Name (netid): Taha Wasiq (twasiq2)
CS 445 - Project 2: Image Quilting

Complete the claimed points and sections below.

Total Points Claimed	**[120] / 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

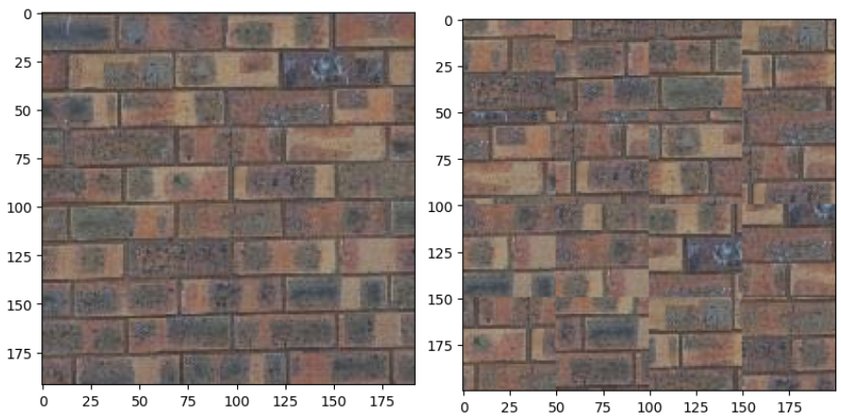
B&W

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

1. Randomly Sampled Texture

Include

- Sample and output images



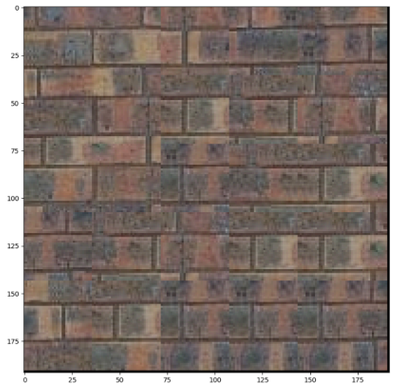
- Parameters: patch size, output size

```
out_size = 200  # change these parameters as needed
patch_size = 50
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 = 192  # change these parameters as needed
patch_size = 47
overlap =  11
tol = 2
res = quilt_simple(sample_img, out_size, patch_size, overlap, tol)
if res is not None:
    plt.figure(figsize=(10,10))
    plt.imshow(res)
```

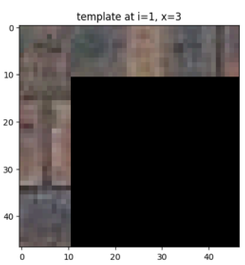
3. Seam Finding

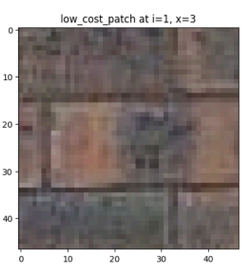
Include

- Output image for same sample as part 1

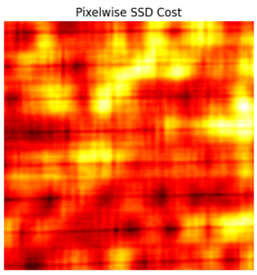
- Illustration: for a selected patch, display (a) the two overlapping portions; (b) pixelwise SSD cost; (c) horizontal mask; (d) vertical mask; (e) combination mask. The mask is binary and tells which pixels come from which patch.

  a)

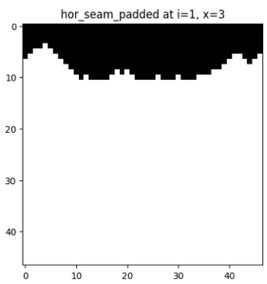




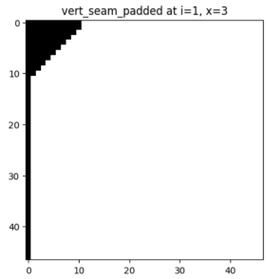
b)



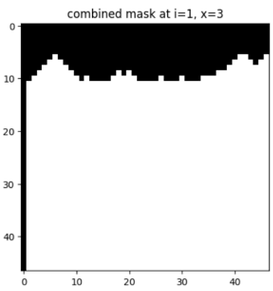
c)



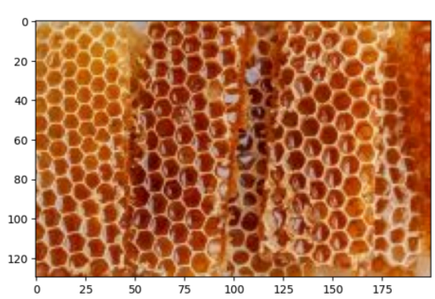
d)

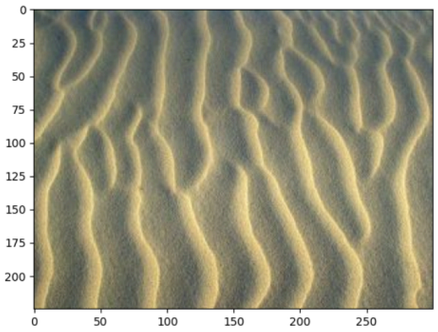


e)



4. Additional Quilting Results






5. Texture Transfer

- Brief description of texture transfer method and parameters

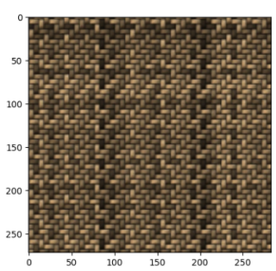
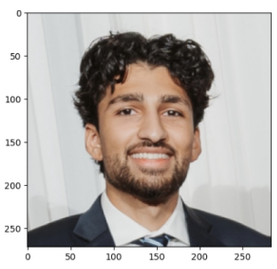
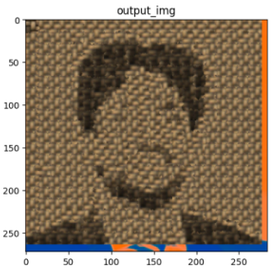
I first convert both images to Lab color space, making them the same size, and using the luminance channel as the grey scale. This way I can also keep track of the texture_img A and B channels to add the color back after calculating the low cost patch. I blurred both luminance channels, and added a random first patch. I then set the template on each iteration as the portion we are trying to match to. I set the mask and mask_inverse based on the overlapping region. I calculated the ssd for each potential patch in the texture Luminance image against the overlap, and against the portion of the guidance image. With both SSD calculations I select the best tol samples based off combined cost: alpha * ssd_overlap + (1.0 - alpha) * ssd_transfer

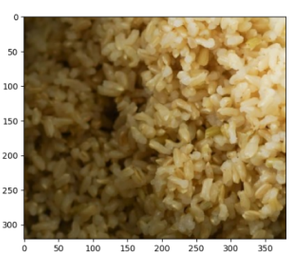
```
patch_size =17
overlap = 4
tol = 2
alpha = 0.3
```

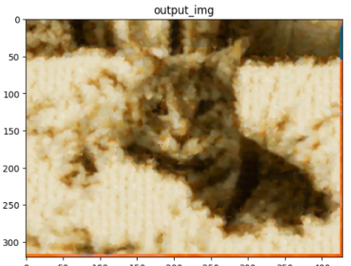
I made alpha as 0.3 so the guidance image would have more influence. I made patch size 17 to try and capture smaller details. I made overlap 4 and tol 2.

I applied the mask to the current image over the overlapping portion, which determines what is kept from the overlap. I then applied mask_inverse to our new selected portion before combining to get our new image patch. That patch is then added to the image and the process is repeated.

- At least two texture transfer results



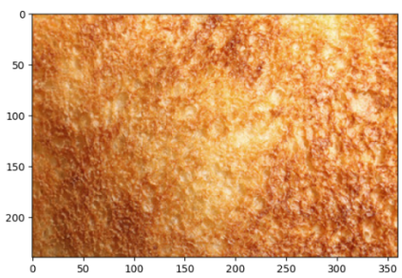


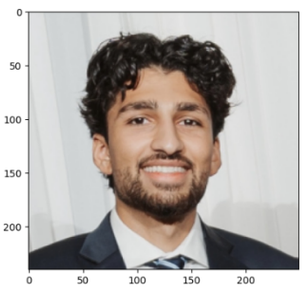
6. Quality of results / report

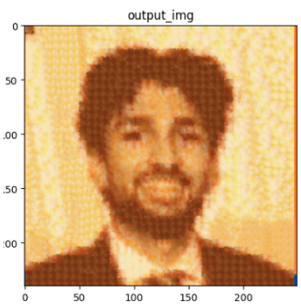
10

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

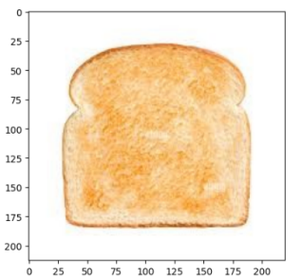
Firstly, I found an image for the texture of toast online, and I performed texture transfer the same way I had before to get an initial image.



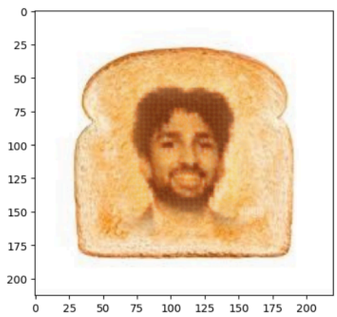




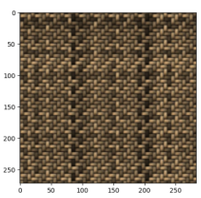
Blending computed texture onto the toast:



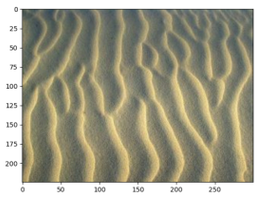
I used Alpha Compositing with Feathering, so I created a mask and applied gaussian. I then multiplied my face texture to the blended mask, and the toast picture to its inverse, then added them together.



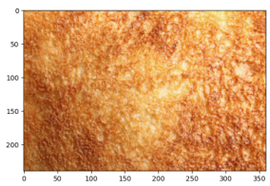
Acknowledgments / Attribution



https://depositphotos.com/photo/basket-weave-seamless-texture-tile-54054287.html



https://www.reclaimedflooringco.com/natural-patterns-how-they-affect-us-in-any-interior/



https://stock.adobe.com/search?k=bread+texture