Name (netid): Baoyu Li (baoyul2)
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	[ ] / 40

1. Randomly Sampled Texture

Include

- Sample and output images
- Parameters: patch size, output size

- **Patch size = 20; Output size = 200.**

2. Overlapping Patches

Include

- Output image for same sample as part 1
- Parameters: patch size, overlap size, tolerance

- **Output size = 300; Patch size = 23, overlap = 11, tolerance = 5.**

3. Seam Finding

Include

- Output image for same sample as part 1
- Illustration: for a selected patch, display (a) the two overlapping portions; (b) pixelwise SSD cost; (c) horizontal mask; (d) vertical mask; (e) combination mask. The mask is binary and tells which pixels come from which patch.
    - Note: we'll accept anything that looks like a genuine attempt to meet illustration instructions. (a) was intended to mean the two RGB patches (template and selected) that are being cut; (b) can be the SSD values of all the overlapping pixels (i.e. per-pixel SSD masked by template mask), or either one of the SSDs that you feed into cut.

- **Output size = 300; Patch size = 23, overlap = 11, tolerance = 5.**

Illustration

(a) the two overlapping portions

(b) pixelwise SSD cost

(c) horizontal mask; (d) vertical mask; (e) combination mask

4. Additional Quilting Results

Include

- At least two quilting results on your own images (excluding provided samples). Each result should show input texture image and output, and output should be more pixels than input.

sample image
quilt random
quilt simple
quilt cut

sample image
quilt random
quilt simple
quilt cut

5. Texture Transfer

Include

- Brief description of texture transfer method and parameters
- At least two texture transfer results (one result can use provided samples). Include the input texture and target images and the output (output should be same size as target image)

- **Patch size = 25, overlap = 11, tolerance = 3, alpha=0.5.**

The texture transfer is the augment of the image quilt cut algorithm by requiring that each pixel satisfy a desired correspondence map between the texture image and target image, as well as satisfying the texture synthesis requirements. In this case, we need to compute two types of SSD (SSD_overlap and SSD_transfer) and compute the cost as **alpha*SSD_overlap + (1-alpha)*SSD_transfer**, where the previous term matches the texture and the synthesized patches, while the latter term is matching the texture and target corresponding patches.

6. Quality of results / report

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

7. Iterative Texture Transfer (B&W)

Include

- Describe method
- Results on same images as shown for texture transfer.

Parameters:

- num_iter: 3, tol: 3;
- iter: 0, patch_size: 50, overlap: 22, alpha: 0.1;
- iter: 1, patch_size: 16, overlap: 7, alpha: 0.5;
- iter: 2, patch_size: 5, overlap: 2, alpha: 0.9.

The method of iterative texture transfer progresses through several iterations over the synthesized image, diminishing the block size with every iteration. I commence with a large block, subsequently reducing both the block size and overlap by one-third each time. In alignment with the paper, I also determine the alpha during the ith iteration as **alpha_i = 0.8 * (i-1)/(N-1) + 0.1**, where N represents the total number of iterations.

8. Face-in-Toast Image (B&W)

Include

- Describe method
- Show input face image, toast image, and final result

I first apply texture transfer with parameters **patch size = 25, overlap = 11, tolerance = 3, alpha=0.5** to transfer the texture of toast to my face. Then I crop the synthesized image to a 300x300 square and create the mask between foreground and background. Next, I apply Gaussian blur to the mask and blend them (output = fg * blurred_mask + bg * (1-blurred_mask)) to generate the final result.

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

Include

- Describe method
- Show result on at least two images (show input with hole and output)

Acknowledgments / Attribution

List any sources for code or images from outside sources

- Additional texture images: https://freestocktextures.com/
- Additional reference: https://jyeom3.web.illinois.edu/cs445/proj2/
- Contour detection: https://learnopencv.com/contour-detection-using-opencv-python-c/
- Feathering: https://stackoverflow.com/questions/55066764/how-to-blur-feather-the-edges-of-an-object-in-an-image-using-opencv