3.1. Sampling

As discussed in Introduction section, I explored some sampling patterns in order to prove that sampling with grid like in common cameras might not be good for viewing the images, "it looks like looking through a prison's window". When one looks at the complex wavelet interpretation of images, the different between analog image and raster image (digital image) will be visible and the quality of analog image is way better.

In an analog camera, the light-sensitive silver halide crystals on film are distributed random, it means that the sampling pattern is good distributed in all direction, not just in two directions as on a digital camera. So this difference might be the reason for better quality on an analog camera.

3.1.1. mt19937-PRNG for sampling

In order to have a better pseudorandom number generator I used mt19937-PRNG which has a very long period $2^{19937-1}$ than rand() or srand() with 2^{32}. “Note that a long period is not a guarantee of quality in a random number generator”, but “it will take its random sequence much longer to repeat itself”. When we generate random positions for sampling, we have to check if the pixel is already sampled at this position, so the long period of mt19937 might give us better performance in this case; beside that the distribution of pixels on the image might be more balance.

3.1.2. Correlated Multi-Jittered Sampling

In order to prevent point clusters in the sampled image, I decided to use Correlated Multi-Jittered Sampling, the method presented by Pixar Technical **[2]**.

This method is based on quasi-Monte Carlo sequences to generate uniformly distributed samples. But some methods that use Monte Carlo integration can also create structured artifacts. So, they introduced a modification to normal multi-jittered sampling, such as a way to generate patterns with arbitrary sample counts without noticeable gaps or clumps. They claim that this method will combine the robust foundation of jittered sampling with the benefits of quasi-Monte Carlo.

Firstly they divide the image into equal area cells using $m * n$ grid, with $m * n = N$, and N is the number of samples. But for simplification I just find the square number which are bigger than N and closest to N, then factorize this number to find $m = n$, *Fig. 1*. So these cells will create a square that is bigger than the image, but we can still sample on this square and then extract the positions which are not in the image.

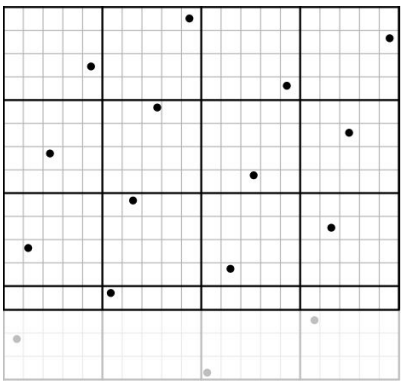

Figure 1 **[2]**

After that, they locate these samples in an ordered in each row of cells, this order called “canonical” arrangement, *Fig. 2*.

In the next step they shuffle the arrangement by shuffle the X coordinates in each column and then Y coordinates in each row. The result of this step looks pretty good, but the distribution of samples is still clumpy and uneven, *Fig. 3*.

So they make a small change to this method, in each column they apply the same shuffle to X coordinates and in each row they apply the same shuffle to Y coordinates. This change clearly reduces the clumpiness and unevenness, *Fig. 4*.

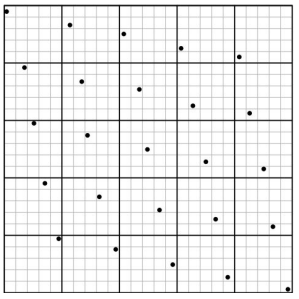

Figure 2 **[2]**

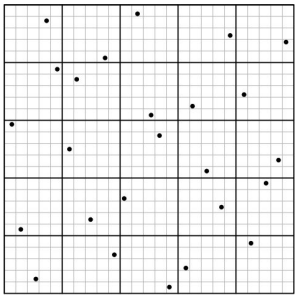

Figure 3 **[2]**

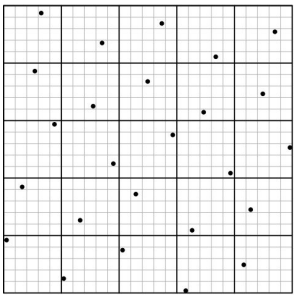

Figure 4 **[2]**

In the Comparison section, I will compare the result of this method with normal random sampling and Halton Sampling.

3.2. Interpolation

3.2.1. Naive Splatting

In the output image, the pixel was not sampled has no information, so it is black. With the splatting method each sampled pixel will be splatted into a square or circle with specify size in order to cover the pixel without information. The problem is that we have millions of pixel and many of them should be splatted, how could one determine one reasonable or best-fit size.

3.2.2. Delaunay Triangulation

To solve the problem of Naive Splatting, I came up with an idea of using Delaunay Triangulation to find the biggest distance between two pixels on the image.

Delaunay Triangulation is the algorithm for creating a set of triangles from a given set of points in a plane such that no point in the set is inside the circumcircle of any triangle in the triangle set **[3]**.

After applying Delaunay Triangulation to the set of sampled pixels, my program go to each triangle and find out which edge is the one with the biggest length in that triangle, then compare all of them to specifies the biggest distance between two pixels on the image. Finally, I use this distance as the size of Splatting.

3.2.3. Alpha Blending

As you can see that the result of Naive Splatting is rough. Because if the size of the splat is bigger than the distance between any two pixels, two splats will overlap on each other. Whichever is splatted later covers a part of the other. If we just apply the transparent gradient to the splats, they will still overlap on each other and the result image could not be much better.

So I use Alpha blending (or Alpha Compositing) **[4]**, **[5]** to combine a translucent color of the latter sampled pixel with the color of the pixel that was sampled before. This process combines also the transparent (alpha) of two or more pixels that overlap on each other, *Fig. 5*.

For the transparent gradient of the splat, transparency is done by evaluating the cos^4 function because the transparency is 8-bit. This function will be scaled to fit the color interval $[0-255]$. The sampled pixel is the center of the splat and therefore it has the maximal value of cos^4 function (1.0), the color of the pixel around the center will be computed by applying distance (in pixel) from itself to the center to the cos^4 function. The pixel that are at the corner of the splat should have the value 0.

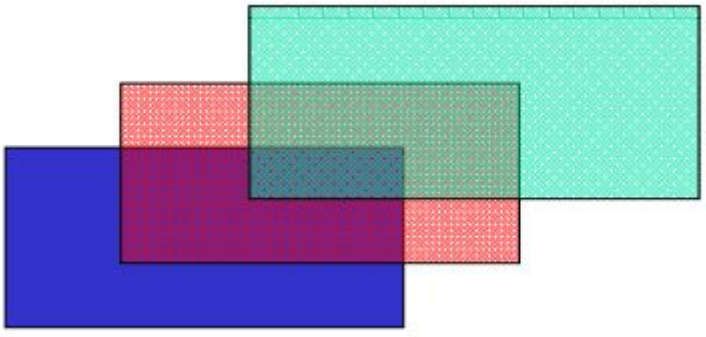

Figure 5: Alpha Blending **[5]**

4. Comparison

4.1. Normal random function and mt19937-PRNG

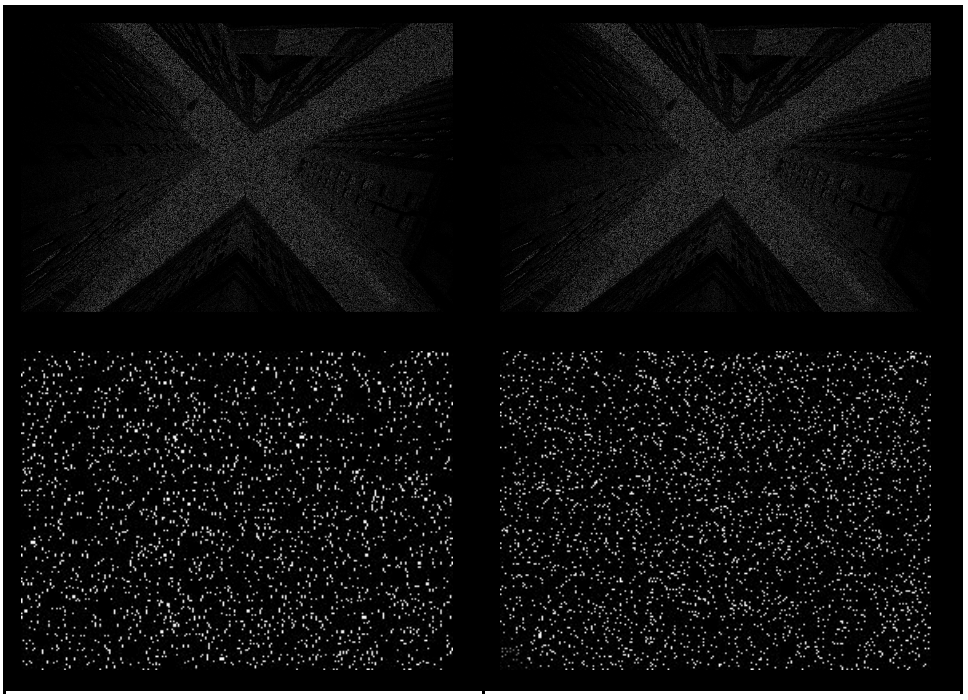

Figure 6: rand() vs. mt19937

The long period of mt19937 does affect the result to have less clustering, but it takes a longer time to generate random values. On a big image, one could not notice a significant improvement, so I decided to use rand() function. But if the performance of this method could be improved, this is one option to gain better quality.

4.2. Random, Halton and Correlated Multi-Jittered Sampling

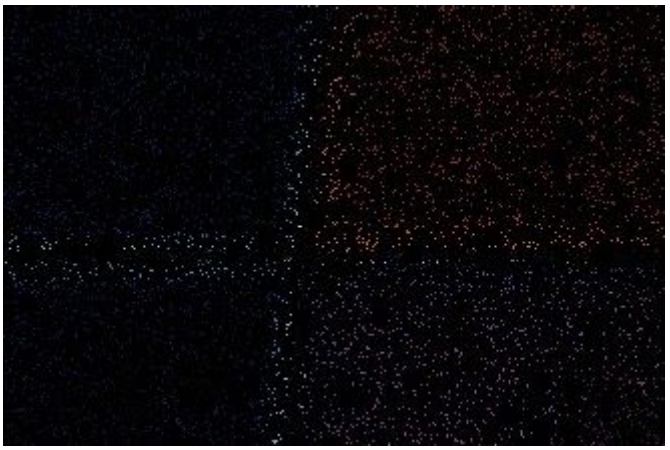

Figure 7: Random sampling (200% crop)

Figure 8: Halton sampling (200% crop)

Figure 9: Correlated Multi-Jittered Sampling (200% crop)

Fig. 7 shows that Random sampling creates clusters, some of them has no image information. A Big cluster without image information is hard for the splatting algorithm to cover. Both Halton sampling *Fig. 8* and Jittered sampling *Fig. 9* have very balanced distribution of pixels. On the whole image of both methods *Fig. 11, Fig. 12*, the sampling pattern is still visible. But one advantage of Jittered sampling is that it reproduce the lines and edges way better than the Halton sampling. This effect can be seen even more clearly with Splatting (*Sec. 4.4*).

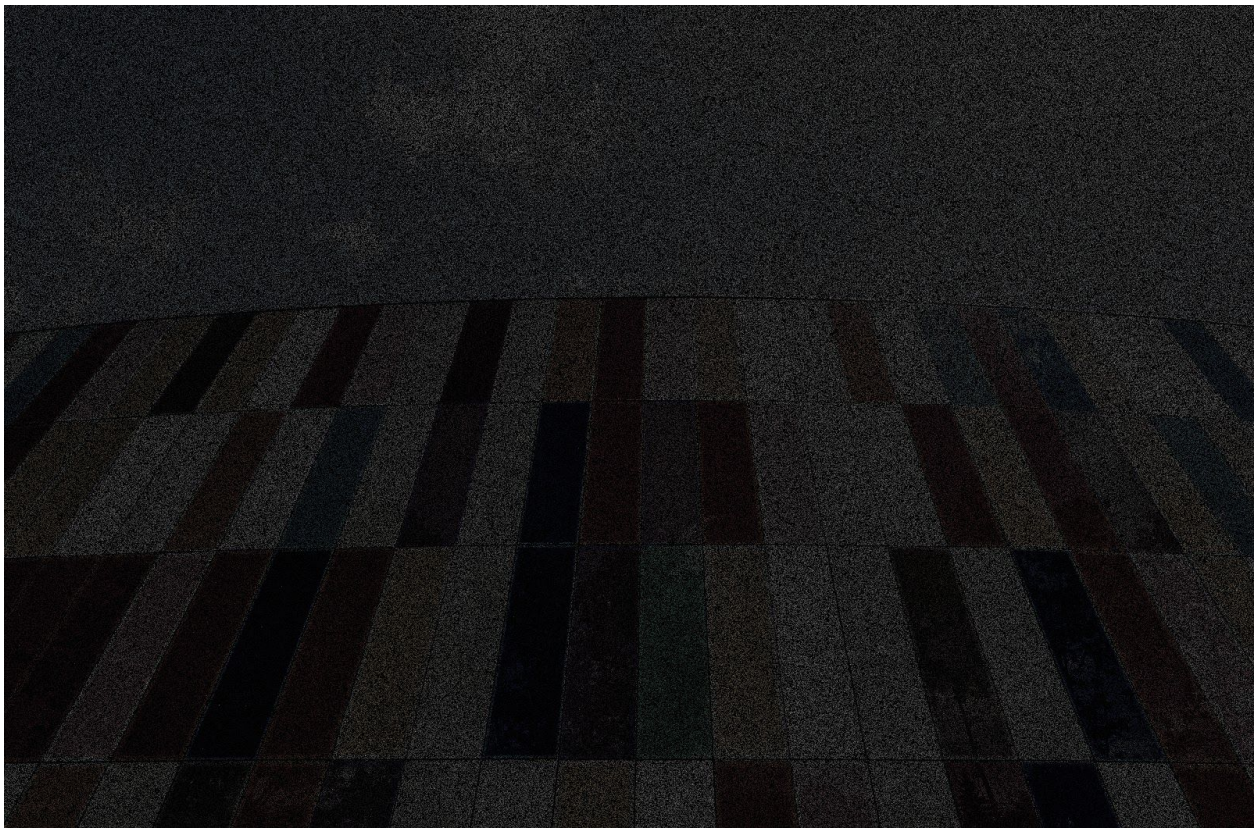

Figure 10: Random Sampling

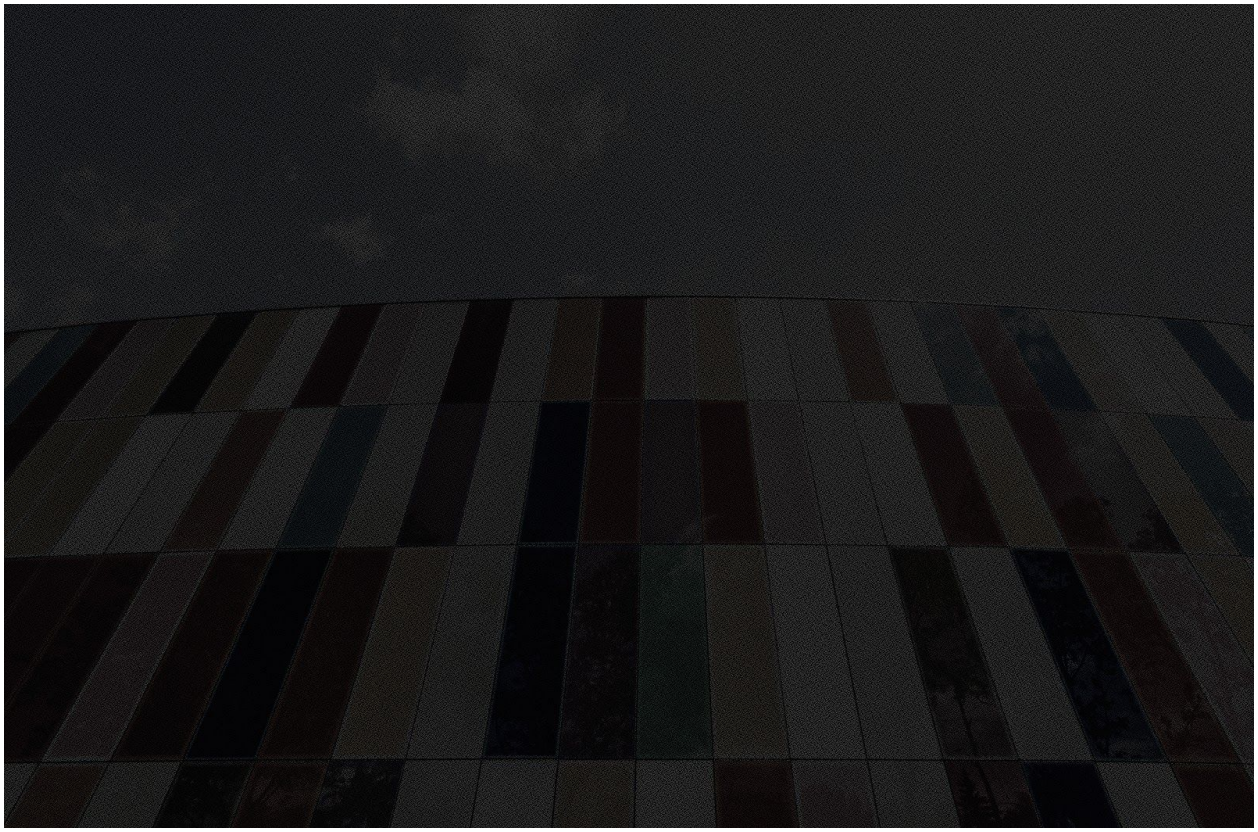

Figure 11: Halton Sampling

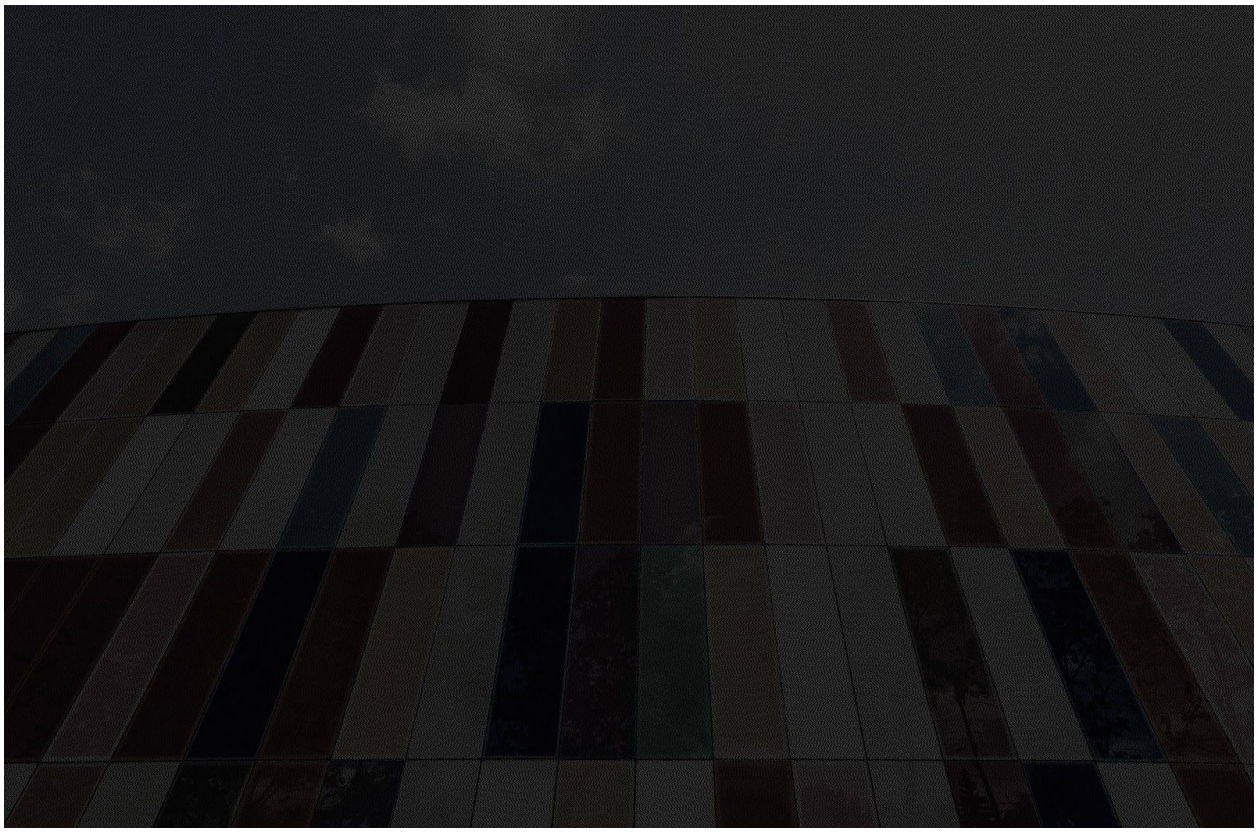

Figure 12: Jittered Sampling

Using Delaunay Triangulation as mentioned above one could determine the splat's radius for each sampling pattern. Table 1 shows that splat's radius for Halton sampling is the smallest and the one for Random sampling is the biggest radius.

Sampling Pattern	Random	Halton Sequence	Multi-Jittered
Distance (px)	15	11	12

Table 1: Maximal distance between 2 pixels for 10% of the pixels

4.3. Naive Splatting and Splatting with Alpha Blending

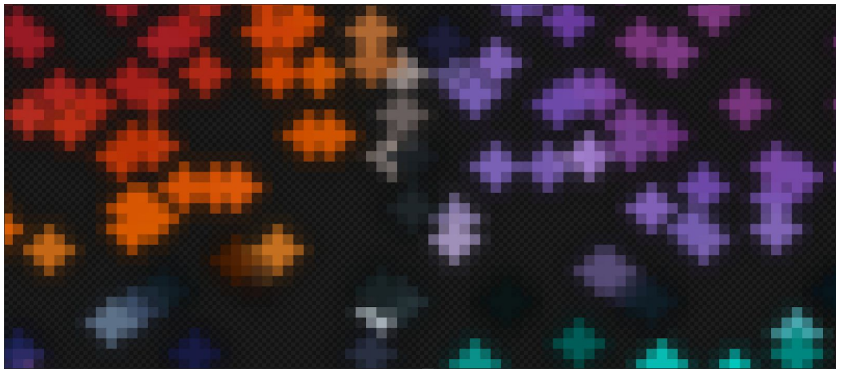

Figure 13: Naive Splatting in detail

Figure 14: Alpha Blending in detail

Fig. 14 shows the result with Alpha Blending technique in comparison with Naive Splatting *Fig. 13.* With Naive splatting method the splatted areas are overlapped on each other without blending effect, one could clearly distinguish between two or more pixels in the overlapped part. The latter sampled and splatted pixel is placed on top of the other, this leads to missing information of splatted area.

4.4. Final results with different sampling patterns

Figure 13: Halton Sampling with 10% of pixels, splat radius: 5px

Figure 14: Jittered Sampling with 10% of pixels, splat radius: 5px

Figure 15: Crop 100% from Halton Sampling’s result

Figure 16: Crop 100% from Jittered Sampling's result

5. Resume

Showing the benefit of using random arranging of pixels of a photo sensor is the main goal of this project. My own goal is exploring another way to sample pixels and how to adjust the sampled pixels to obtain a better quality output image. Correlated Multi-Jittered Sampling method produces a random noise that looks more natural, and reduces the clumpiness and unevenness of the pixel distribution. More than that, the output image from Correlated Multi-Jittered Sampling have significant quality improvement in order to recreate lines and edges. Alpha Blending increases the quality of the result by creating a color and alpha compositing between two or more overlapped pixels. By using Delaunay Triangulation to determine the maximal distance between two pixels on the image, one can easily specify the radius for splatting in order to cover the whole image. Combining these techniques and apply them to the previous work one could produce a better quality image and might have a faster execution time. This result shows that, randomly sample images is possible and random arranging pixels of a photo sensor will gain excellent image quality without the artifact of grid sampling.

6. Future Work

The work of this project focused on the quality of the output image, but did not cover the performance part yet. Only new usage of Delaunay Triangulation might accelerate the speed, but the Delaunay Triangulation itself takes a long time to execute, so another way to determine the splat's radius will be the first thing to consider in the future work. Alpha blending is applied to each pixel in the same way and with the same splat's radius. This method increases the number of overlapped areas on the image and makes the output looks a bit unsharp. By changing the way Alpha blending applied to each pixel or by finding a way to determine how each pixel should be handled, the result will have a better quality.

7. Reference

[1] - 2018 - Phil Jungschlaeger, Artur Solomonik, Muhammad Muksitur Rahman. Visualize My Picture

[2] - 2013 - Andrew Kensler. Correlated Multi-Jittered Sampling

[3] - 2015 - Satya Mallick. Delaunay Triangulation and Voronoi Diagram using OpenCV. https://www.learnopencv.com/delaunay-triangulation-and-voronoi-diagram-using-opencv-c-python/

[4] - Wikipedia. Alpha blending. https://en.wikipedia.org/wiki/Alpha_compositing#:~:text=Alpha%20blending%20is%20the%20process,completely%20transparent%20to%20completely%20opaque.

[5] - The Ohio State University. Transparency Basic Alpha Blending http://web.cse.ohio-state.edu/~parent.1/classes/581/Lectures/13.TransparencyHandout.pdf