Name (netid): Zhilan Wang (zhilanw2)
CS 445 - Project 2: Image Quilting

Complete the claimed points and sections below.

Total Points Claimed **[ ] / 175**

Core

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

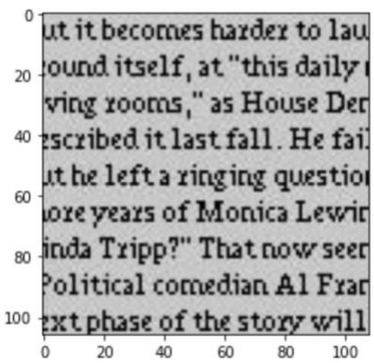
B&W

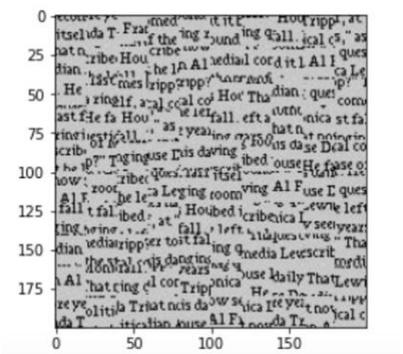
7.	Iterative Texture Transfer	[ ] / 15 ❌
8.	Face-in-Toast Image	[ ] / 20 ❌
9.	Hole filling w/ priority function	[ ] / 40 ❌

1. Randomly Sampled Texture

Include

- Sample and output images





- Parameters: patch size, output size
  out_size = 200, patch_size = 20

```
sample_img_fn = 'samples/text_small.jpg' # feel free to change
sample_img = cv2.cvtColor(cv2.imread(sample_img_fn), cv2.COLOR_BGR2RGB)
plt.imshow(sample_img)
plt.show()

out_size = 200  # change these parameters as needed
patch_size = 20
res = quilt_random(sample_img, out_size, patch_size)
if res is not None:
    plt.imshow(res)
```

2. Overlapping Patches

Include

- Output image for same sample as part 1

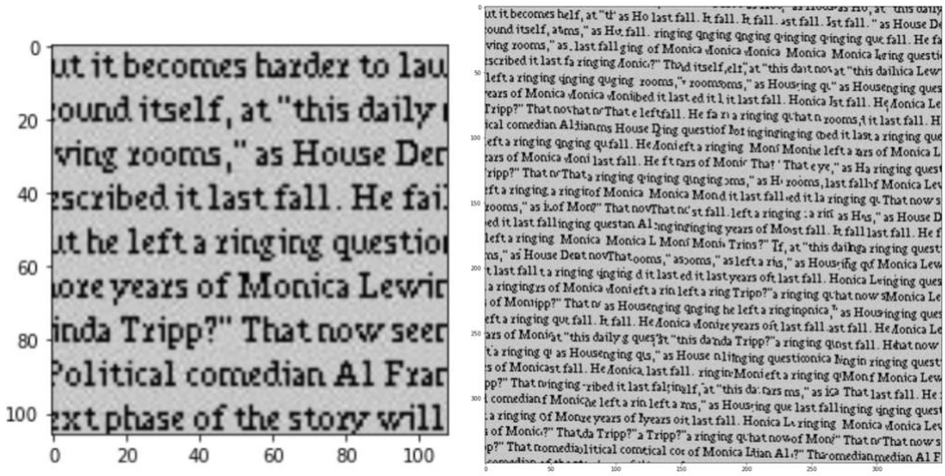
- Parameters: patch size, overlap size, tolerance
  out_size = 350 # change these parameters as needed
  patch_size = 70
  overlap = 35
  tol = 0.1

```
sample_img_fn = 'samples/text_small.jpg'
sample_img = cv2.cvtColor(cv2.imread(sample_img_fn), cv2.COLOR_BGR2RGB)
plt.imshow(sample_img)
plt.show()

out_size = 350  # change these parameters as needed
patch_size = 70
overlap = 35
tol = 0.1

res = quilt_simple(sample_img, out_size, patch_size, overlap, tol) #feel free to change parameters to get best resul
if res is not None:
    plt.figure(figsize=(10,10))
    plt.imshow(res)
```

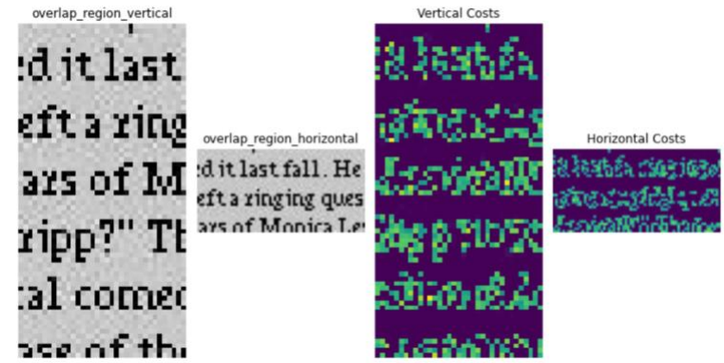
3. Seam Finding

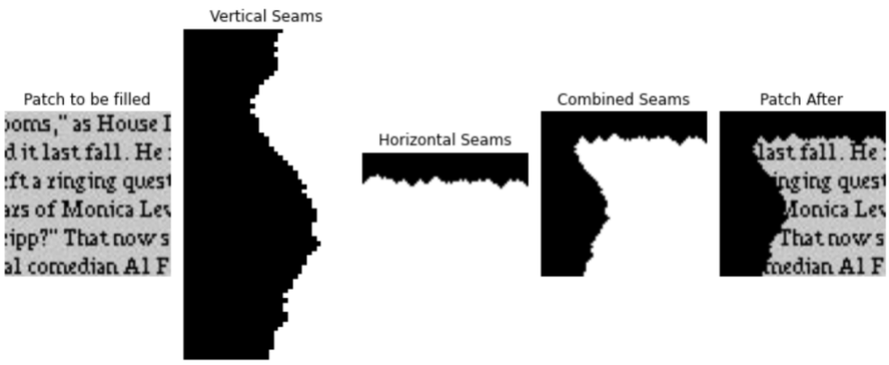
Include

- 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.



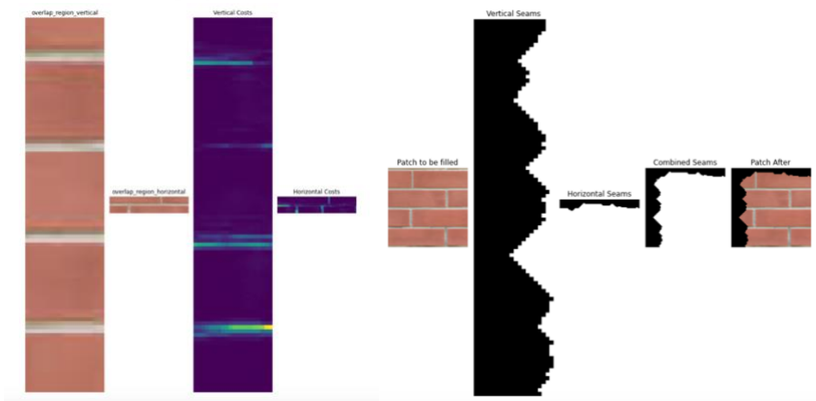


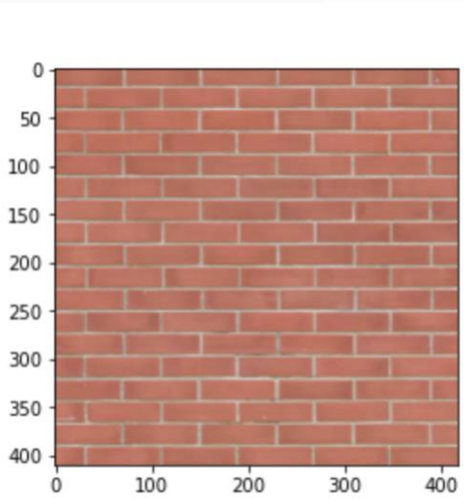
4. Additional Quilting Results

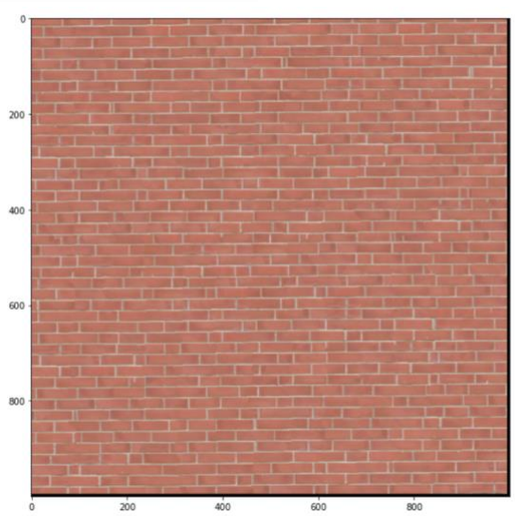
Include

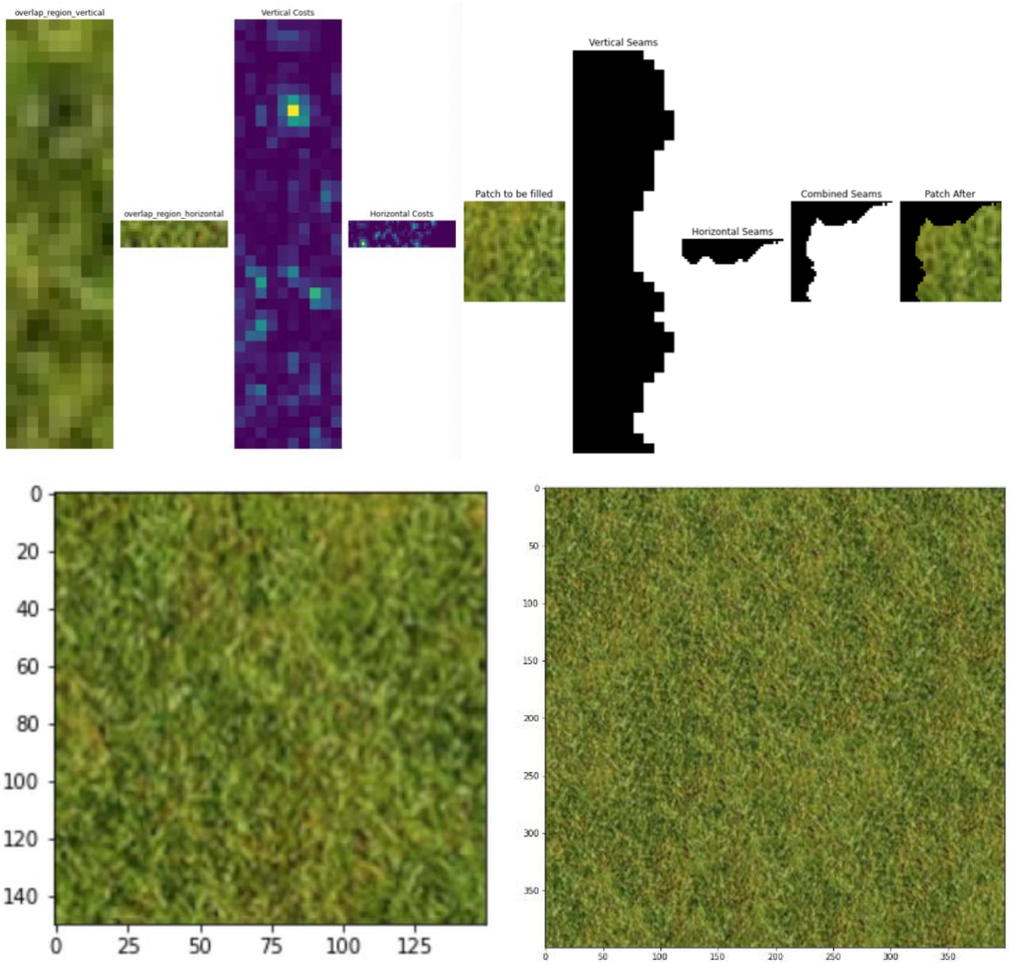
- At least two quilting results on your own images (excluding provided samples). Each result should show input texture image and output, and output should be more pixels than input.

Results using quilt_cut:







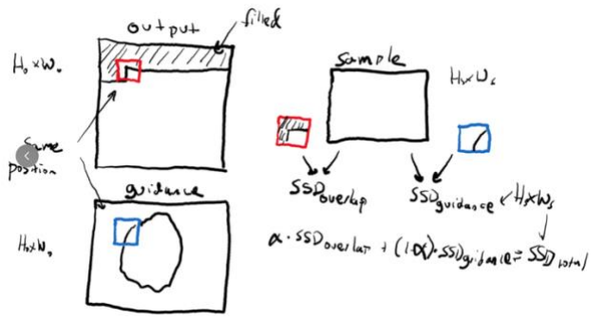


5. Texture Transfer

Include

- Brief description of texture transfer method and parameters

Compared to the quilt_cut function, texture_transfer includes an additional guidance image as input parameters. And we aim to composite a texture image onto a guidance/target image, creating a synthesized output that blends the texture's visual details with the structure of the guidance image. So in texture_transfer function, I computed two separate cost map, texture_cost_map and guidance_cost_map. THe texture_cost_map is computed by comparing the texture/sample against the current output, and guidance_cost_map is computed by comparing that against the guidance (within the overlap region). Finally, the result was weighted by alpha to balance the texture and the general shape.



*From Course Website

```
# texture, patch, mask
texture_cost_map = ssd_patch_float(texture, template_patch, mask)
guidance_cost_map = ssd_patch_float(texture, guidance_patch, mask)

cost_map = alpha*texture_cost_map+(1-alpha)*guidance_cost_map
```

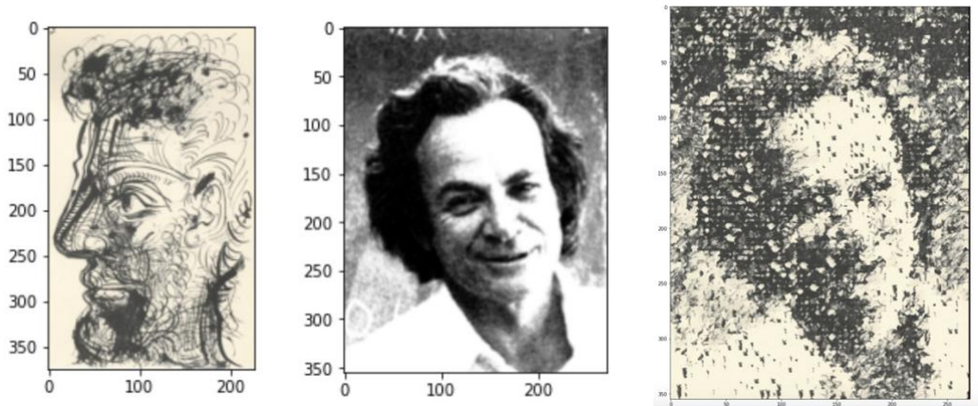
The rest of the steps are similar to quilt_cut. Specifically, I used cut function from utils.py to generate a mask that could blends the edges of the patches smoothly and minimizes visible seams. Finally, I applied the mask to the selected texture patch and its inverse on output image(returnit) where the new patch will be placed.

```
patch_temp = patch * np.repeat(cut_mask[:, :, np.newaxis], 3, axis=2)
curr_temp = template_patch * (1 - np.repeat(cut_mask[:, :, np.newaxis], 3, axis=2))
returnit[top:bottom, left:right] = curr_temp + patch_temp
```

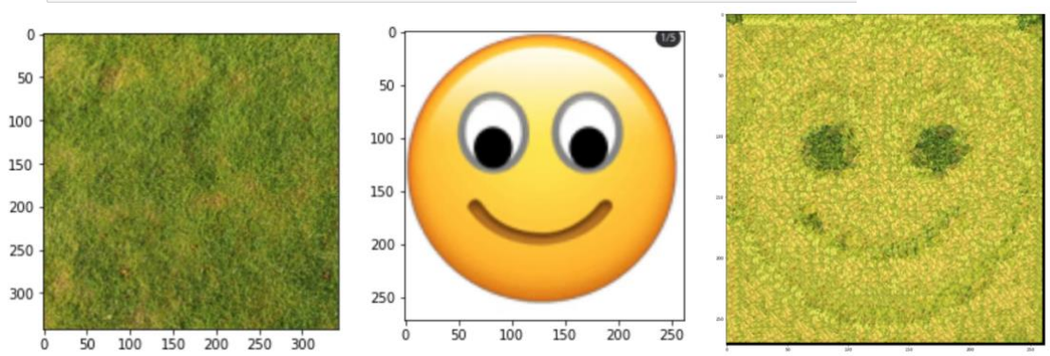
**Note: For all functions, the helper functions ssd_patch and choose_sample have been adjusted to ssd_patch_float and choose_sample_float, respectively, to accept float tolerance and mask(Inspired by post from Campuswire). This adjustment allows for finer control over the blending process and enhances the output quality. I also normalized it at the end of the function to make sure the values are within [0,1]*

- 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 = 10
overlap = 5
tol = 0.001
alpha = 0.0005

res = texture_transfer(texture_img, patch_size, overlap, tol, guidance_img, alpha)
```



```
patch_size = 15
overlap = 10
tol = 0.0001
alpha = 0.1

res = texture_transfer(texture_img, patch_size, overlap, tol, guidance_img, alpha)

plt.figure(figsize=(15,15))
plt.imshow(res)
plt.show()
```



6. Quality of results / report

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

7. Iterative Texture Transfer (B&W)

Include

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

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

Include

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

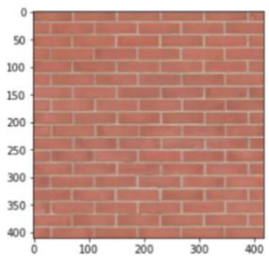
9. Hole filling w/ priority function (B&W)

Include

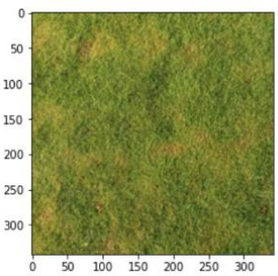
- Describe method
- Show result on at least two images (show input with hole and output)

Acknowledgments / Attribution
List any sources for code or images from outside sources

Online Images
Bricks: https://www.textures.com/category/brick/1



Grass: https://stock.adobe.com/search?k=%22grass+texture%22&asset_id=275270350.



Screen Shot:
https://yxw.cs.illinois.edu/course/CS445/Content/projects/quilting/ComputationalPhotography_ProjectQuilting.html