CS445 Final Project Report

Texture Transferring with Multiple Sources

ruisil3, xiaosen2, zihengc2

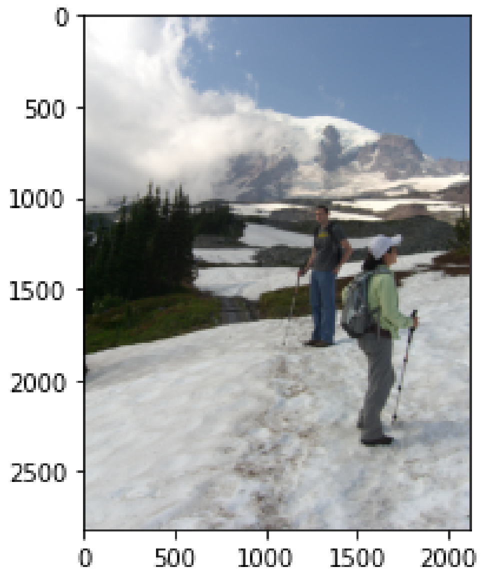
GitHub: https://github.com/JackZihengChen/CS445-Final-Project
README: https://github.com/JackZihengChen/CS445-Final-Project/blob/main/README.md

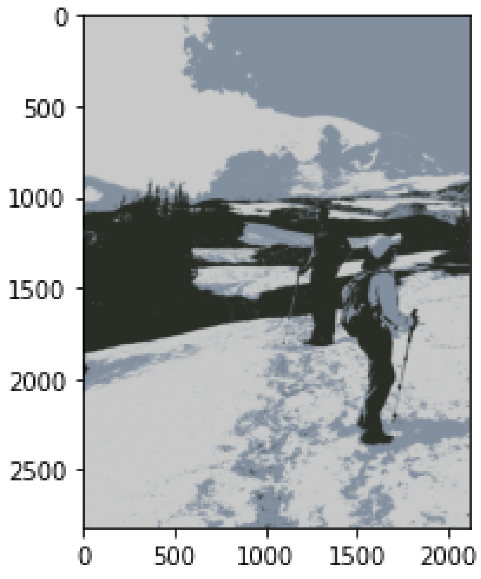
Motivation and Impact

Originally, the motive behind this project was to experiment more on texture transferring as a continuation of MP2 since we were intrigued by the procedure taught in class. We successfully expanded the idea by transferring textures from multiple sources, which yielded interesting results. The general impact of our project, besides producing pictures of unique art styles, is generating more visually pleasing and realistic pictures with limited input by combining image segmentation and image quilting.

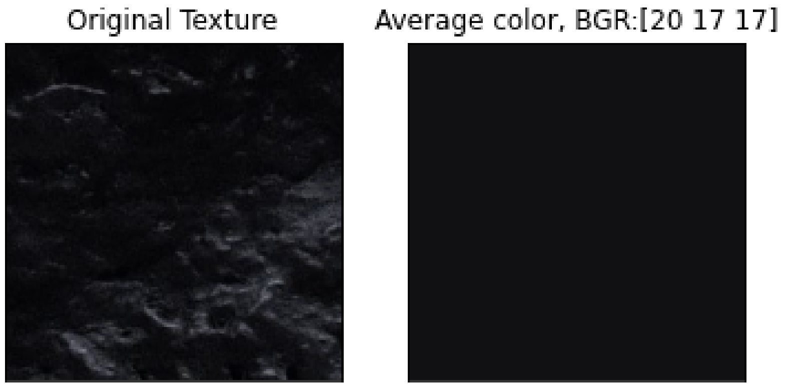
Approach

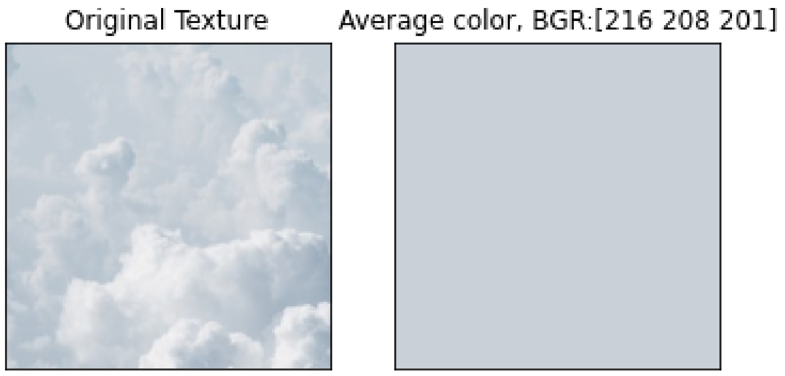
In order to transfer textures from multiple sources, we first need to determine the corresponding segments of an image. For this process, we chose the K-means algorithm. It allows us to break the image into segments with different mean colors. Depending on the color of a pixel, that pixel would have the closest cluster mean color. The cluster mean colors also get updated depending on the colors of pixels that are in that current cluster. After some iterations, the amount of update of the cluster means would level off, and every pixel belongs to one of the clusters of a cluster mean. The number of means is a hyperparameter and should be set accordingly, and we arbitrarily set it to be 3 since it is concise while the image remains informative. The algorithm is robust and works surprisingly well. As an example, here are the results from a provided picture:



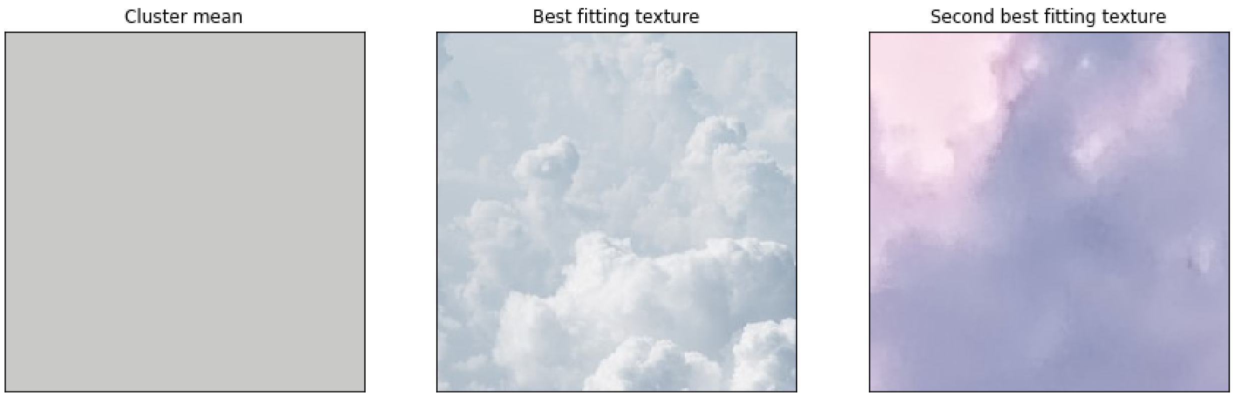


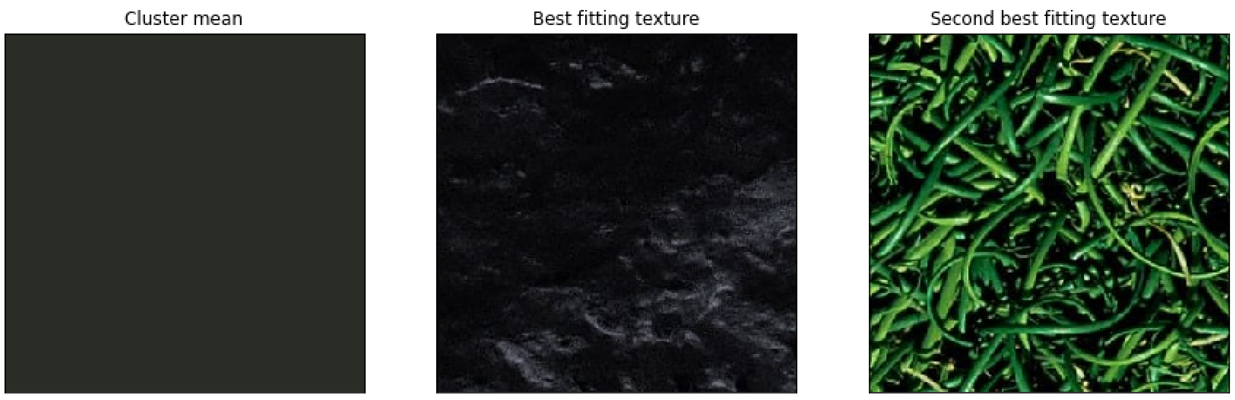
After breaking the image into segments, we then chose the texture that we would fill into each segment. After collecting some textures of a wide range of color, we first compute the average color of a texture by taking the mean of each of the rgb channels. As a result, here are some of the average colors of textures:

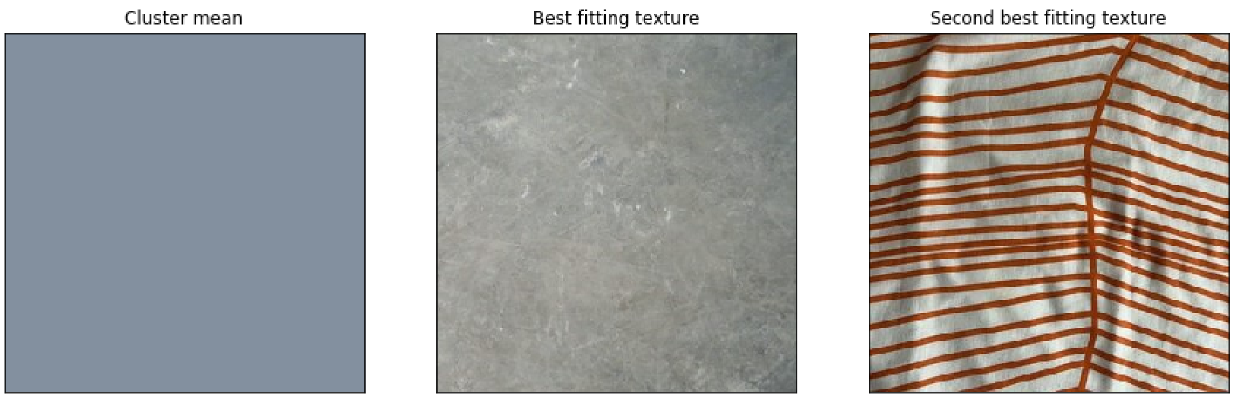




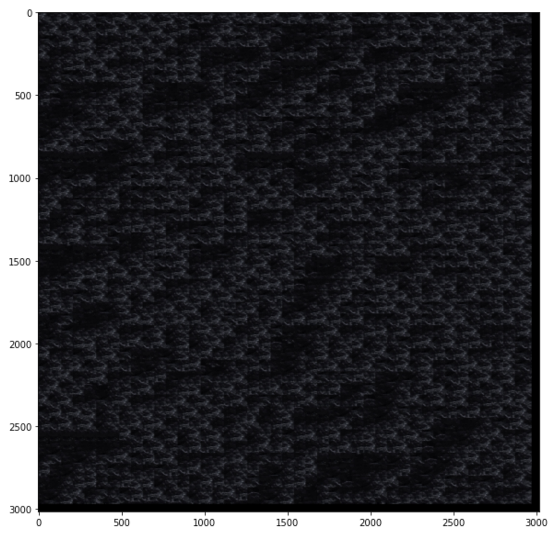
Then, for each segment color, we apply the texture with the closest color by analyzing the difference between the input color and average color of each texture, and we choose the texture with the smallest Euclidean difference while treating the color of a pixel as a vector. Since the difference of rgb color space does not reflect our perception well, we compare the colors in LAB color space. Here are the best and second best matches of a given color:

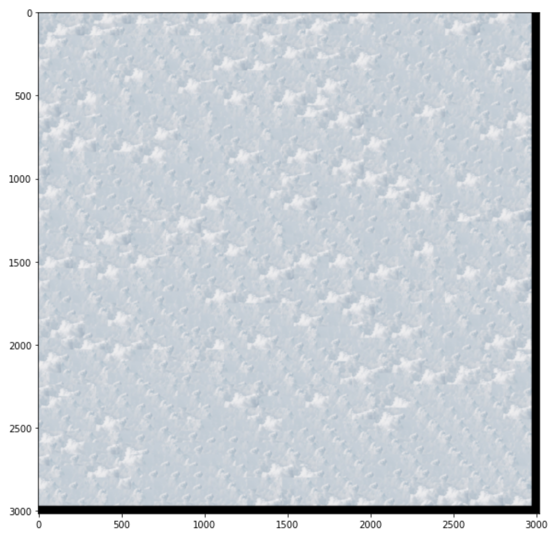




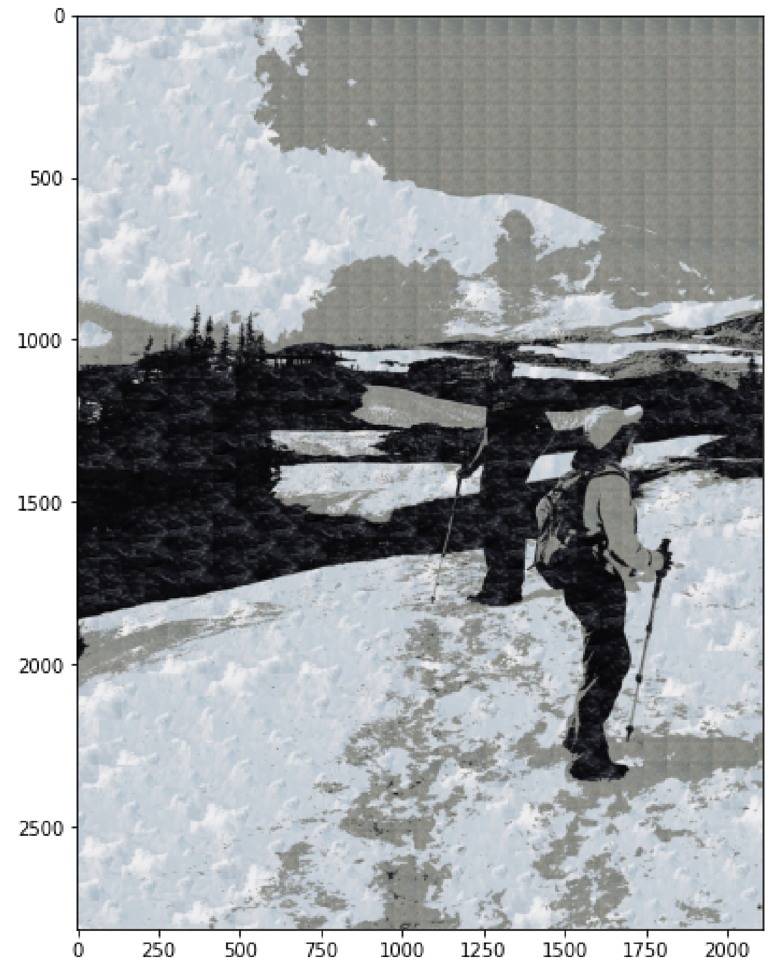


Next, we generate the texture using the simple technique and seam finding technique practiced in MP2. Our goal is to have the generated texture to be larger in size than the input image. After multiple trials and consideration, we recognized that the simple method was the most stable and yields pleasing results consistently, so it was chosen over the seam finding method. Here are a few generated results:



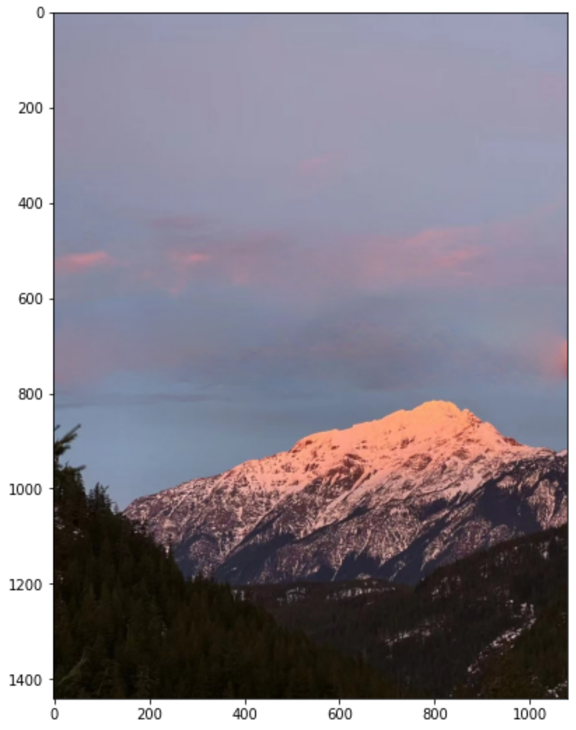


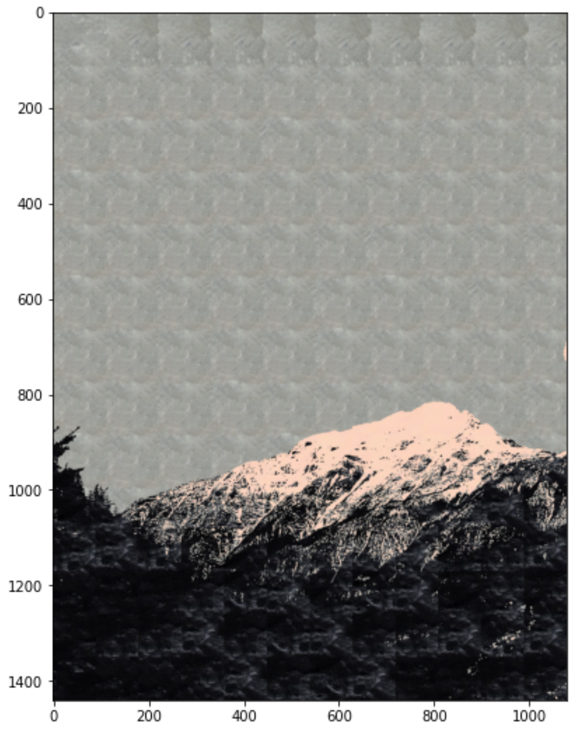
Finally, we construct an image using the textures while using the image segmentation as a mask to determine which texture should be visible on the final image.



Besides an interesting art style, as mentioned above, our project enables us to construct a visually pleasing image with minimal information, which essentially are the segmentation information and texture patches that are 300x300 in size. Compared to normal output from the k-means algorithm, the addition of textures added some depth and realism to the final outcome.

• Other Results





Implementation details
Similar to MPs, our final project utilized the cv2 and matplotlib.pyplot modules.
The k-means example we followed can be found here:
https://docs.opencv.org/4.x/d1/d5c/tutorial_py_kmeans_opencv.html
All pictures used in the project were taken by the authors or have sources listed in sources.md.

Challenge / innovation
Some difficulties that came up along the way include finding a wide range of textures of characteristic colors and determining how close two colors are based on human perception. Overall, we think our project should earn 17/20 points for innovation. Although we did not come up with completely new implementations that are paper-worthy, we incorporated ideas that are not fully explained in the lecture, like k-means.