Name (netid): Byeong Kyu Park (bkpark2)
CS 445 - Project 2: Image Quilting

Complete the claimed points and sections below.

Total Points Claimed	**[175] / 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

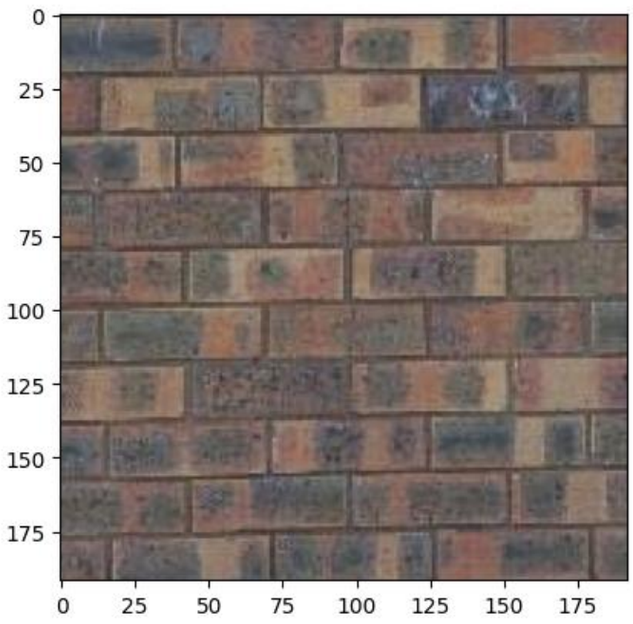
B&W

7. Iterative Texture Transfer	[15] / 15
8. Face-in-Toast Image	[20] / 20
9. Hole filling w/ priority function	[20] / 40 (works only for circular holes)

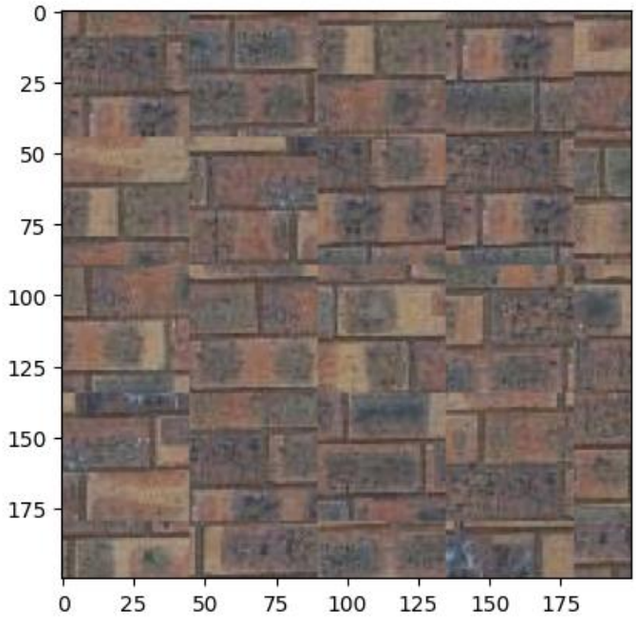
1. Randomly Sampled Texture

Include

Original



Quilt_Random



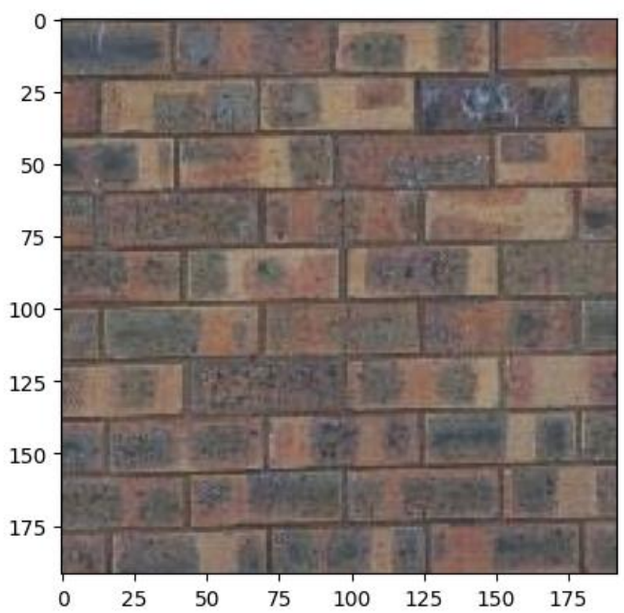
- Output Image

```
out_size = 200
patch_size = 45
```

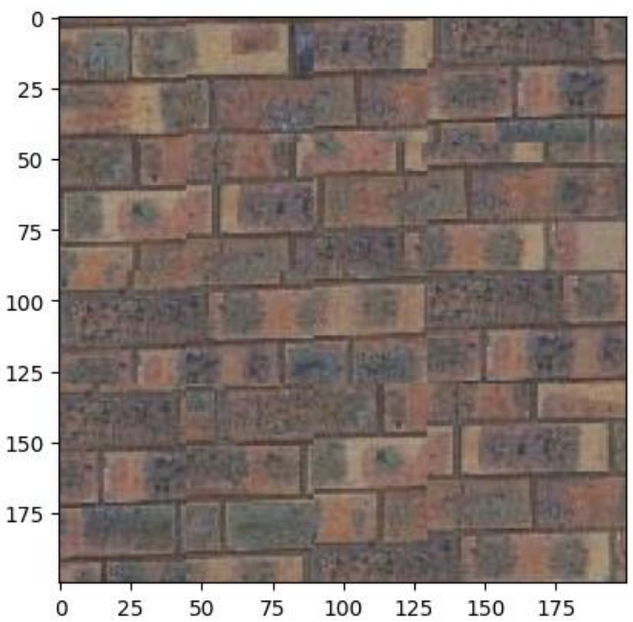
2. Overlapping Patches

```
patch_size = 70
overlap = 25
tol = 15
out_size = 200
```

Original

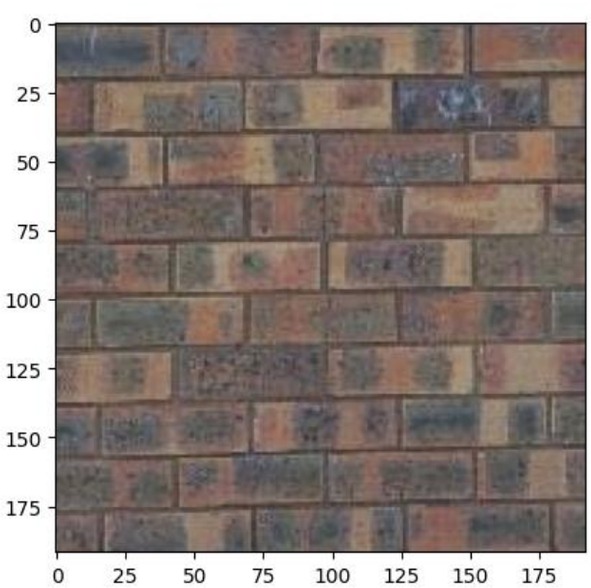


Quilt Simple

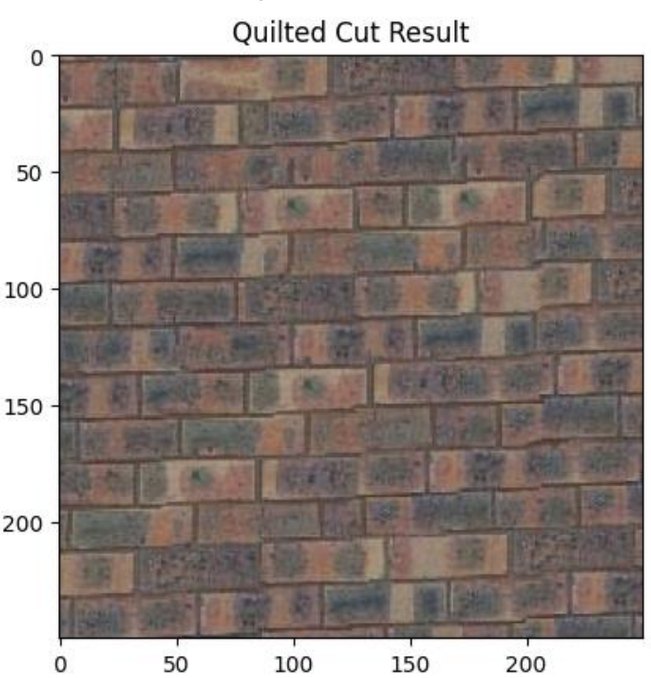


3. Seam Finding

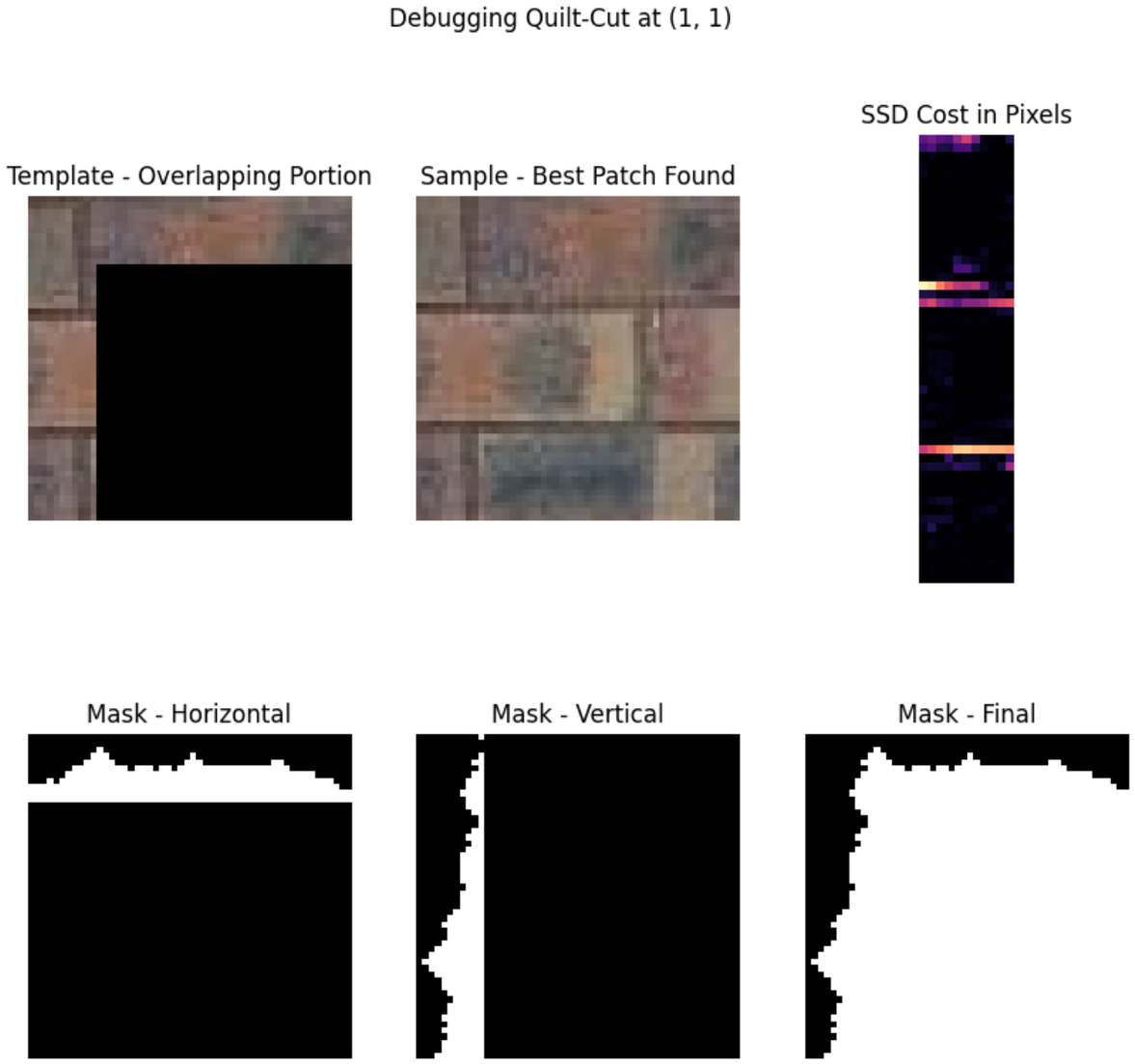
Original



Quilt Cut



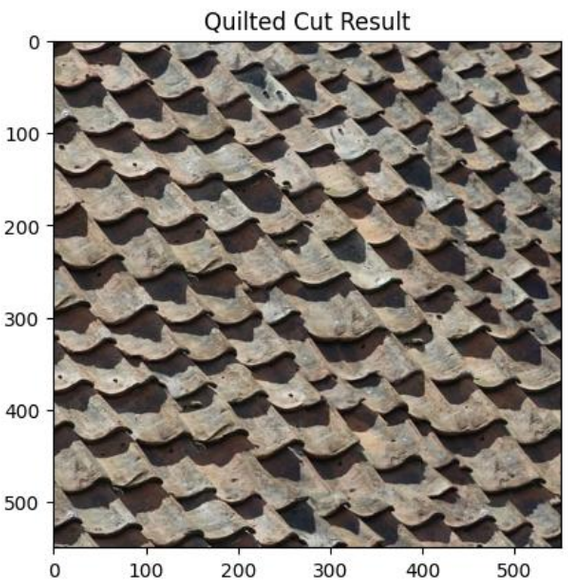
- Illustration (Debugging)



4. Additional Quilting Results

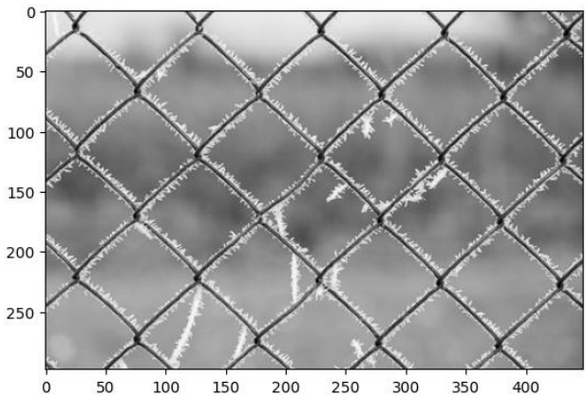
Original-1

Quilt Cut – 1

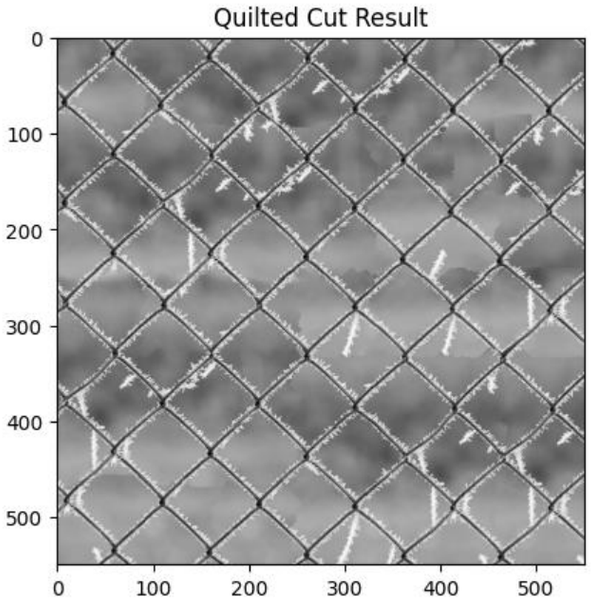



```
out_size = 550
patch_size = 55
overlap = 11
tol = 3
```

Original-2



Quilt Cut – 2



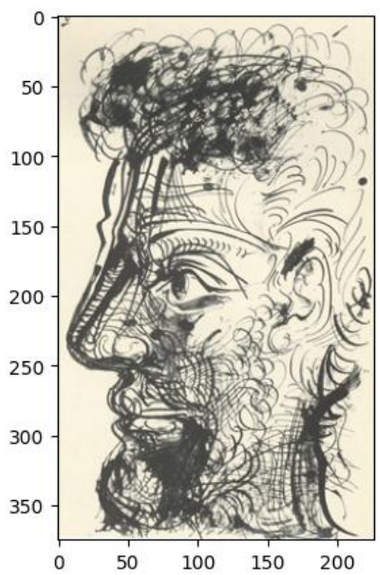

```
out_size = 550
patch_size = 95
overlap = 15
tol = 3
```

5. Texture Transfer

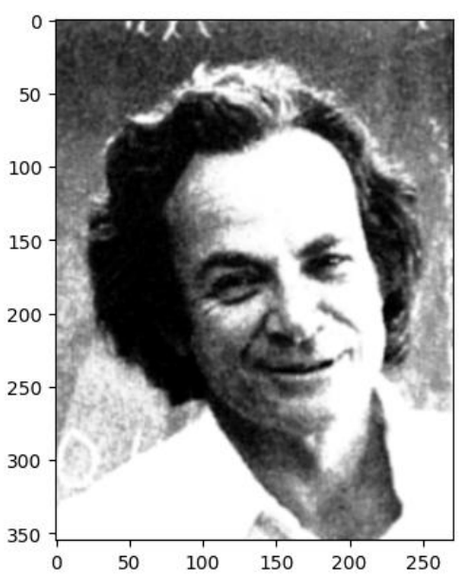
This method divides the guide image into small, overlapping patches and selects the best-matching patches from the sample image based on texture continuity and similarity to the guide, controlled by the parameter alpha (0.5). The selected patches are then blended using a scissoring technique along the cheapest pixels (best-matching pixels) to ensure smooth transitions.

To introduce some randomness while still closely following the guide, only the top 3 best matches are considered. Mixing half and half has been found to produce the best results. Since the texture contains long strokes, it was difficult to find good matching patches. To address this, the patches were kept small, and overlaps were also kept minimal for the same reason.

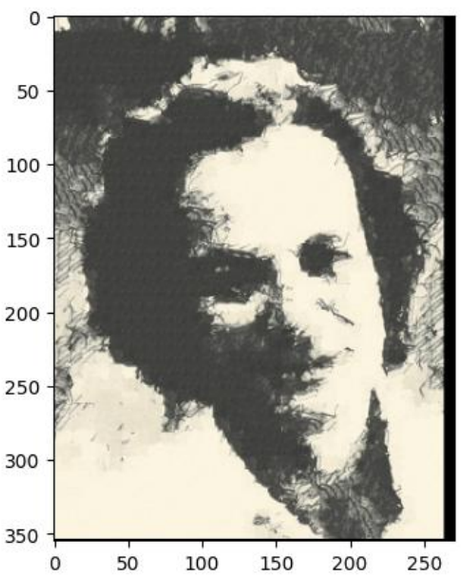
Texture Sample – 1



Guidance - 1

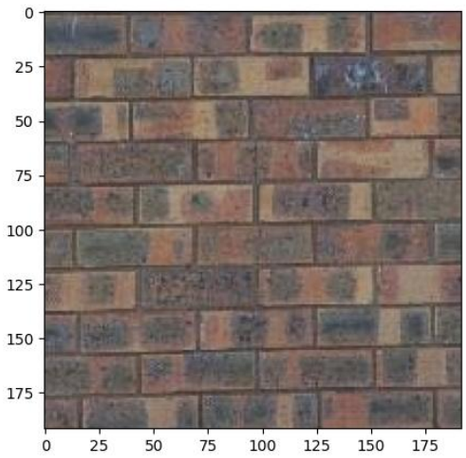


Result – 1

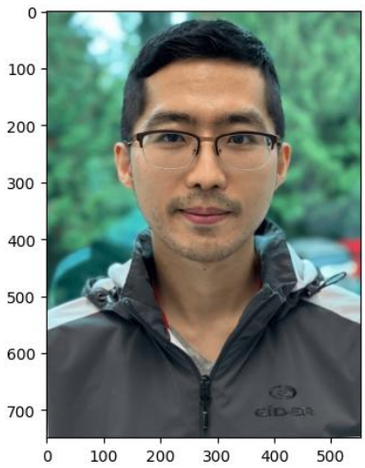


```
patch_size =12
overlap = 3
tol = 3
alpha = 0.5
```

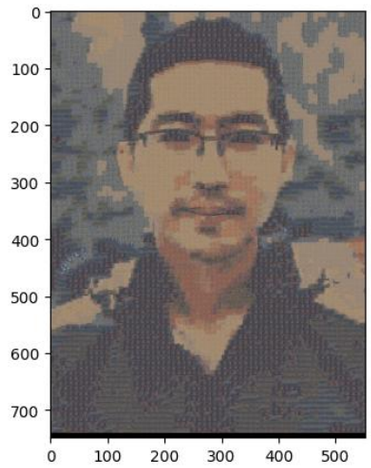
Texture Sample – 2



Guidance – 2



Output – 2



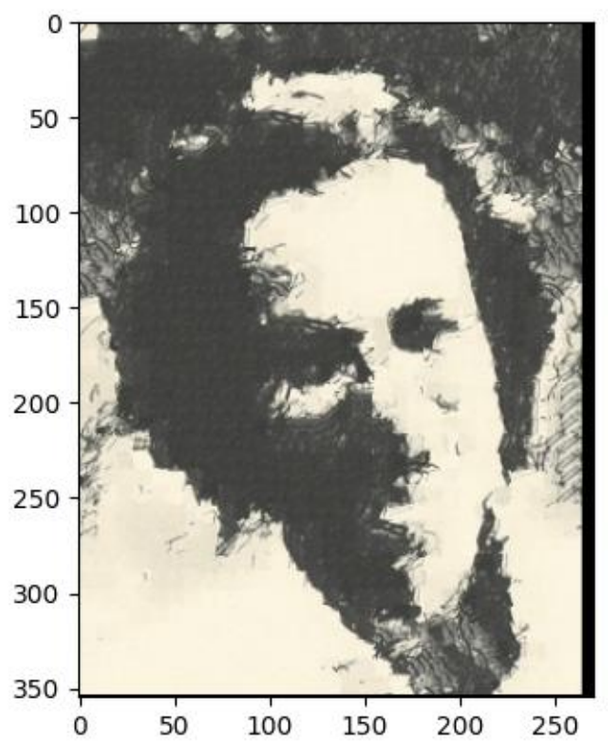
(same parameters)

6. Quality of results / report

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

7. Iterative Texture Transfer (B&W)

This method keeps refining texture transfer over multiple rounds, making patches blend better each time. It picks patches based on overlap consistency and how well they match the guide image, using a weighted SSD score. At first, it overfitted and kept reinforcing the same textures, making things look kind of repetitive. (*But after using bigger patches and a looser tolerance, it started mixing things up better, bringing back more natural variation and smoother blending.*)

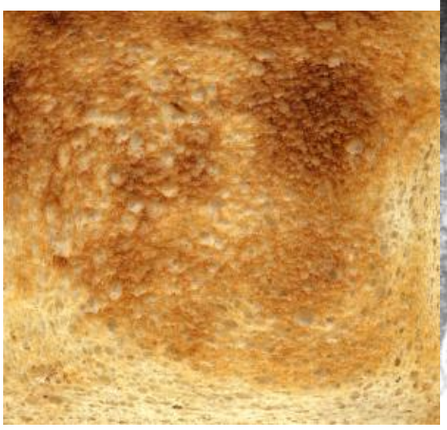


```
patch_size = 12
overlap = 3
tol = 3
alpha = 0.5
num_iters = 3
```

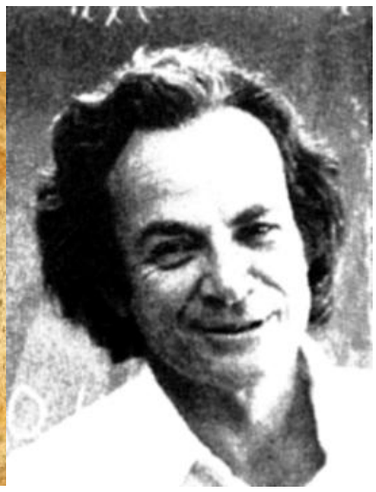
8. Face-in-Toast Image (B&W)

This method blends a face into toast by replacing its texture with small overlapping toast patches, keeping the structure aligned. A feathered mask softens the edges, preventing harsh transitions. To guide the texture transfer, the boundary of the face region is detected and prioritized, ensuring patches are selected from the most relevant areas. The selection also considers edge strength (data term), favoring regions with strong gradients to preserve structural details. Finally, blends Laplacian pyramid to ensure smooth integration by mixing details at multiple scales.

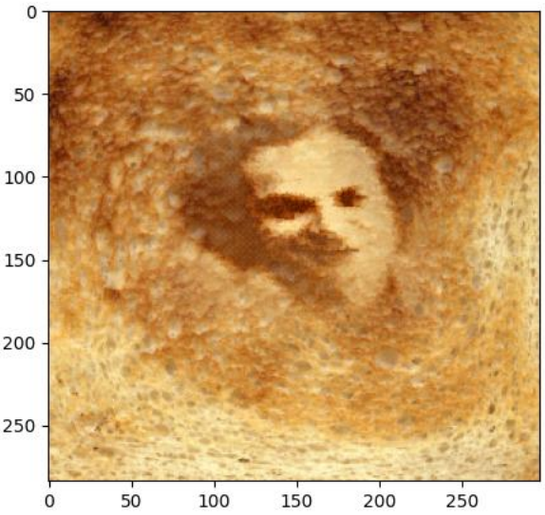
Texture – 1



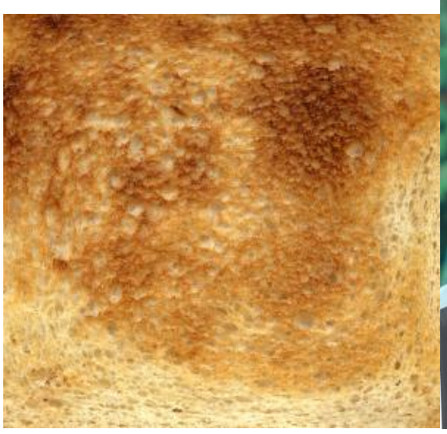
Guidance – 1



Result – 1

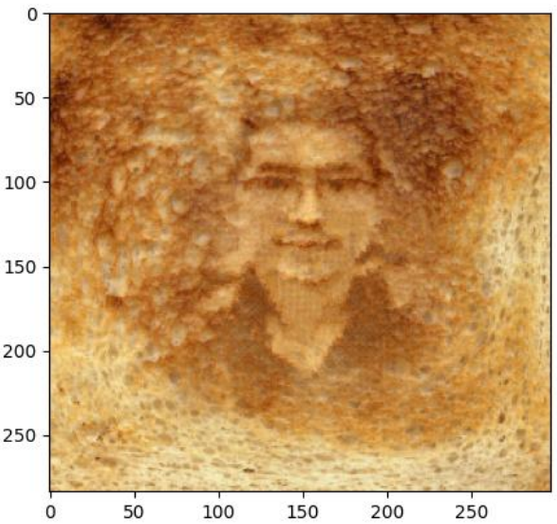


Texture – 2



Guidance – 2

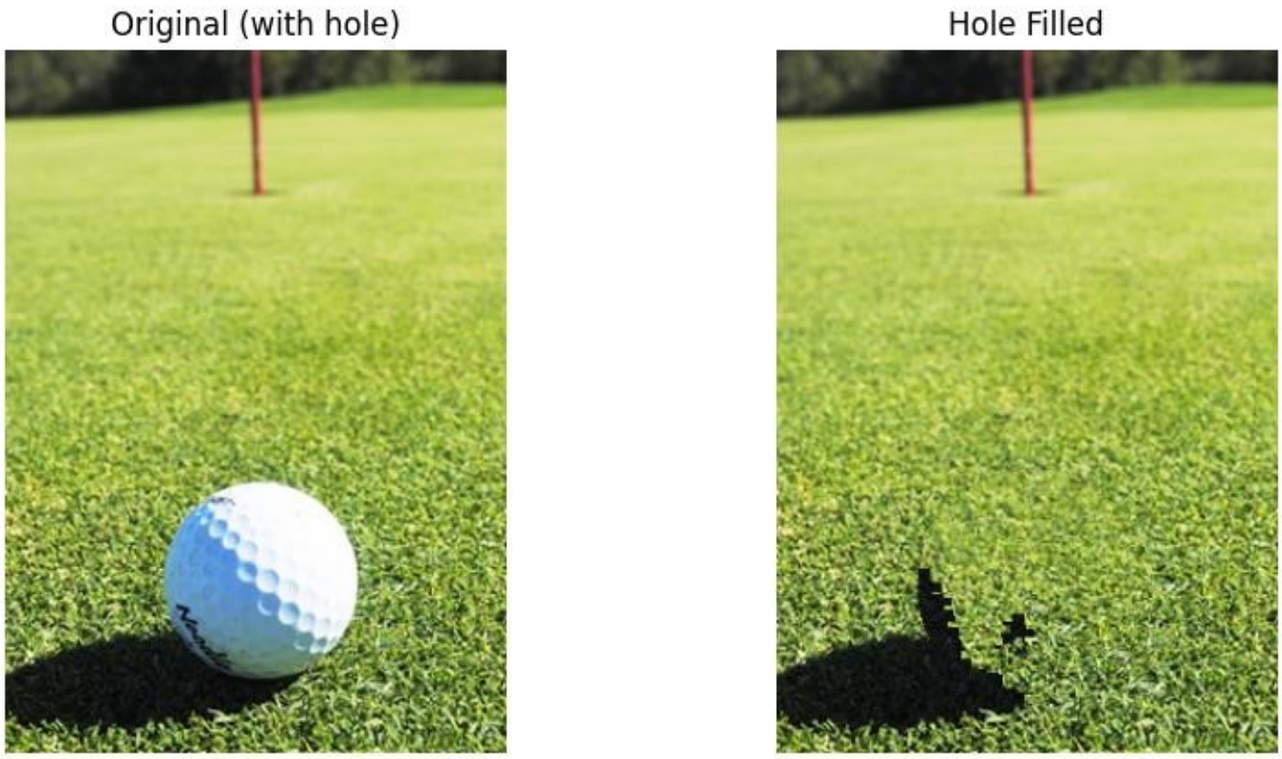
Result – 2



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

This method assumes a circular hole and fills it by expanding the texture outward. It prioritizes pixels based on solidity (known pixel ratio) and edge strength (gradient magnitude) to guide patch selection. The boundary updates iteratively, ensuring a smooth blend. However, in images like golf balls, shading variations make detecting the frontier difficult, affecting patch selection and synthesis quality.

```
patch_size = 10
cv2.circle(mask, (128, 73), 39, 0, -1) # assuming circular
```




```
cv2.circle(mask, (131, 273), 60, 0, -1)  # circular base
patch_size = 10
```

Acknowledgments / Attribution

References

- OpenCV Hole Filling
  Adapted from [Deep Learning Study](https://deep-learning-study.tistory.com/226)
- Object removal by exemplar-based inpainting
  A Criminisi, P Perez, K Toyama - 2003 IEEE Computer Society …, 2003 - ieeexplore.ieee.org
- IOS version implementation
  https://github.com/agnivsen/Exemplar?tab=readme-ov-file
- iOS Implementation of Exemplar-Based Inpainting
  [GitHub Repository](https://github.com/agnivsen/Exemplar)

Image Sources

- Hole in Wood
  [] (https://en.wikipedia.org/wiki/Hole#/media/File:Hole_in_wood.jpg)
- Ice-covered Fence
  [Pixabay](https://pixabay.com/photos/fence-bars-cold-winter-ice-grid-8500152/)
- Roof Tiles Pattern
  [Pixabay](https://pixabay.com/photos/roof-roof-tiles-pattern-house-8193787/)
- Golf Ball Image
  [Pixabay](https://pixabay.com/photos/golf-green-ball-flag-course-sport-7600947/)
- Given Sample Images

AI-Assisted Development

- GitHub Copilot & Google Colab AI
  Used for generating code templates, auto-completing boilerplate code, and refining inpainting logic.