**Assignment 2**

**bf289**

**Texture Synthesis and Image Inpainting**

* **Texture Synthesis**

Totally, the performance for texture synthesis will be better with the increment of the window size. We first use window size parameter to 5, 9, 11 pixels to do the texture synthesis, and the T5 still did not perform well, so we also use window size 15, 21 to do texture on T5, which is apparently better than previous parameters.

**T1, T2, T3, T4**

|  |  |  |
| --- | --- | --- |
| Window size = 5 | Window size = 9 | Window size = 11 |
|  |  |  |
|  |  |  |
| A picture containing outdoor  Description automatically generated |  |  |
| A picture containing text, building  Description automatically generated | A picture containing outdoor, wall  Description automatically generated | A picture containing outdoor, wall  Description automatically generated |

**T5**

|  |  |  |
| --- | --- | --- |
| Window size = 5 | Window size = 9 | Window size = 11 |
|  | A close up of text on a white background  Description automatically generated | A close up of a newspaper  Description automatically generated |
| Window size = 15 | Window size = 21 |  |
| A close up of a newspaper  Description automatically generated | A close up of a newspaper  Description automatically generated |  |

And here is the running time for different images and different window size.

A picture containing text, newspaper

Description automatically generated

* **Image Inpainting**

Make some changes on the code in question 1, make the black region as the unfilled part, and to find the best matches in the rest part. And we also use the window size parameter to 5, 9, 11 to do the image inpainting.

|  |  |
| --- | --- |
| Test Image 1 | Test image 2 |
| A picture containing photo, sky, outdoor, white  Description automatically generated | A waterfall in a forest  Description automatically generated |

**Test image 1**

|  |  |
| --- | --- |
| Window size = 5 | A black and white photo of a town  Description automatically generated |
| Window size = 9 | A crowded city street  Description automatically generated |
| Window size = 11 | A black and white photo of a city  Description automatically generated |

**Test image 2**

|  |  |
| --- | --- |
| Window size = 5 | A waterfall with trees in the background  Description automatically generated |
| Window size = 9 | A large waterfall  Description automatically generated |
| Window size = 11 | A large waterfall  Description automatically generated |

And here is the running time for different images and different window size.

A screenshot of a cell phone

Description automatically generated

* **Object Removal**

Creminis, Perez and Toyama’s algorithm has better performance, which has higher quality than the Efros and Leung’s algorithm, but both way cost so much time to remove object from the image.

A picture containing object

Description automatically generated

|  |  |
| --- | --- |
| Efros and Leung’s algorithm | Creminis, Perez and Toyama’s algorithm |
| A black and white photo of a building  Description automatically generated | A black and white photo of a building  Description automatically generated |

A close up of a logo

Description automatically generated

|  |  |
| --- | --- |
| Efros and Leung’s algorithm | Creminis, Perez and Toyama’s algorithm |
| A black and white photo of a building  Description automatically generated | A black and white photo of a building  Description automatically generated |



|  |  |
| --- | --- |
| Efros and Leung’s algorithm | Creminis, Perez and Toyama’s algorithm |
| A black and white photo of a building  Description automatically generated | A black and white photo of a brick building  Description automatically generated |

* **Image Quilting**

**Quality**

The Efros and Freeman’s method is much better than the Efros and Leung’s method when we use the same image at the same window size 5, which means the patch size is 25.

|  |  |  |
| --- | --- | --- |
| texture | Efros and Leung’s Algorithm | Efros and Freeman’s Algorithm |
|  | A picture containing building  Description automatically generated | A computer mouse and keyboard  Description automatically generated |
| A picture containing outdoor object, honeycomb  Description automatically generated |  |  |
| A picture containing indoor, photo  Description automatically generated | A picture containing outdoor  Description automatically generated |  |
| A close up of a building  Description automatically generated | A picture containing text, building  Description automatically generated | A picture containing wall  Description automatically generated |
| A screenshot of a cell phone  Description automatically generated |  | A close up of a newspaper  Description automatically generated |

**Running Time**

For the running time, the Efros and Freeman’s method is not always bettre than the Efros and Leung’s method. For different images, the result is different, for T1 to T4, the Efros and Freeman’s method is better since these texture is easy and Efros and Leung’s method works on one pixel each time, which is slower than one patch each time. And for T5, which is a hard texture, although one patch is quicker than one pixel, but the overlapping cost much time, which makes the total time longer.

Running time for Efros and Freeman’s method

A picture containing indoor

Description automatically generated

The implementation of the Efros and Freeman’s Algorithm is obtained from <https://github.com/veslam/Exemplar-Based-Inpaining-Python>.

* **How to Run**

The texture synthesis and the image inpainting are implemented in efros.py, and object removal is implemented in exemplarBasedInpainting.py, and to get the results for first three questions, just run the main.py, which has included all these parts.

The image quilting is implemented in PathBasedSynthesis.py, you can just run this file to get the result for the last question.