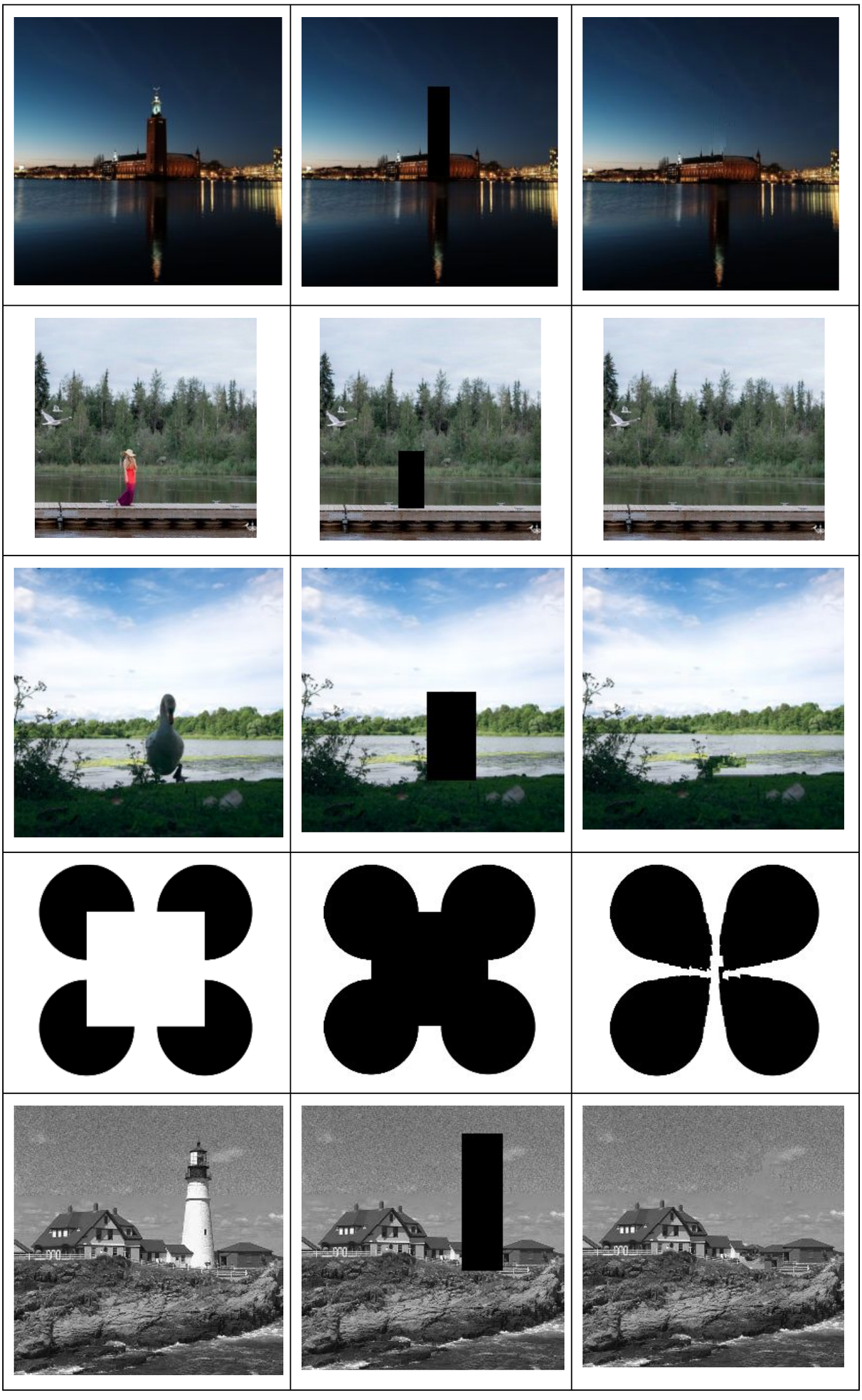
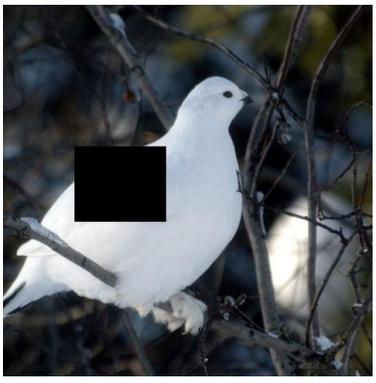
Region Filling and
Object Removal by
Exemplar-Based Image Inpainting





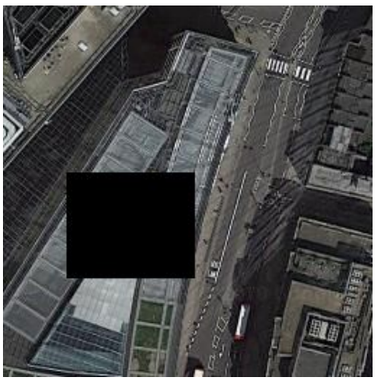









<table>
<tr><td>  </td><td>  </td><td>  </td></tr>
<tr><td>  </td><td>  </td><td> 
Psi: 10 , window: 30</td></tr>
<tr><td>  </td><td>  </td><td>  </td></tr>
</table>

Tuning parameters:

For better results, there are a few tuning parameters which we vary for different types of images:

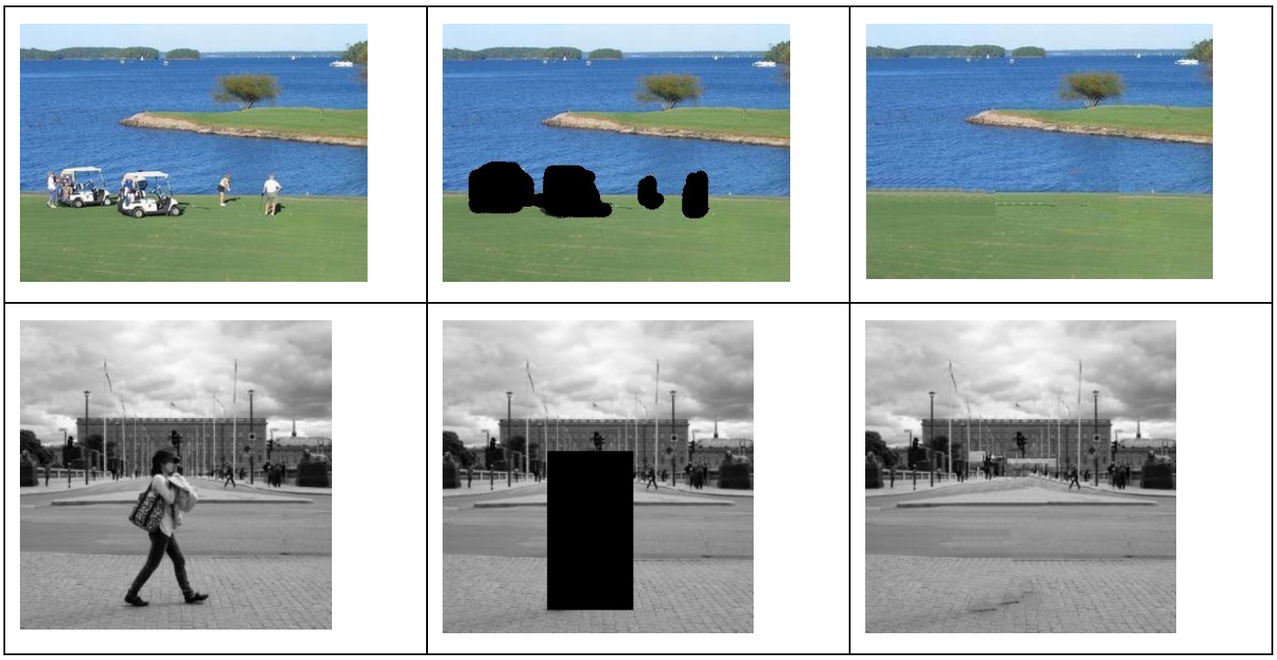
1) **Patch size**: It should be slightly larger than the largest distinguishable texture element. “psi” is half patch size in out code. Increasing psi results in lesser time.
2) **Window:** “window” parameter is the search window for patches around our source patch. Increasing this parameter means that the matching pattern is at a large distance from our patch.
3) **Alpha:** Normalisation factor which we have kept 255 as given in the paper.
4) **F:** This denotes the weightage given to the data term as compared to the confidence term.
5) **Grad_window:** It is the window around the patch in which we search for the maximum gradient for finding the isophote.

Our GUI tool: (data generation purpose)

For generating masks for images, we have developed a GUI tool in python. Pressing ‘m’ key toggles between brush and rectangle. In the brush tool, ‘+’ or ‘-’ changes brush size.

Object Removal:

1. **Removal of objects from image**



2. **Removing overlaid text from image**

psi = 4; window = 40; width=3; grad_window = 2; f = 1.5;
Time taken = 6 min (with fast normal)

3. **Speckles and scratches removal**

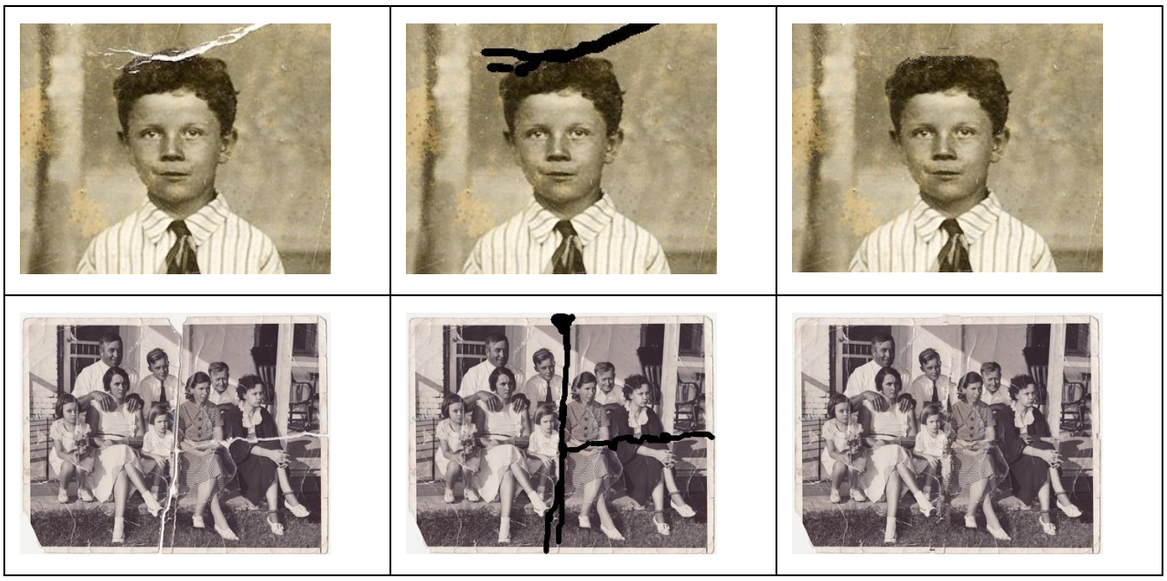
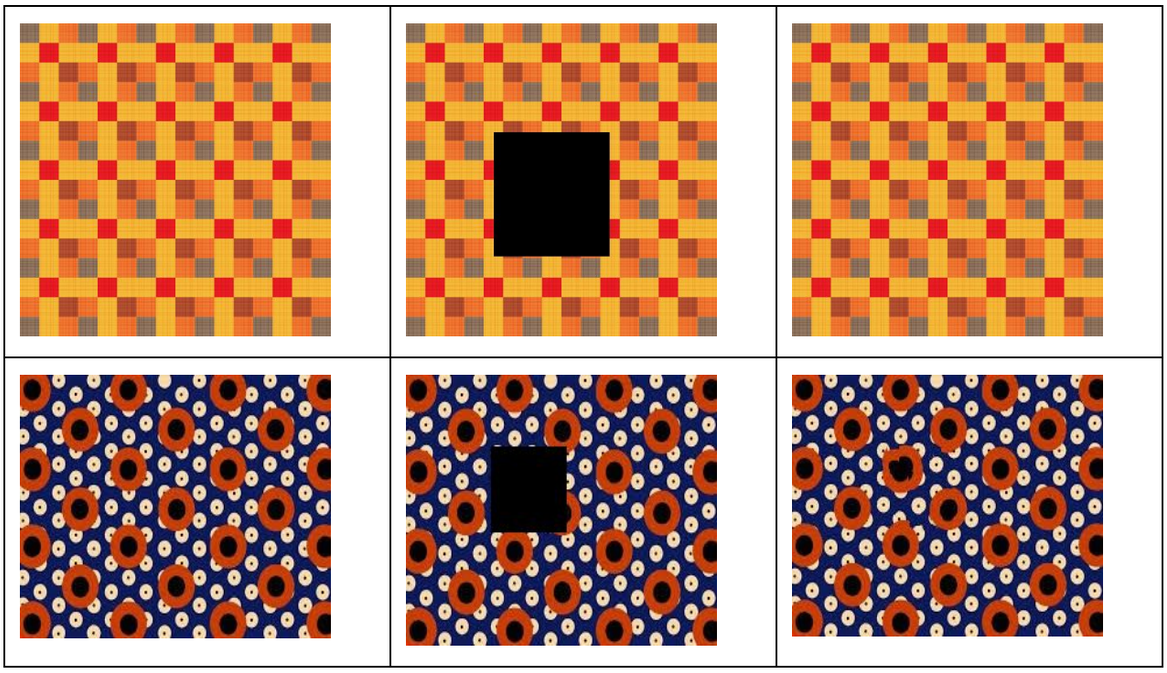
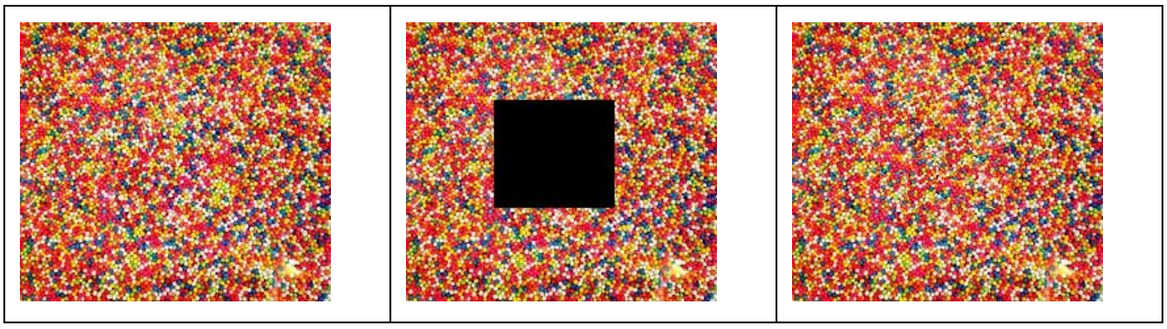


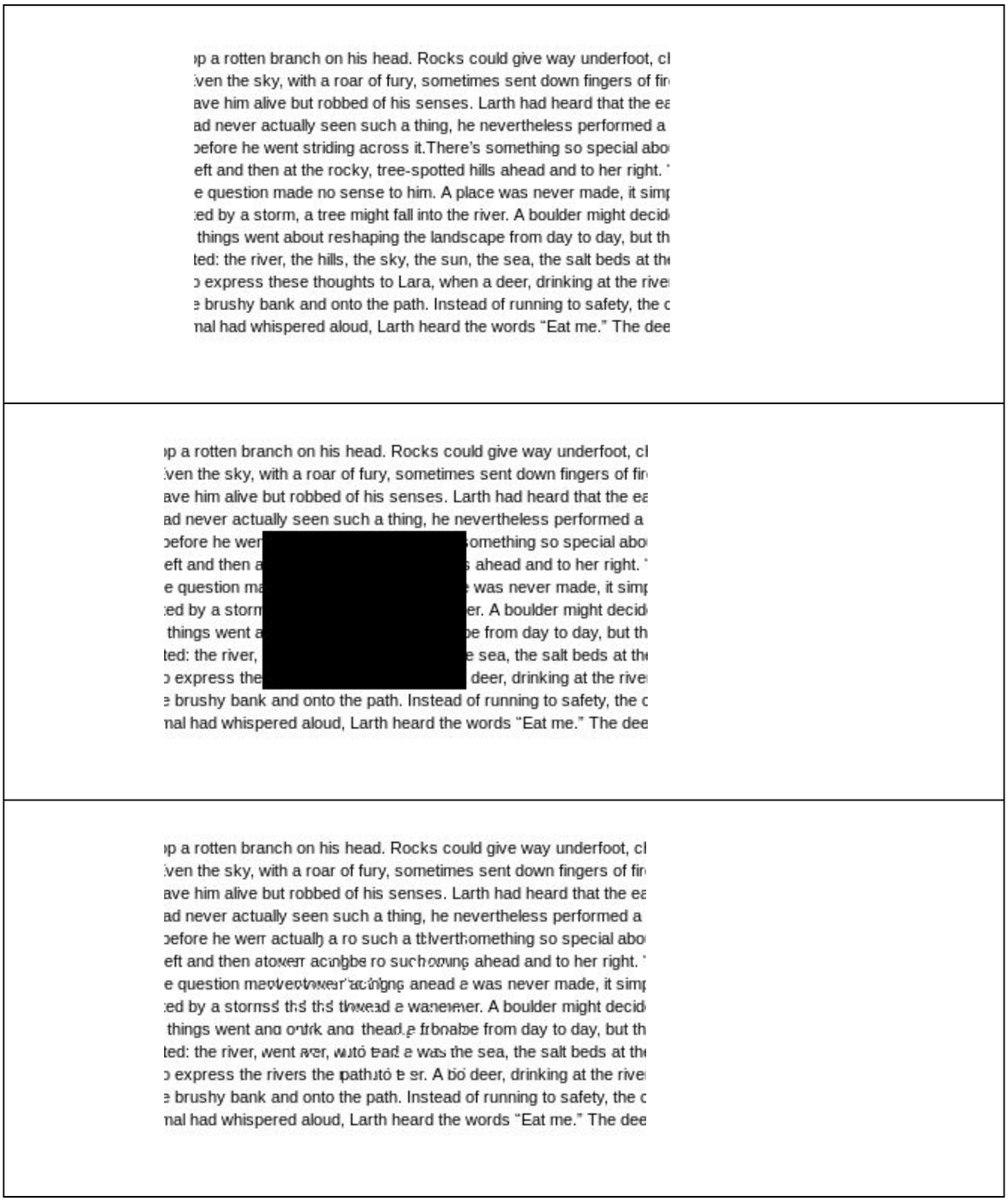
Image regeneration:

1. **Texture pattern images**





2. **Region-filling on an image of a text**



Experiments performed:

1) We found that by changing the priority to (**confidence_term + data_term**) from (confidence_term*data_term) as given in the paper, we got better visual results.

Possible reason : Whenever the data term becomes zero, the priority should depend upon the confidence term only but this is not taken into account in the original paper as priority becomes same for all such terms.

2) We observed that converting our image to **YCbCr** gave good results rather than directly using **RGB** image
*Possible reason***:** We find the most similar patch through mean squared error Between two patches. Since two very different **RGB** images can give the same MSE Values (because it is distributed in 3 channels), while **YCbCr** stores most information in it's **Y** channel.

Future improvements :

1. Normal line: We have used a rather crude method (although better than that was used in the paper) to evaluate the normals of the contour of the target region. Some fast methods can be used to get a good normal line.
2. Calculation of the isophotes is also done approximately which greatly affects the build order of image. Techniques can be used to find better isophotes.
3. Square patch create some unwanted effects in images with circular patterns. Variable patch shapes can improve upon this.

Results after changes

Psi = patch size for filling patches
Psi1 = patch size for comparisons (sum of difference squared)
For our previous implementation psi = ps1. But in our modified algorithm as suggested by professor, we are comparing our results when psi < psi1.

1.

 Input Image	 psi = 8, psi1 = 8
 psi = 4, psi1 = 8	 psi = 1, psi1 = 8

2.

Input Image

psi: 8 psi1: 8

psi: 4 psi1: 4

psi: 1 psi1: 8

3.

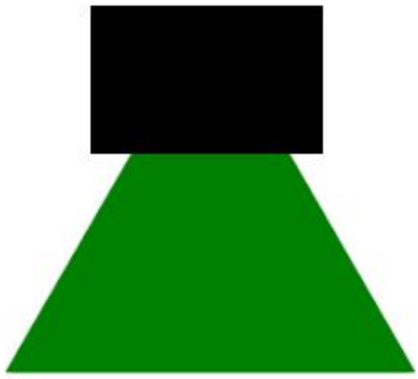
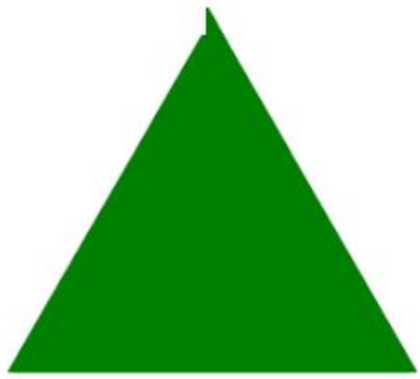
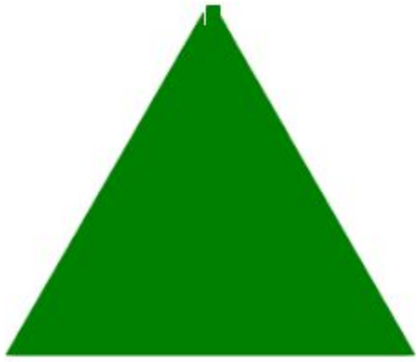
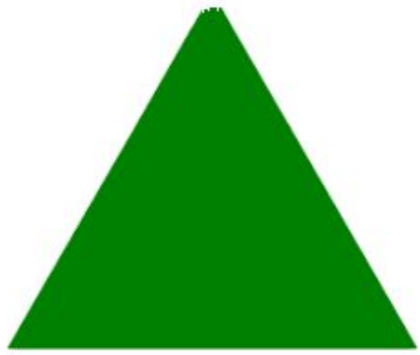
Input masked image

psi=10, psi1=10
Time taken = 50 sec

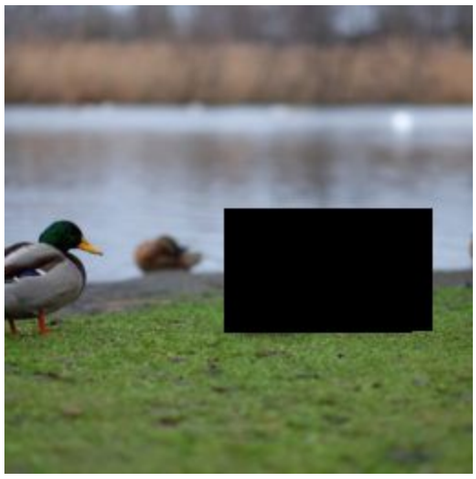
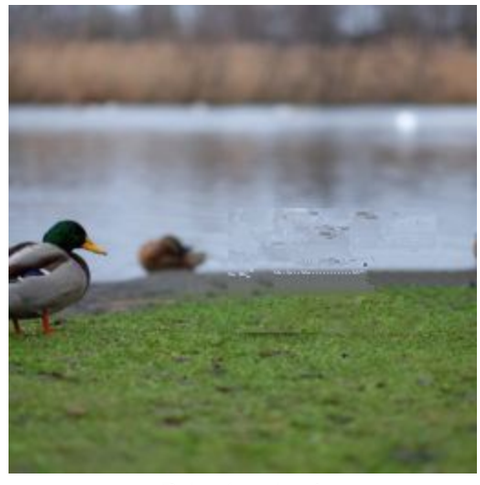
psi=5, psi1=10, window=100
Time taken = 121 sec

psi=1, psi1=10
Time taken = 1240 sec

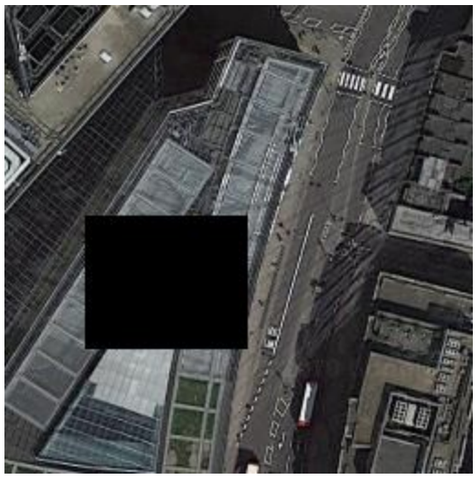
4.

 Input masked image	 psi=15, psi1=15 Time taken = 17 sec
 psi=7, psi1=15 Time taken = 40 sec	 psi=2, psi1=15 Time taken = 439 sec

5.

 Input masked image	 Psi = 5, psi1= 5 Time = 1 min
 Psi = 3, psi1 = 5 Time = 2 min	 Psi = 1, psi = 4 Time = 14 min

6.



Input masked image

Window: 50, psi: 8 psi1: 8

Window: 50, psi: 5 psi1: 8

Window: 50, psi: 1 psi1: 8

7.

Input Image

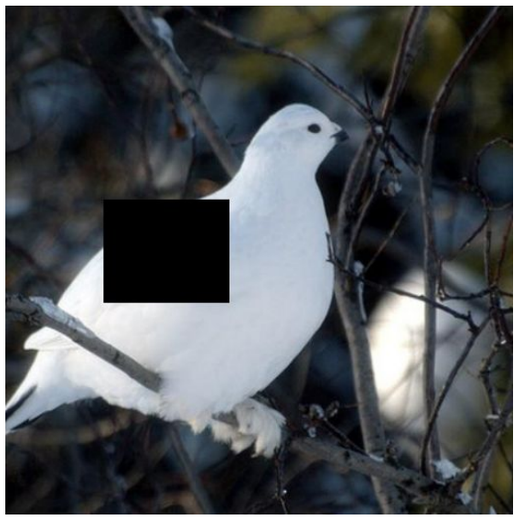
Window: 30, psi: 8, psi1: 8

Window: 30, psi: 5, psi1: 8	Window: 30, psi: 1, psi1: 8

8.

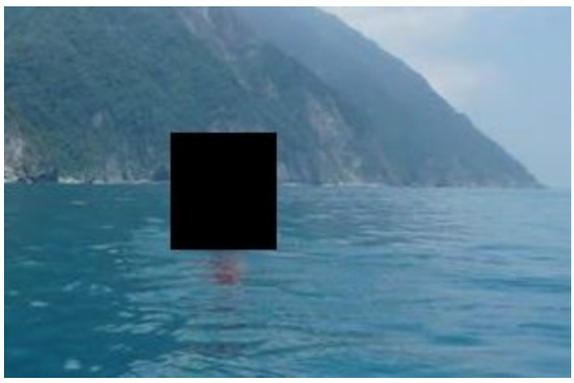
 Input Image	 Window 30, psi 8, psi1, 8
 Window 30, psi 5, psi1, 8	 Window 30, psi 1, psi1 8

9.

 Input masked image	 psi=8, psi1=8
 psi=4, psi1=8	 psi=1, psi1=8

	Time taken = 1316 sec

10.

 Input masked image	 psi=5, psi1=5
 psi=3, psi1=5	 psi=1, psi1=5 Time taken = 249 sec

Observations on changing the algorithm:

As suggested in the project-review, we modified the algorithm to distinguish between the compare window and the fill window. So we fill a small region only while comparing among big patches.
The output in the modified algorithm is more grainier in some cases as in image 8 above as compared to previous method. Sometimes it gives worse results as seen in images 4 and 5. Mostly, if we use psi = 1, the results are usually better.Better Alignment of edges among the patches only when the psi is too small relative to psi1 otherwise mostly it degrades the image regeneration.