Computer vision Lab 4
Exemplar-based methods and applications
Esther Gonzalez and Wilmer Bandres

First of all, we changed the main script a little bit to run several experiments at once. The first three lines are the important ones. In the first line we specify which images we want to run the experiment. The second line is an array of the size of patches we want to use. Finally the third line, is an array of tolerances we want. So the main script will run an experiment for each image, for each patch size and for each tolerance. All the experiments that we were able to run are in the *results* folder, they follow the format <sample_picture>-<patch_size>-<tolerance>.png where the <sample_picture> refers from which image we are going to patch.

Exercise 1.

For this exercise, we ran experiments with a patch size of 5, 7 and 9 and tolerances of 0.05, 0.1 and 0.2, for each image in the data folder. For some samples the patch size seems to be really important, for example in the blobs image, if the patch size is 9 it performs extremely well the texture generated is basically the samples given but repeated several times, this is because the original image is very symmetric and given a patch size which can cover the symmetric parts of the texture then it can reproduce the same forms pretty well. Of course a small patch does not let the same thing since it does not catch all the symmetric part of the blobs, then the resulting texture tends to deform. See **Figure 1.a, 1.b and 1.c.** for the results in the blobs sample for several patch sizes.

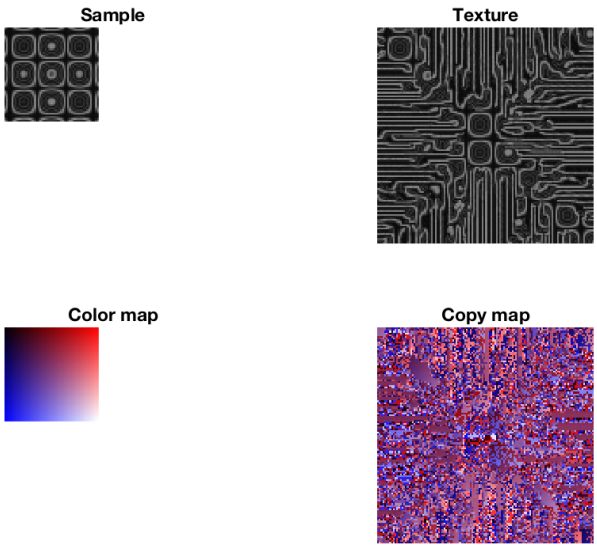


Figure 1.a Blobs picture with patch size of 5 and a tolerance of 0.3 **(a failure case)**

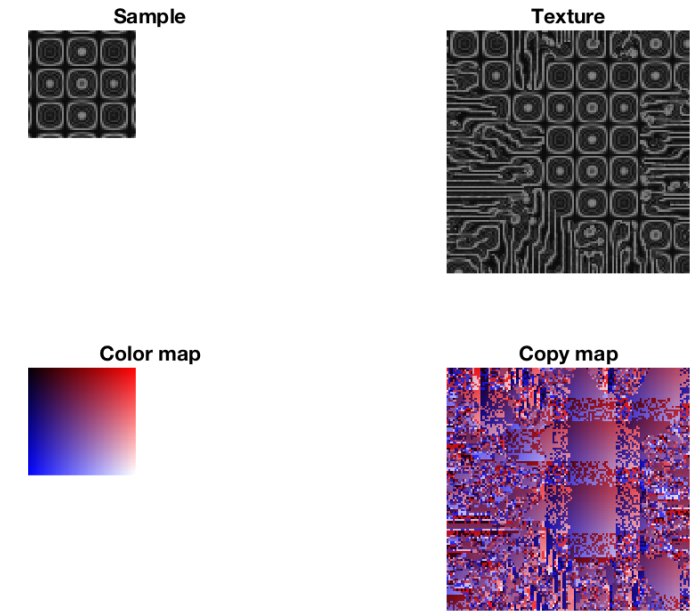


Figure 1.b Blobs picture with patch size of 7 and a tolerance of 0.3 **(a failure case)**

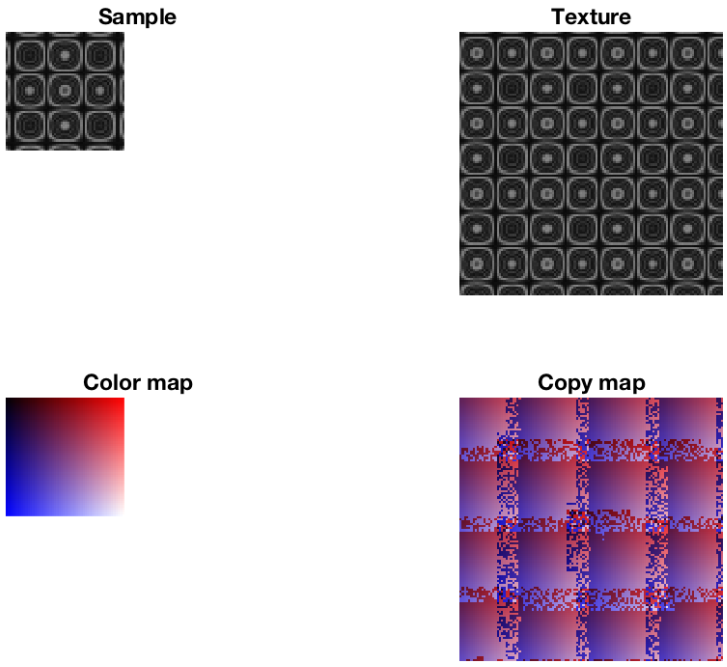


Figure 1.c Blobs picture with patch size of 9 and a tolerance of 0.3 **(success case)**

On the other hand, trying to change the tolerance parameter just gives other effect. The number of patches that could fill a single pixel increases and so the

texture tend to deform more. This parameter is known for influencing the *innovation* of the algorithm, it means that as long as this parameter increases the texture being filled has areas which cannot be found in the exemplar, it does more combinations when selecting each patch since it has more options. When this parameter decreases, the number of patches that match decreases so the texture is more similar to the exemplar in more areas of the exemplar. See **Figure 2.a, 2.b and 2.c** to see how the generated texture differs more and more with the exemplar and the color map shows how it takes patches from different places, basically from everywhere.

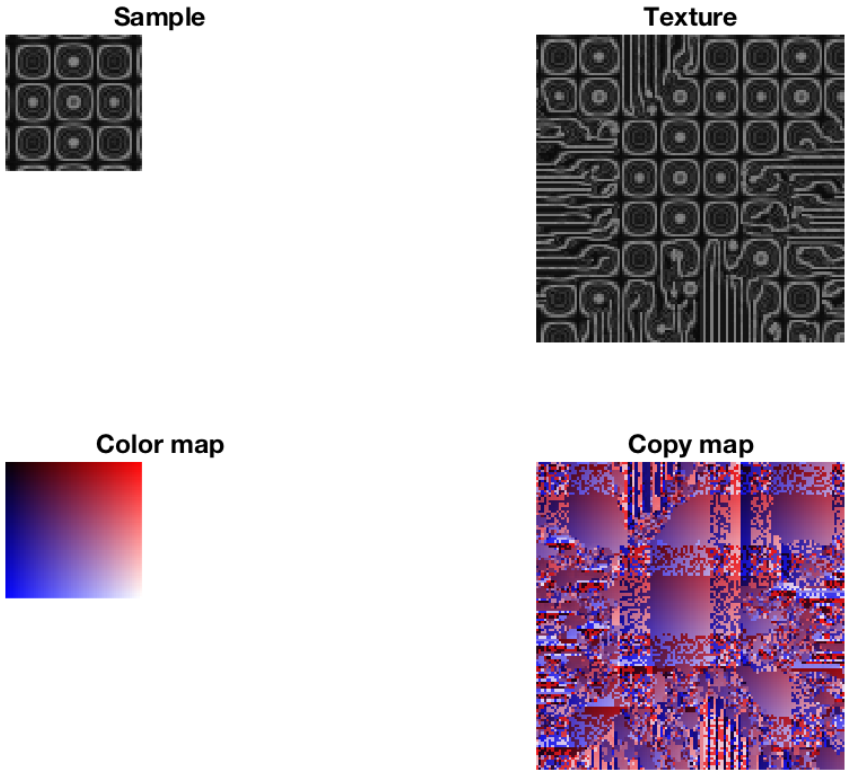


Figure 2.a Blobs picture with patch size of 7 and a tolerance of 0.05

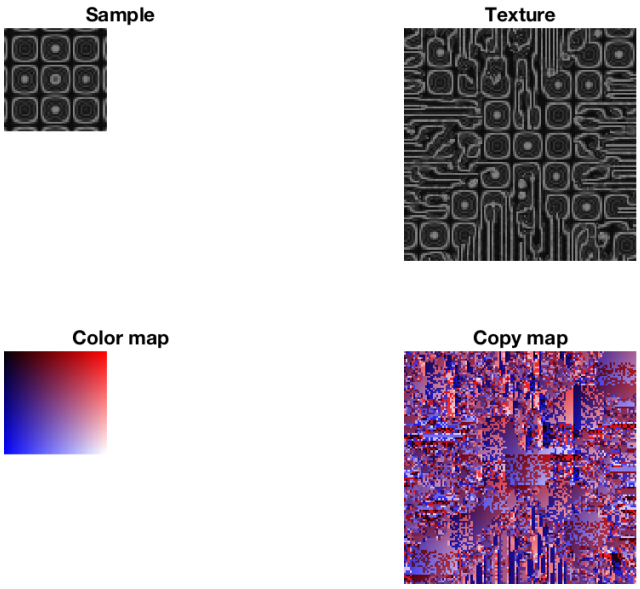


Figure 2.b Blobs picture with patch size of 7 and a tolerance of 0.2

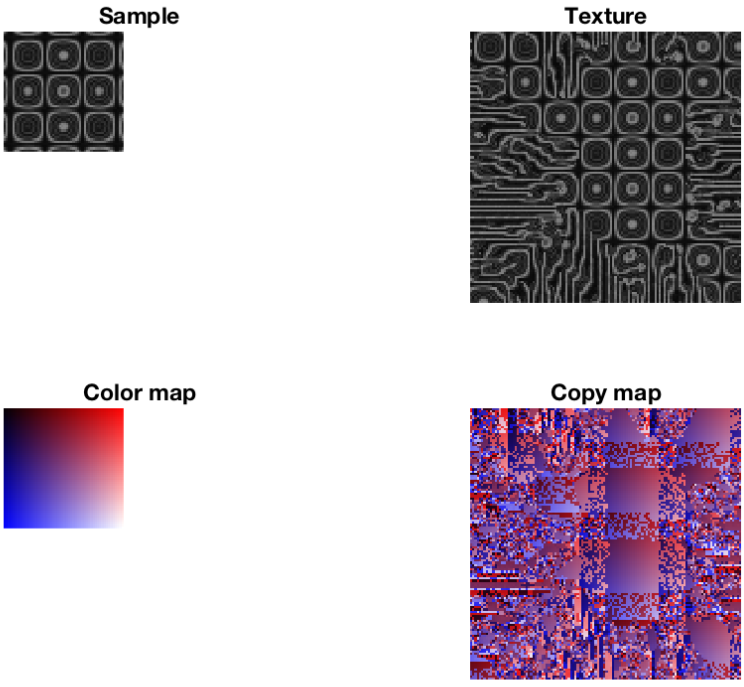


Figure 2.c Blobs picture with patch size of 7 and a tolerance of 0.3

Other examples are shown in **figure 3** for the chess image with different patch sizes.

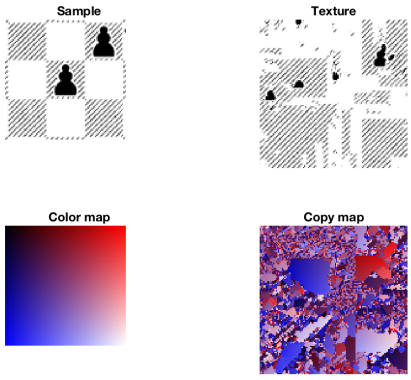


Figure 3.a Patch of 9 and tolerance 0.1

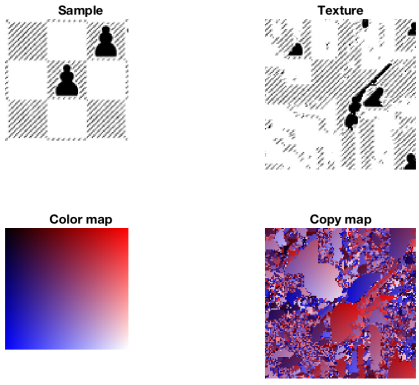


Figure 3.b Patch of 9 and tolerance 0.3

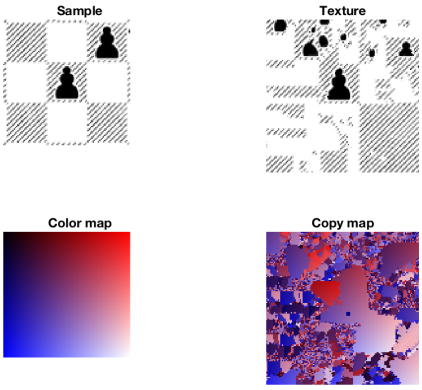


Figure 3.c Patch of 11 and tolerance 0.1

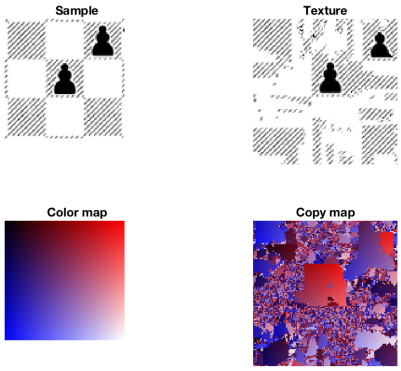
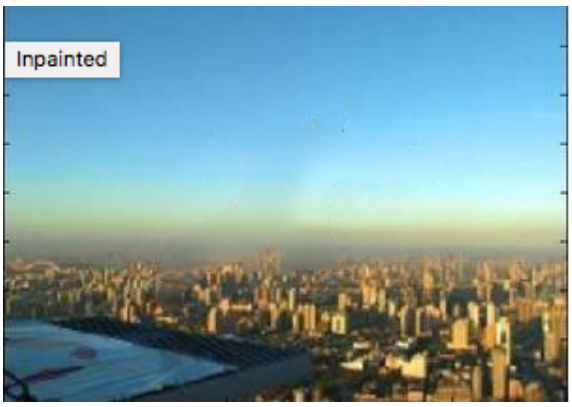


Figure 3.d Patch of 11 and tolerance 0.3

Because of the lack of computational power, we were not able to try with higher patches or in the other samples like bricks, or D1.

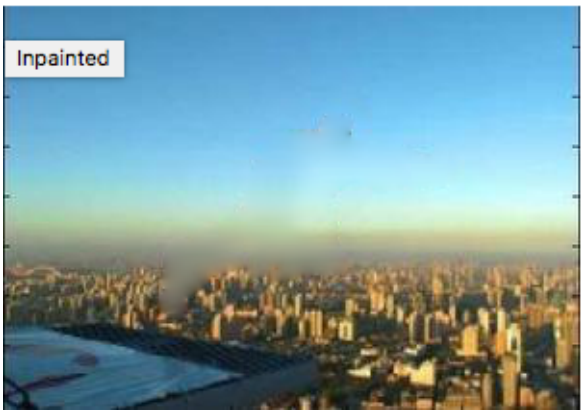
Exercise 2.

As shown in **figure 4**, it seems that the used method has a relation with the way the objects gets deleted (it has more blur depending on the method).



4.a nlmeans method **(success)**

4.b nlmedians method



4.c nlpoisson method **(failure)**

4.d Original image

Figure 4. Changing inpainting method only.

On the other hand, we also tried to delete the other objects in the image, and it seems that the best method to fill the space is the nlmedians, see **Figure 5**.

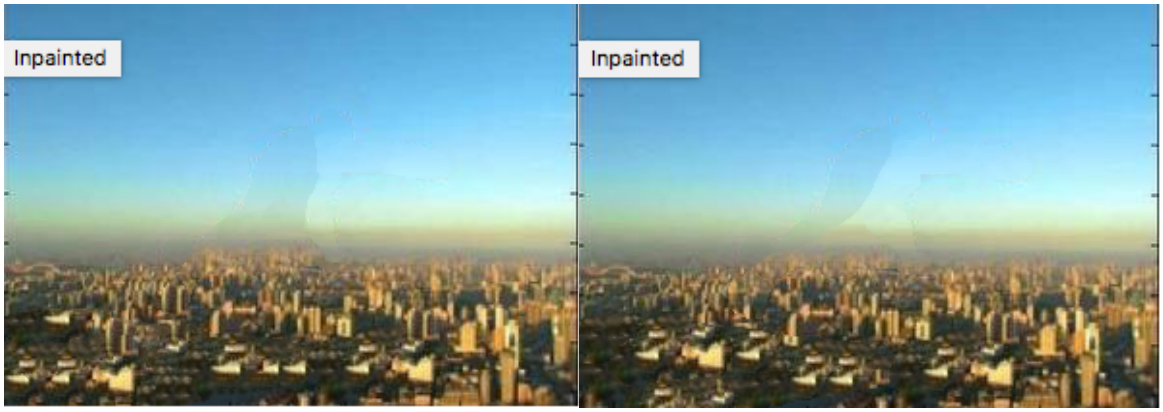
5.a nlmeans method **(failure)**

5.b nlmedians method (**success**)

5.c nlpoisson method **(failure)**

Figure 5. Changing method with two objects removed.

For the original image, it seems that small details are really important to reproduce the lost buildings. So with a small patch size the image looks more realistic as shown in **Figure 6**.



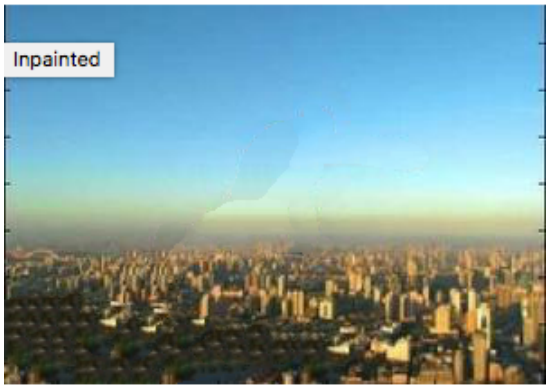
6.a Patch size 3 **(sucess)** **6.b** Patch size 7 (**success**)

6.c Patch size 13 **(semi-success)**

Figure 6. Changing patch size with two objects removed and nlmedians method.

Finally, changing the number of scales seems to be really important, in the sense that increasing the number of scales with the nlmedians method tend to support darker colors, as a result the objects just deforms and the image look worse, see **Figrue 7**.

7.a Number of scales: 5 **(sucess)** **7.b** Number of scales: 15 (**failure**)



7.c Number of scales: 30 **(failure)**

Figure 7. Changing number of scales with two objects removed and nlmedians method.