Name (netid): Johnathon Small (jssmall3)
CS 445 - Project 2: Image Quilting

Complete the claimed points and sections below.

Total Points Claimed	**[135] / 175**

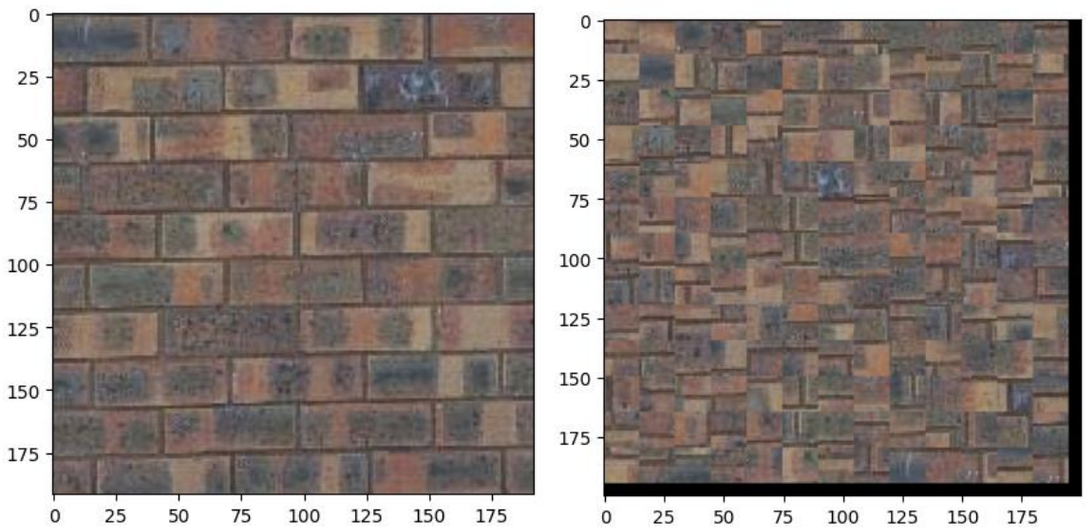
Core

1. Randomly Sampled Texture	[10] / 10
2. Overlapping Patches	[20] / 20
3. Seam Finding	[20] / 20
4. Additional Quilting Results	[10] / 10
5. Texture Transfer	[30] / 30
6. Quality of results / report	[10] / 10

B&W

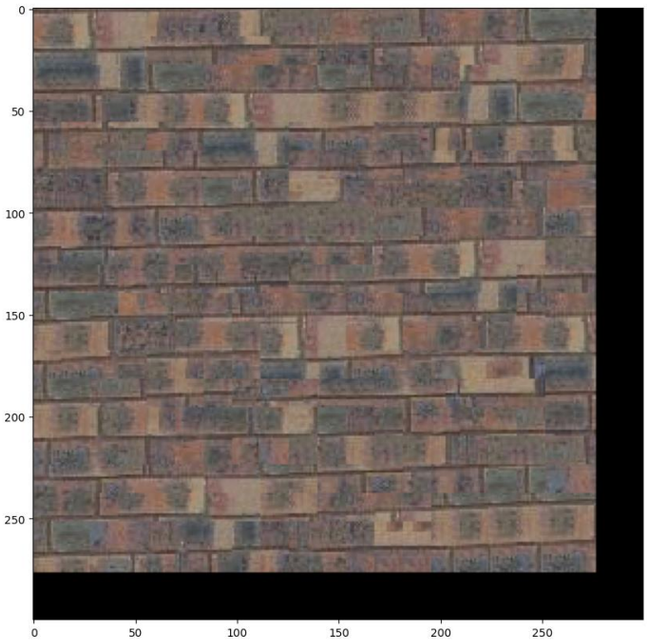
7. Iterative Texture Transfer	[15] / 15
8. Face-in-Toast Image	[20] / 20
9. Hole filling w/ priority function	[0] / 40

1. Randomly Sampled Texture



The images above were generated with a target output size of 200x200 and a patch size of 15. Random patches were taken from the texture (left) and used to assemble a new texture (right). The brick pattern has a lot of proportion data that matters at a more macro scale, which leads to the poor reproduction in the output image as edges do not line up meaningfully.

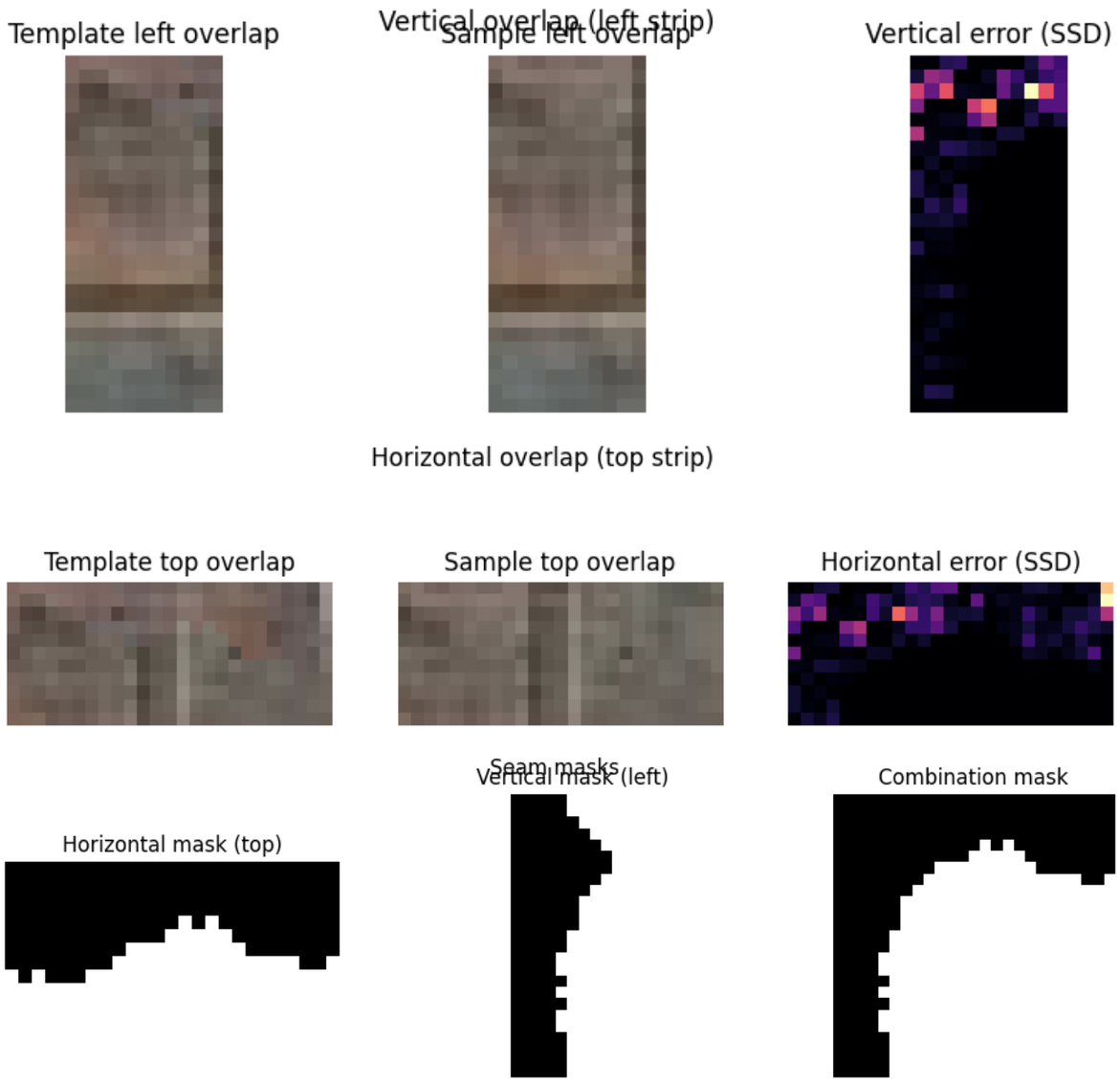
2. Overlapping Patches



This image was generated with a patch size of 25, overlap of 11, and tolerance of 5. It preserves significantly more of the low frequency information from the input texture and, from a distance, appears to be a brick pattern. However, there are still misalignments and edges created by the overlapping process that make the result less suitable under close inspection.

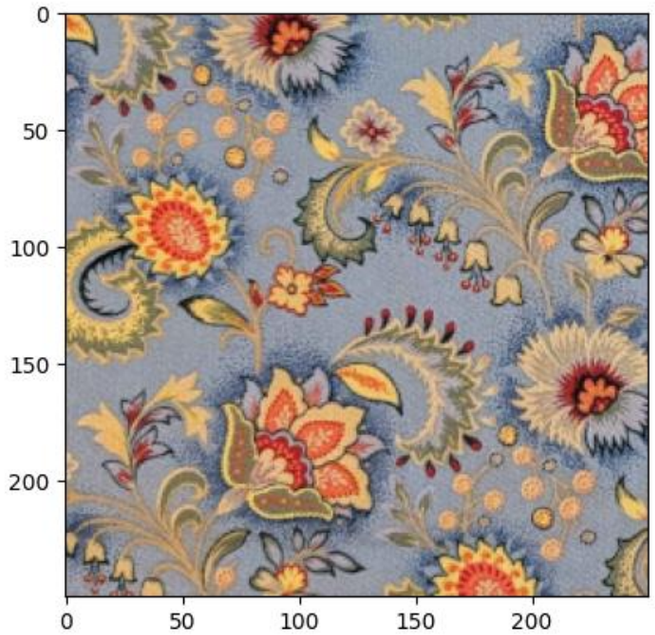
3. Seam Finding

The overlapping patches algorithm from above was edited simply to call a customized cut function prior to adding the patch to the image. Parameters were kept the same (patch_size=25, overlap=11, tolerance=5). This customized function computes the error surface as described in Efros & Freeman (2001), then uses the provided utils.cut function to obtain the cut mask, which is composited for both directions, applied to the sample & template patches, and returned. Results for each of these steps are shown below:

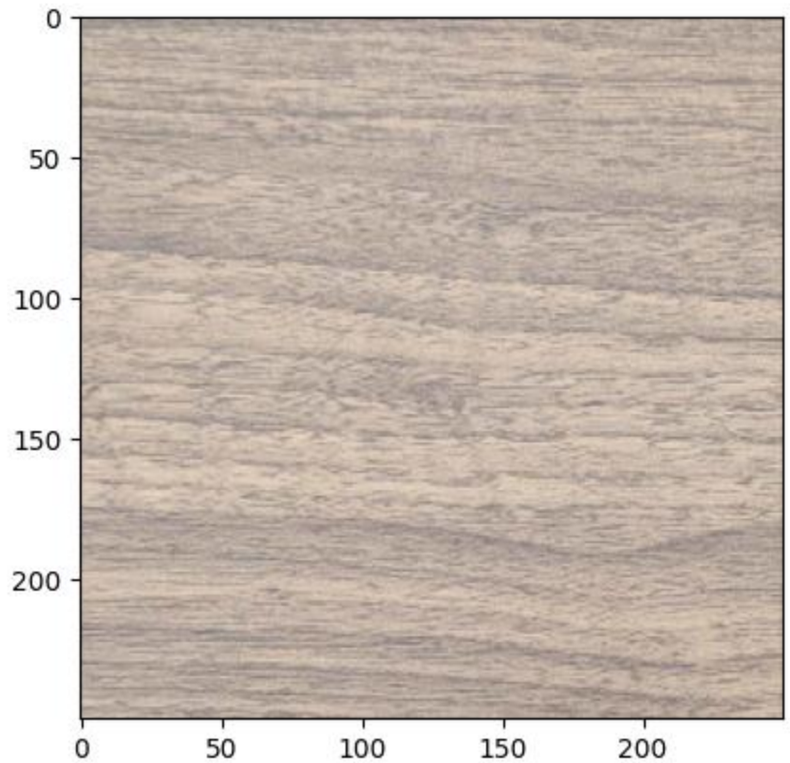


4. Additional Quilting Results

Input image is a tablecloth patch 250x250 (scaled down from a 1000x1000 patch to increase density of detail). Output is generated at 500x500. Parameters for both additional results were kept at patch size 25, overlap 11, and tolerance 5. Reproduction was quite convincing; though certain floral elements lost their original shape, they still appeared to belong in the pattern. The sample and patch size were not large enough to capture the repeating pattern in the textile, so relative positions of elements were not retained.

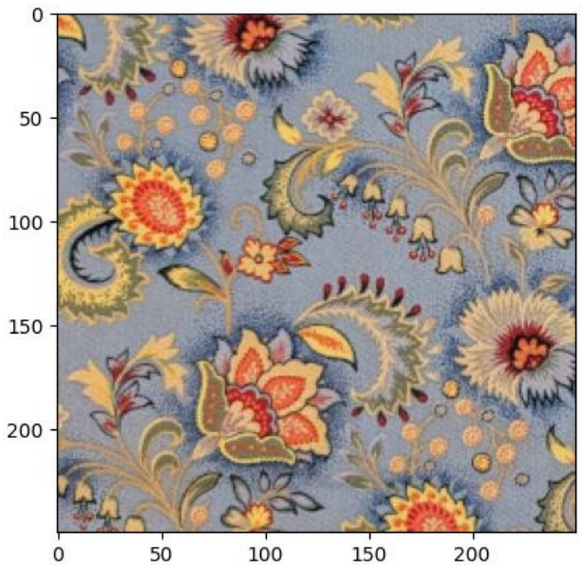


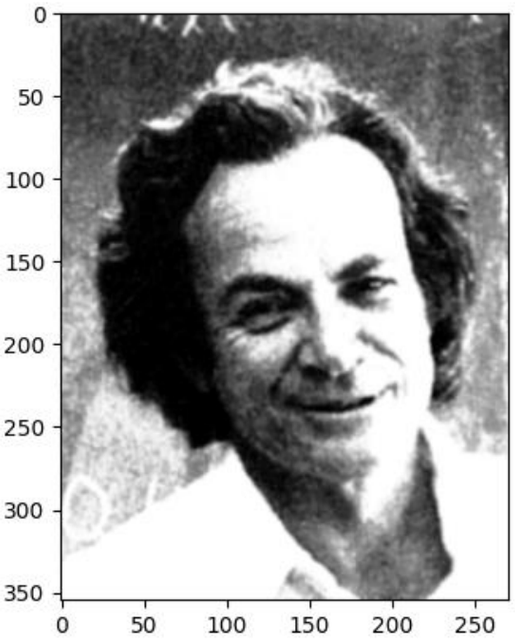
The second image is wood grain, again sampled at 1000px square and scaled down to 250px scare, then synthesized at a size of 500px square. This texture is more stochastic and provides a very convincing result with no visible seams or distortions.

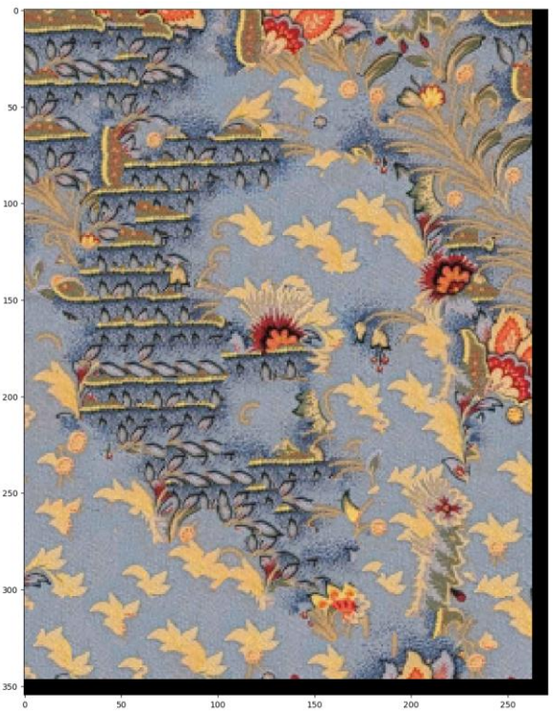

0
50
100
150
200
0
50
100
150
200


0
100
200
300
400
0
100
200
300
400

5. Texture Transfer

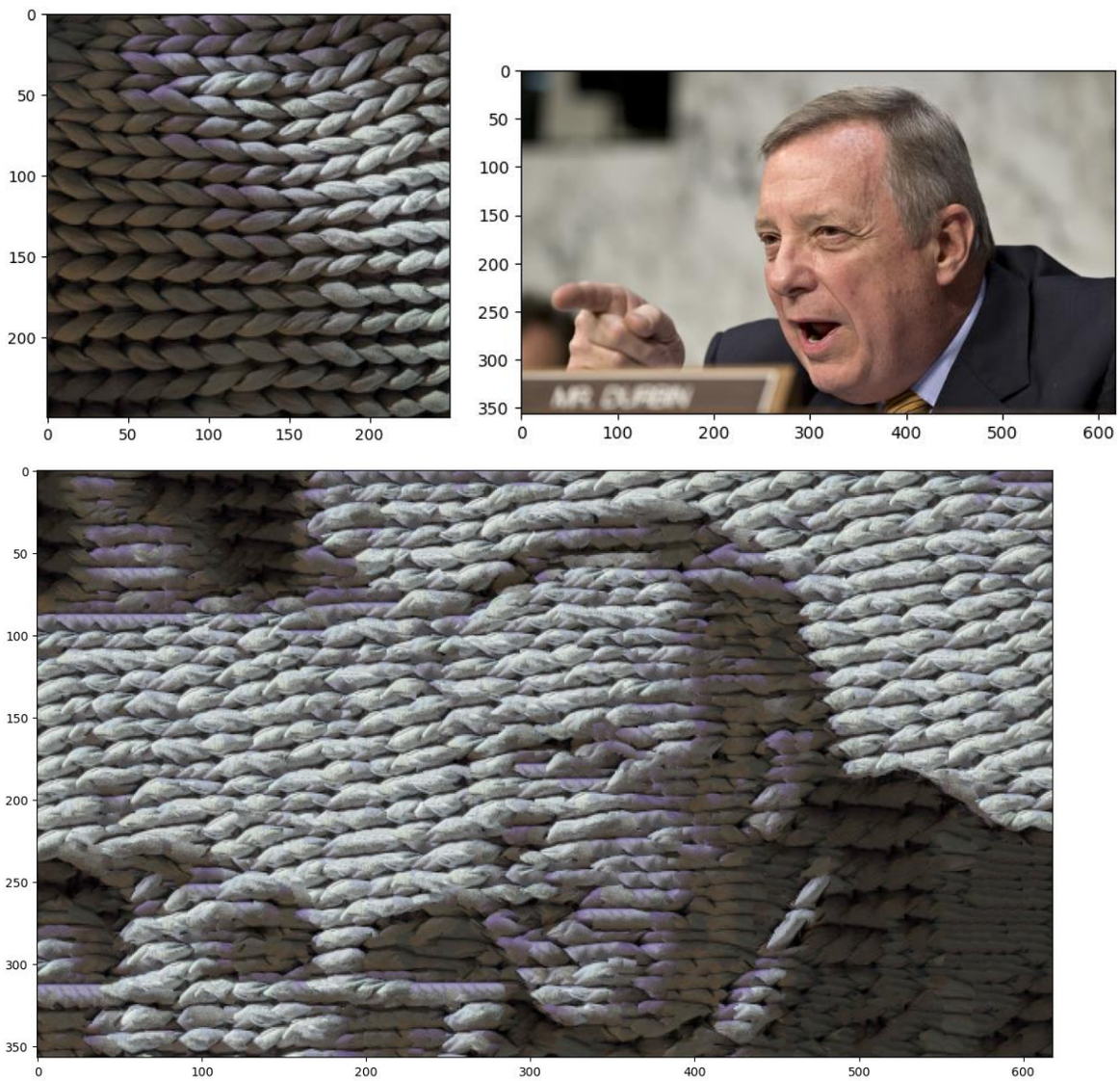






This first result maps my table cloth texture on to Feynman with a patch size of 25, overlap of 11, tolerance of 3, and alpha of 0.5. This operation reuses the SSD patch, sample choice, and customized cut methods described above. When computing the cost image, instead of just looking at the cost image for texture continuity, we compute an additional cost image from the luminance channels of the sample and the guidance image, then blend those cost images weighted by alpha before continuing to choose, cut, and stitch the sample patches.

While the general shape of the guidance image is preserved here, the higher contrast in the guidance image compared to the texture limited the fidelity of the texture transfer and led to fairly significant repeats of certain patches. The black & blue leaves are the darkest patch of the texture image, while the tan & blue leaves are the brightest. Because the Feynman image has areas with near-zero luminance and areas with near-255 luminance, these areas of the texture image are heavily sampled, and we only see decent variance in grey areas of the guidance image.



In this second result, I intentionally took a photo for the source texture with more pronounced highlights and shadows, which provides a much better range of luminance to map onto the guidance image of Illinois Senator Dick Durbin. Additionally, I used a smaller patch size of 15, with an overlap of 6 and tolerance of 3 to get a more faithful reproduction of the luminance channel (larger samples limit the amount of detail we can show).

6. Quality of results / report

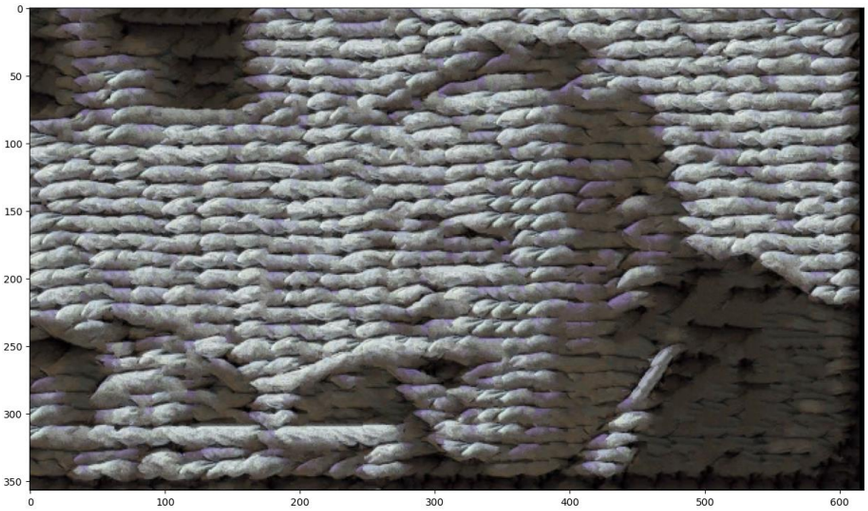
Nothing extra to include (scoring: 0=poor 5=average 10=great).

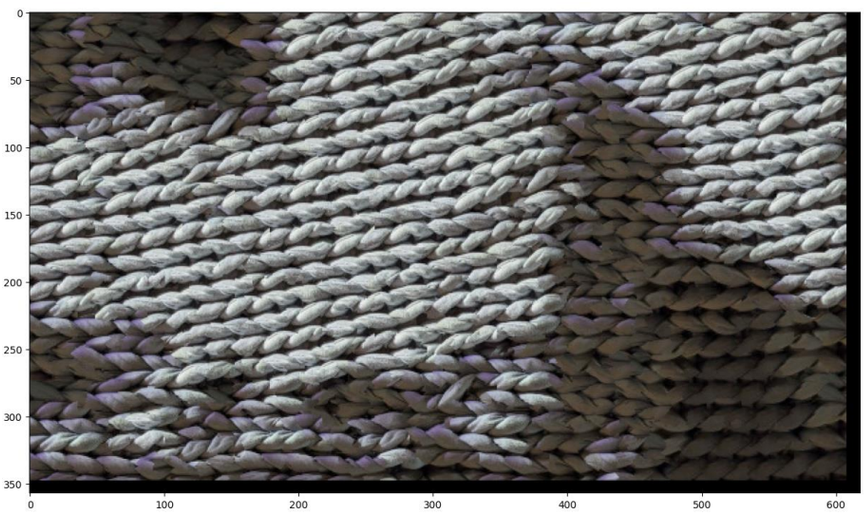
7. Iterative Texture Transfer (B&W)

I regenerated the texture transfer images from above using an iterative method as described in the SIGGRAPH paper. The iterative method starts by rendering a single image, as before, but plugs that iteration back into the algorithm, using the previous render as an additional guidance image. I combine the overlap cost image with the cost image of the previous render (additively) before blending the resulting cost image with the guidance cost image. Each iteration, the patch size is shrunk by a factor of two thirds, and the alpha value is set as described in the paper.

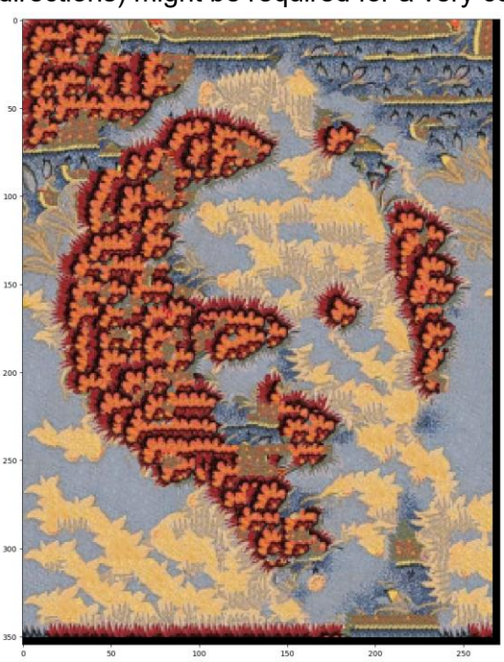
The top result, below, shows the output of the function with a starting patch size of 25 and overlap of 11. The iterative algorithm manages a slightly more faithful reproduction of the guidance image, but similarly fails to capture the original texture's macro pattern. The bottom result is generated with a starting patch size of 50 and overlap of 22 over 5 iterations. The larger

patch size does a much better job of capturing the blanket’s texture, but loses some detail in Illinois Senator Dick Durbin’s facial features and hand.





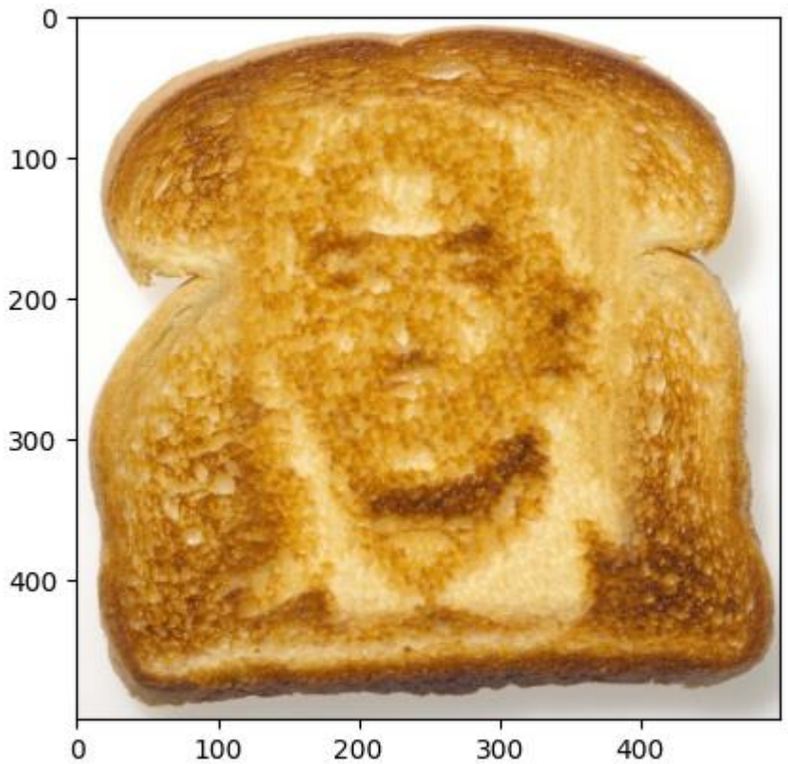
Feynman, below, is regenerated again with starting patch size 25 and overlap 11 over 3 iterations. We get a much more faithful representation of the guidance image, but again the macro detail of the texture suffers. It follows that the more stochastic the input texture, the better we can reproduce arbitrary luminance maps while maintaining that texture’s features. A larger input texture with more edge detail (i.e. varying intensity borders between light and dark in many directions) might be required for a very convincing image using the textures I provided.



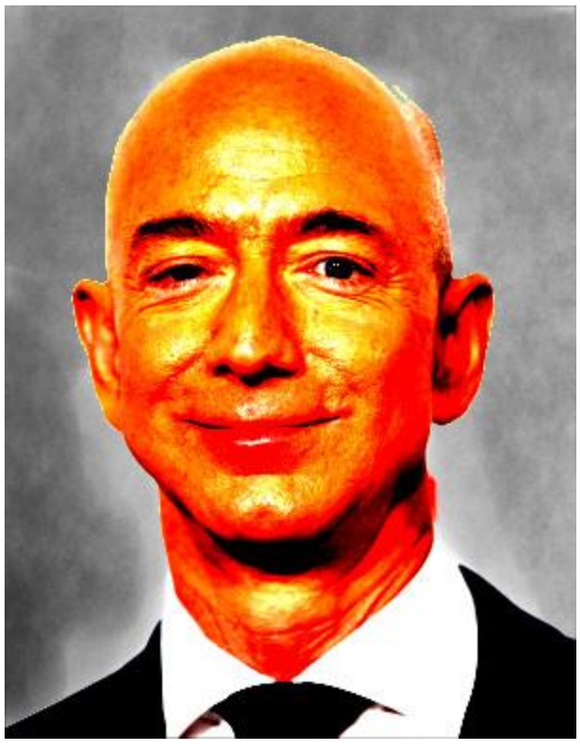
8. Face-in-Toast Image (B&W)
I implemented the face-in-toast process as an extension of the methods described above. Instead of a sample texture, the function takes a target image and two points within that image that describe a rectangle. The space inside that rectangle is consider the sample texture, while the space outside of it describes the border (the crust).

The inner image is produced by the iterative texture transfer method described in part #7. I produce this image at the full size of the guidance image and resize it to the submitted rectangle later so the texture transfer methods can be used without alteration and at the full scale of the guidance image. (Resizing early could lead to a loss of information in the guidance image.)

Next, the inner texture transfer image is resized to match the described hole (plus an overlap factor to handle edge effects). An mask is calculated where areas outside the hole are zero and areas inside the hole are one. By applying a gaussian blur to this mask, we can soften the hard edge and gradually transition between the inner image and the outer image. With this mask, I simply add the two images together, multiplying the inner image by the mask and the outer image by the identity minus the mask.



Above, Jeff Bezos is imprinted in toast. I ran into a few challenges with this implementation. The same issues with contrast were present here, and I found the best results when I took the input image and significantly modified it provide better guidanced to the algorithm. Below, left is the original image of Mr. Bezos. On the right, the guidance image used to produce the face-in-toast image above. Mr. Bezos's skin tone is very even in this photograph, so the algorithm benefited from a sharp increase in contrast. Additionally, the stark white background made the transplant very visible, so I removed it and replaced it with noise in GIMP (Filter > Render > Noise > Plasma, then de-saturate).

9. Hole filling w/ priority function (B&W)

Not completed.

Acknowledgments / Attribution

List any sources for code or images from outside sources

- AI (ChatGPT) was used for some of the image display code (matplotlib) to get more attractive results (e.g. for seam finding images).
- Help on argpartition (for ranking in texture synthesis): https://stackoverflow.com/questions/34226400/find-the-index-of-the-k-smallest-values-of-a-numpy-array
- Image of Illinois Senator Dick Durbin (the article does not specifically attribute the photo to anyone): https://nypost.com/2013/07/08/whos-a-journalist/
- Image of Jeff Bezos credited to Franziska Krug/Getty: https://people.com/jeff-bezos-kids-everything-to-know-11757697
- Image of toast credit to Jupiterimages on Getty: https://www.gettyimages.com/detail/photo/toast-royalty-free-image/86539233