EEC 289A Assignment 2

Non-parametric Data Synthesis

Randall Fowler

ECE Department, UC Davis

rlfowler@ucdavis.edu

Introduction:

This project is an attempt to implement the texture synthesis algorithm published by Efros and Leung [1]. Several images are provided with varying textures, and the goal is to synthesize new images using these examples. Synthesis represents the process of selecting unfilled pixels and determining a similar neighborhood of pixels for filling. It is assumed that the distribution within a similar neighborhood is the only information needed to predict new pixels. By iteratively generating new pixels, an entire new image can be created with similar texture.

Methodology:

The texture synthesis algorithm that Efros and Leung used in their paper discusses a modeling strategy based on Markov Random Fields (MRF). It is assumed that the probability distribution for a pixel given the values of its spatial neighborhood is independent of the rest of the image. For pixels that are unknown, the closest neighborhood similar to the known pixels can be referenced.

Patches are taken from an example image to gather information relating to the shapes and colors found in the image. The size of the patches will be denoted as kernel size and should be similar in scaling to common features. Kernel size will greatly vary depending on the resolution and changes in the image. Collected patches from the example image will be a reference when searching for the closest neighborhood to a target pixel.

To get started with generating a new image, a small patch is taken from the example image to provide a starting point. This patch can be nearly random, but it should have a similar distribution to a patch in the image. Pixels will be synthesized one at a time, and selected target pixels will be chosen based on distance from the starting patch. Figure 1 shows a potential map for selecting a target pixel as the targets will be selected from the brightest points on the map. The only zero points on the map would be the center as this is the placement of the starting patch. As pixels are generated, the map will replace the location with a zero value.

When a target pixel is selected, the neighborhood around the pixel will be referenced as the target patch. This patch will also have the kernel size to properly compare with example neighborhoods. Since not all pixels in the neighborhood will be known, the known pixels with zero values on the potential map will become the mask. Masking will remove the unknown pixels when comparing the difference between the target patch and all example patches. There should weighting on the pixels' differences as the pixels closer to the center will have more significance when generating a new pixel. One method to provide this weighting is to apply a gaussian filter to the pixels' distribution.

After finding the difference between all example patches with the target patch, the closest patches will be considered in a random selection. Closest refers to an arbitrary threshold selected, and the distance will determine the weighting for the random selection. The center pixel of the selected example patch will become the generated pixel.

Experiment:

Synthesized Textures:

This experiment proved to be more challenging than initially perceived. The search space for hyperparameters was explored, but many revisions were made while the exploration took place. These hyperparameters would include the threshold for closest example patches, variance for the gaussian kernel, size of the windowing kernel, and attenuation of pixel values. Revisions were mainly within the method for selecting the most likely example patch.

As time was a concern, synthesized images were given a size of 530 by 530 pixels, but only 10% of the image was generated. This provided a large enough window to visualize the textures, but as the time complexity was large, only the center of the image was generated. A smaller image size would be fine, but a large enough portion of the image would need to be generated for comparing results.

For the first series of tests, a picture of rocks was selected to provide an obvious texture to replicate. Figure 2 contains image 7 from the provided images, and figure 3 shows the first results. These first results were not good, and the search through hyperparameters included variance between 0.5 and 3, threshold between 40% to 80%, and window size of 9, 15, and 21. Similar results were found when changing the location of applying the gaussian kernel to before or after the squared differences. At this point, hyperparameter tuning did not seem to provide obvious improvements.

For each pixel, there should be a value for red, green, and blue to make up all the possible colors. One potential bias could come from the alpha value as darker or brighter pixels may be considered over the actual color. From this point, this alpha value was removed for all future tests, and a sweep over all image textures was conducted. It was believed that a window size of 15 would provide a reasonable search space to capture necessary features, starting patch size was set to 3 and 15, and variance was set to 1. Figure 4 demonstrates the newly generated patches, but these results were also unsatisfactory.

After further evaluating the method for selecting a close example patch, thresholding was simplified to be proportional to the smallest distance without any attenuation. With small alterations and influenced hyperparameters from others, figures 5 through 16 present satisfactory synthesized images. Variance was set to 6.4 while window size remained at 15, and the threshold was set to be 80% larger than the minimum patch difference. Going one step further, the complement of the gaussian kernel was applied to the squared difference between patches, and the threshold percentage was the percentage between the maximum and minimum distances between patches. Figure 17 shows this adjustment for image 7 with a variance of 1, 3, and 6. Previously, the variance needed to be large enough to treat the pixels relatively the same because the gaussian kernel was applied on the distances. It’s possible that the previous implementation could have produced better results with a higher variance, but the final implementation produced the best results with no known mistakes.

MNIST Dataset:

One final test for exploration included generating numbers from the MNIST dataset. Since these images were much smaller with a size of 28 by 28 pixels, the kernel size was reduced to 9, but further reduction could be much better. Instead of selecting a single image for example patches, all images for a given label were selected for extracting patches. Only images with the same label should be grouped together to allow a distribution for the correct number generation. Figure 18 shows the results of the generation, and numbers above 3 are considered good. The numbers 1 and 0 are far from human predictable, but it appears that most other could be made out. With hyperparameter tuning, the number generation could be much better.

Conclusion:

The final results for synthesizing images were promising. Hyperparameter tuning does not have a major impact on the performance of the algorithm, but this is difficult to compare with human perception. When applying this method to MNIST data, most numbers were generated with room for improvement.

This project was challenging because of the mistakes when selecting a similar patch. When comparing a neighborhood of pixels to a target with all example patches, there is different methodology for selection techniques. Where the gaussian kernel is applied does not matter as much as the thresholding and weighting of patch selection. This gaussian kernel aims at reducing the impact of the further away pixels, but this was not done correctly till later stages of the project.

Appendix:

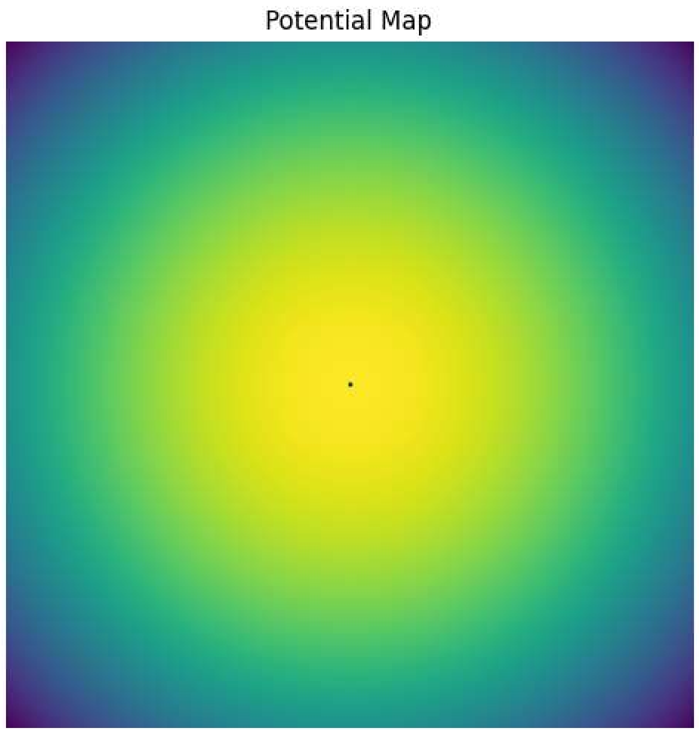

Fig. 1: Target Pixel Values

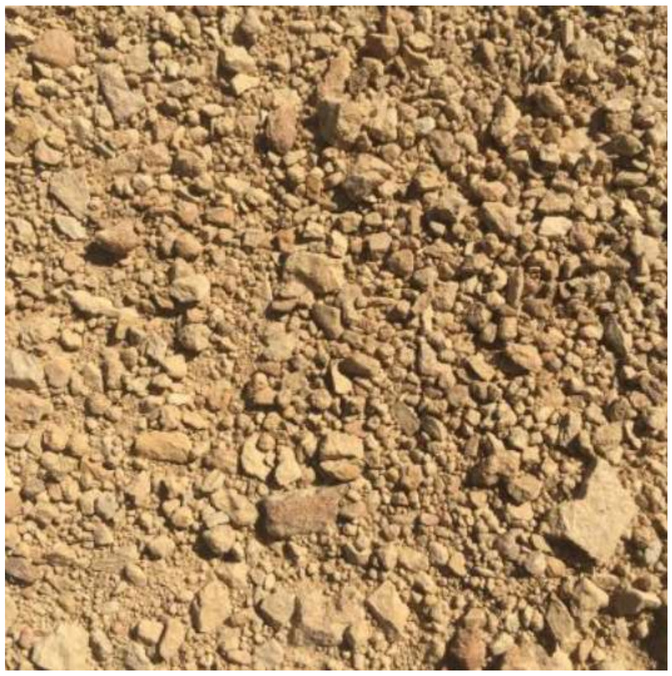

Fig. 2: Image 7 Rocks

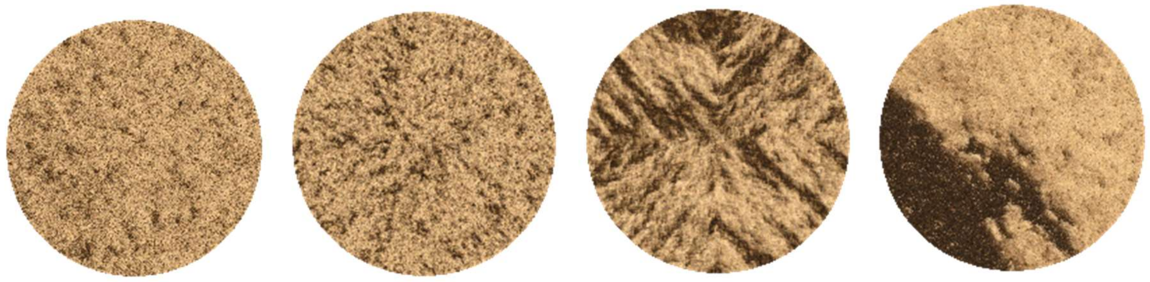
Fig. 3: First Results from Image 7

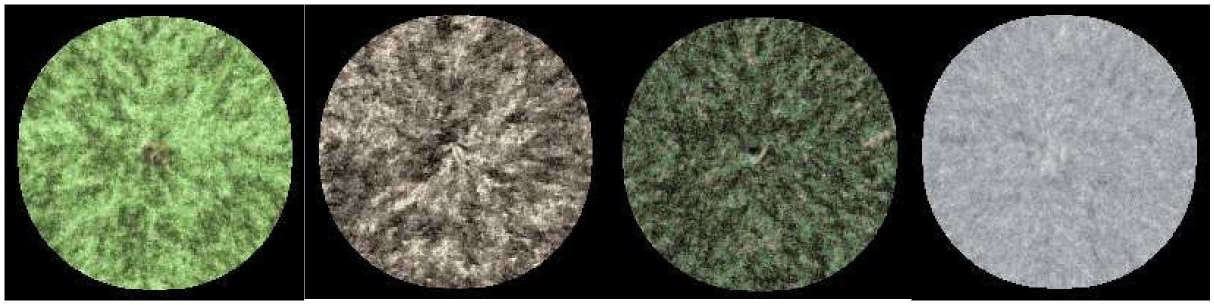
Fig. 4: Second Results from various textures

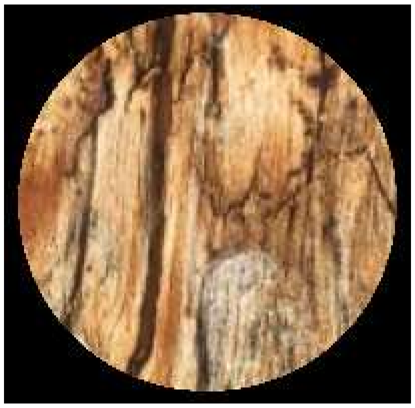
Fig. 5: Synthesized Image 1

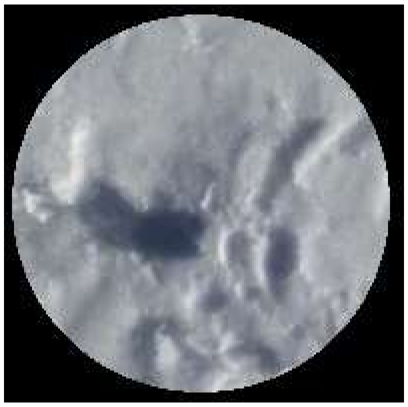
Fig. 6: Synthesized Image 2

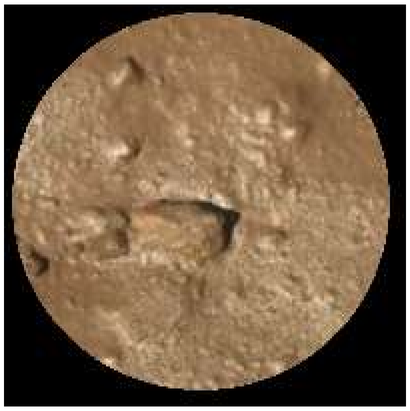
Fig. 7: Synthesized Image 3

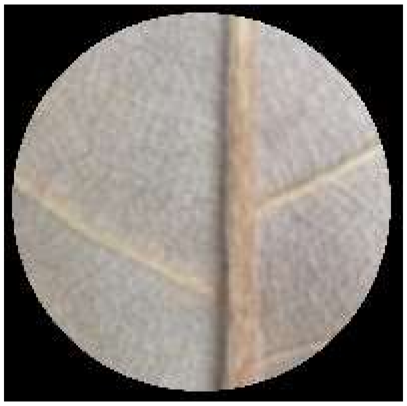
Fig. 8: Synthesized Image 4

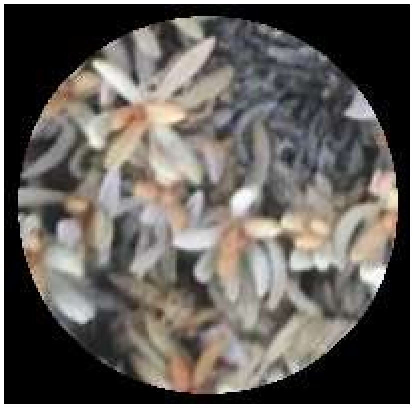
Fig. 9: Synthesized Image 5

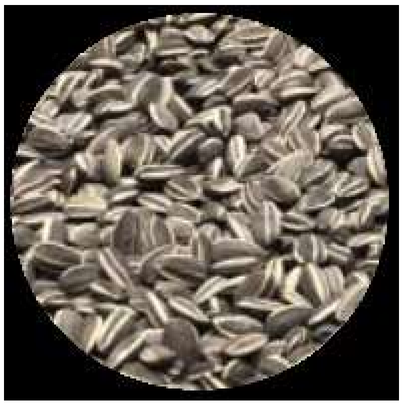
Fig. 10: Synthesized Image 6

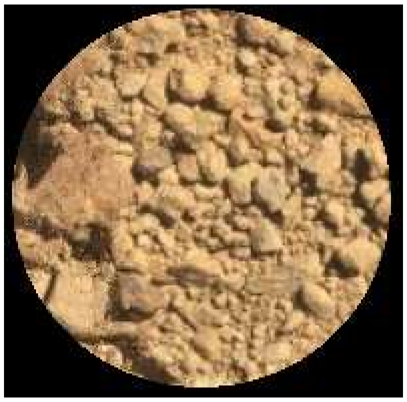
Fig. 11: Synthesized Image 7

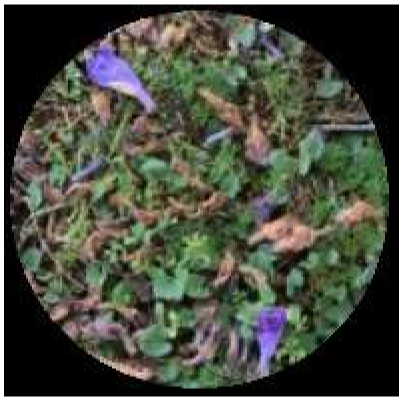
Fig. 12: Synthesized Image 8

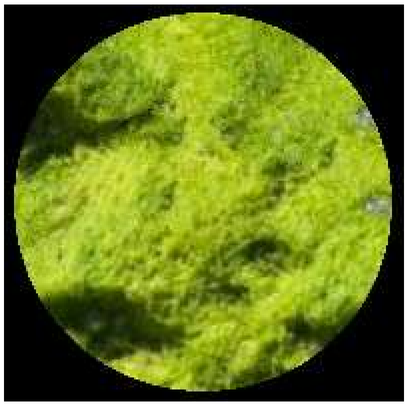
Fig. 13: Synthesized Image 9

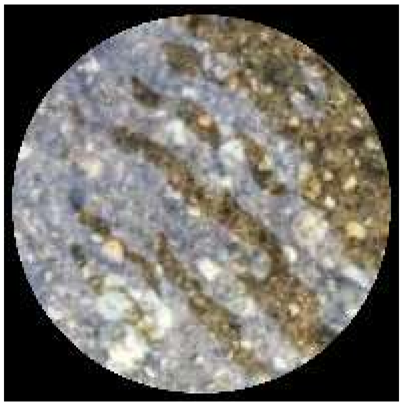
Fig. 14: Synthesized Image 10

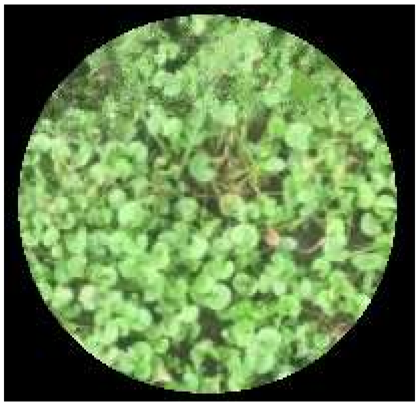

Fig. 15: Synthesized Image 11

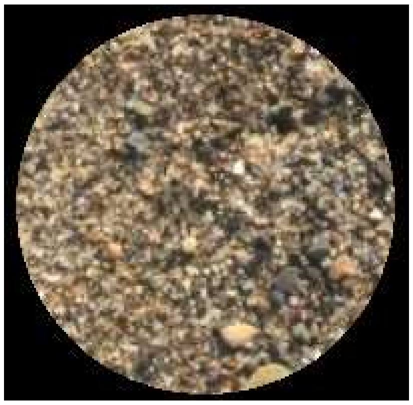

Fig. 16: Synthesized Image 12

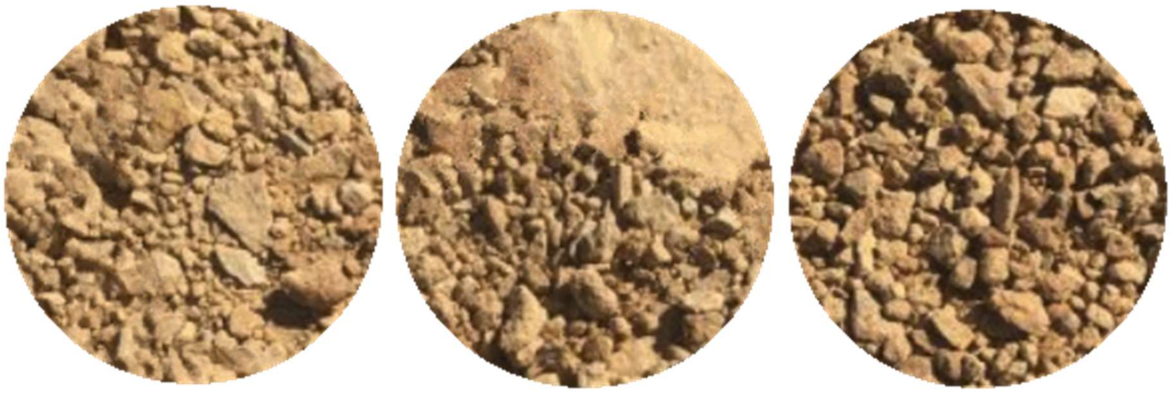

Fig. 17: Synthesized Image 7 with variance of 1, 3, and 6 respectively.

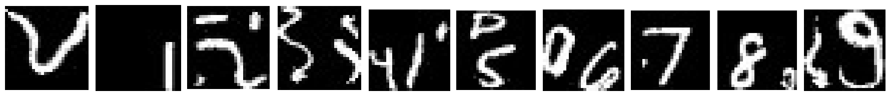

Fig. 18: Synthesized MNIST Dataset from 0 to 9

References:

[1] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 1033-1038 vol.2, doi: 10.1109/ICCV.1999.790383.

[2] Chen, Yubei, (2024). "Generative Models – Autoregressive Models," Lecture Notes.

[3] Github Repository: https://github.com/Th3RandyMan/Non-Parametric-Data-Synthesis/tree/main