CS371 Digital Image Processing
Exercise 1
Image Resolution Manipulation

Alexandros Angelakis
csd4334@csd.uoc.gr

November 5, 2022

The code was written and tested in Octave-7.2.0, as I was not able to install MATLAB.

1 Sub-sample and up-sample using Nearest Neighbour

In this section we are going to compare the restored (using the nearest neighbour method) images from the two functions (my function and Matlab's) based on two values, the mean approximation error and the local difference. In gerenal, interpolating with the nearest neighbour method is not the best (look at the demonstrations, pages 10-end), but it depends on what you want to do with the image. The quality of the up-sampled image is very low, we lose a lot of information of the image, especially in the more detailed images (big changes between pixels), thus the texture of the images plays a significant role on the restoration quality. For example let's look at two images, one has a lot of details and the other one doesn't (for more examples go to the "Demonstrations" section).

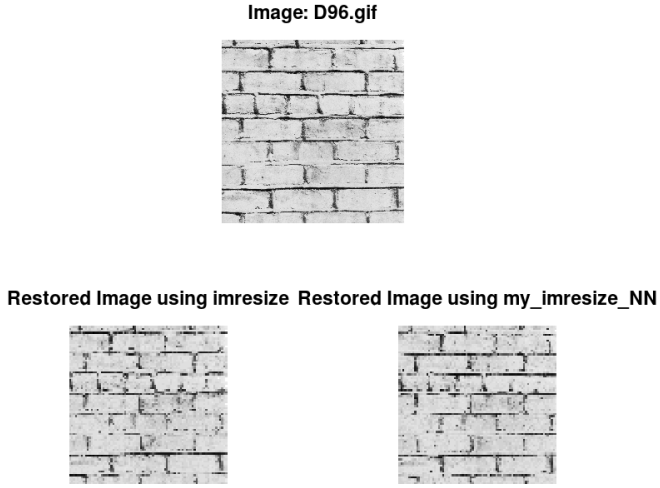

Figure 1: Image with no big details

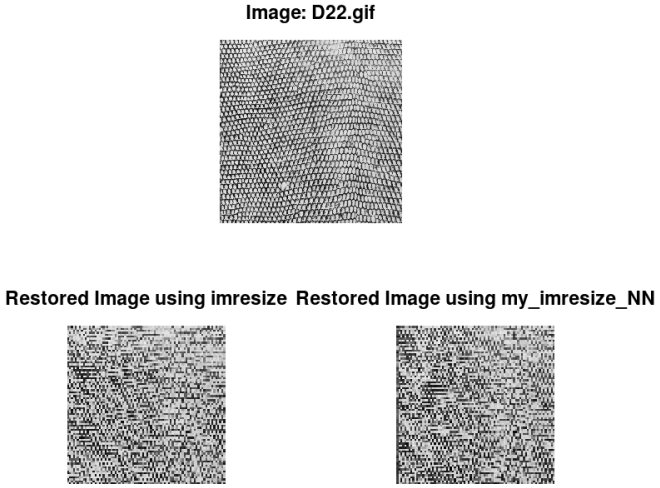

Figure 2: A very detailed image

As we can see, the restored detailed image has lost a lot of details, we can not clarify a lot of characteristics of the original image, unlike the left image, where the restoration quality is better than the right's one.

In the left figure, we can definitely distinguish the wall of bricks in both restored image by using the Matlab's function and mine. We can say that they are just the low resolution images of the original one.

In the right figure, the original image has a grid of little circles that can not be distinguished in both restored images. There might be a little bit of information, but for the most part of the image, the only thing we can see is black and white pixels represented together, forming an image.

Let's look at some figures that compare the errors and the image characteristics.

1.1 Mean Approximation Error

Comparing the errors of the restored images from my function and the Matlab's, as we can see, the Matlab's function error is smaller than mine's, except of two images. In the demonstration section we can see the images sub-sampled and up-sampled using the Nearest Neighbour method. The more complex the image is (has a lot of details), the more the error value is going to be very high, as we can see from the figure. For the most images, the error difference of my function and imresize is ±0.1.

Figure 3: x axis corresponds to the specific image in Demonstrations section

1.2 Local Difference Metric

Considering the local difference metric as a means of assessing the information along the horizontal and vertical axis of an image, we can take a look at the image below. The horizontal and vertical differences between the pixels (variations in the texture) result in higher error in the original image. In this image, as we can see, the differences between the pixels is always very high (from black to white and vice-versa), so we expect the metric value for this image to be very high and in the reconstructed ones to be lower, since we can not retain all the fine information (only the coarse ones).

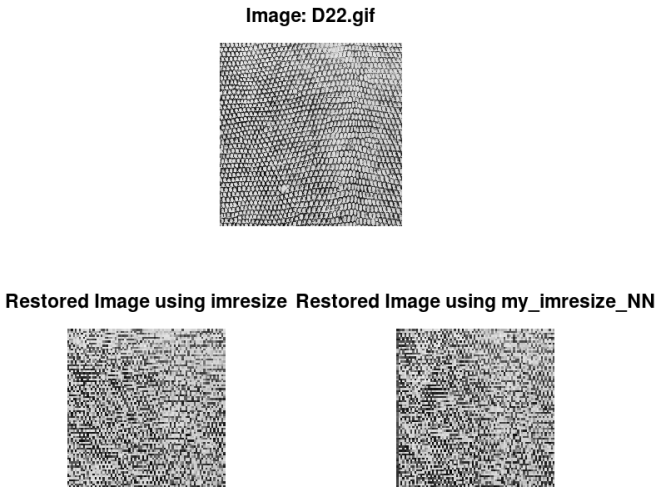

Figure 4: x = 5 in the plot below

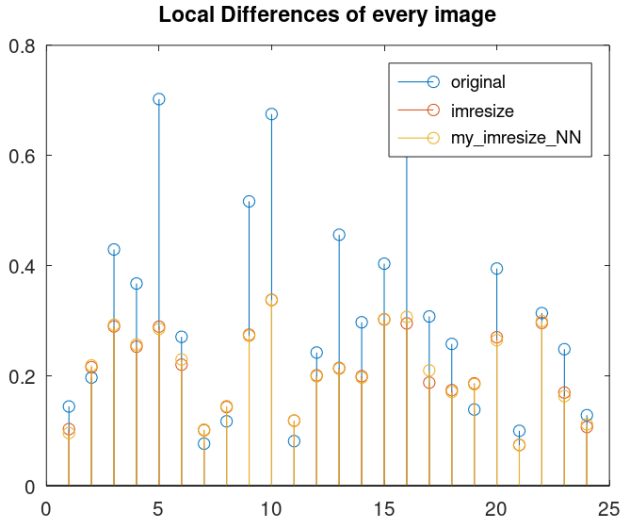

Figure 5: x axis corresponds to the specific image in Demonstrations section

2 Sub-sample using Nearest Neighbor and up-sample using Bi-linear Interpolation

In this section we are going to compare the restored (using the bi-linear method) images from the two functions (my function and Matlab's) based on two values, the mean approximation error and the local difference. Interpolating with the bi-linear method has better results with our images (look at the demonstrations). The quality of the up-sampled images is better than interpolating with the nearest neighbour method. Let's take the previous examples, an image that has a lot of details and one that doesn't and let's exactly see the difference.

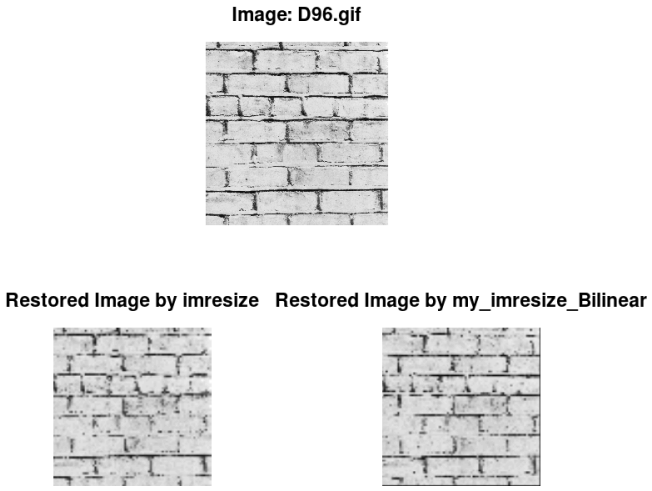

Figure 6: Image with no big details

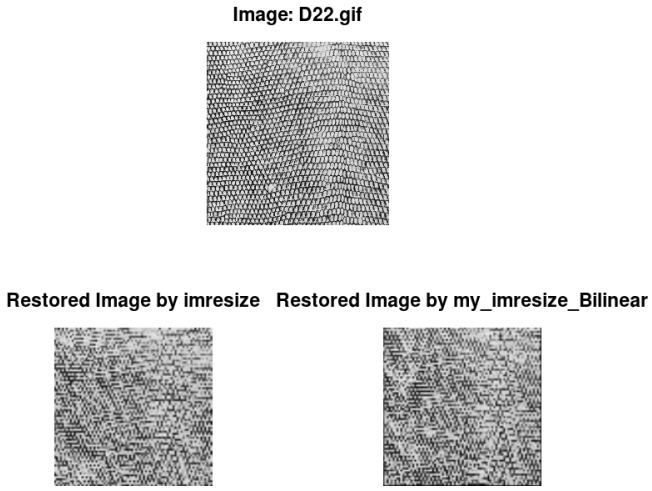

Figure 7: A very detailed image

As one can see, the pixels in both images are placed more smoothly than the previous ones, resulting in better quality of the images. They still have lost a lot of details, but we can easily clarify most of the characteristics of the original images.

To be more precise, in the left figure, we can definitely distinguish better the wall of bricks in both the restored images, great approximation of the original image.

In the right figure, the grid of little circles in the original image can not be seen in the restored images even though the pixels are smoothly placed in the image. There might be a little bit of information, but for the most part of the image, the only thing we can see is black and white pixels represented together, forming an image.

To conclude, the texture of the image plays a major role on the restoration quality even when interpolating, using the Bi-linear method.

Let's look at some figures that compare the errors and the image characteristics.

2.1 Mean Approximation Error

Comparing the errors of the restored images from my function and the Matlab's, as we can see, the Matlab's function error is a little bit smaller than mine's, except of two-three images. In the demonstration section we can see the images sub-sampled with Nearest Neighbour and up-sampled with Bi-linear. The more complex the image is (has a lot of details), the more the error value is going to be very high, as we can see from the figure. For the most images, the error difference of my function and imresize is ± 0.03.

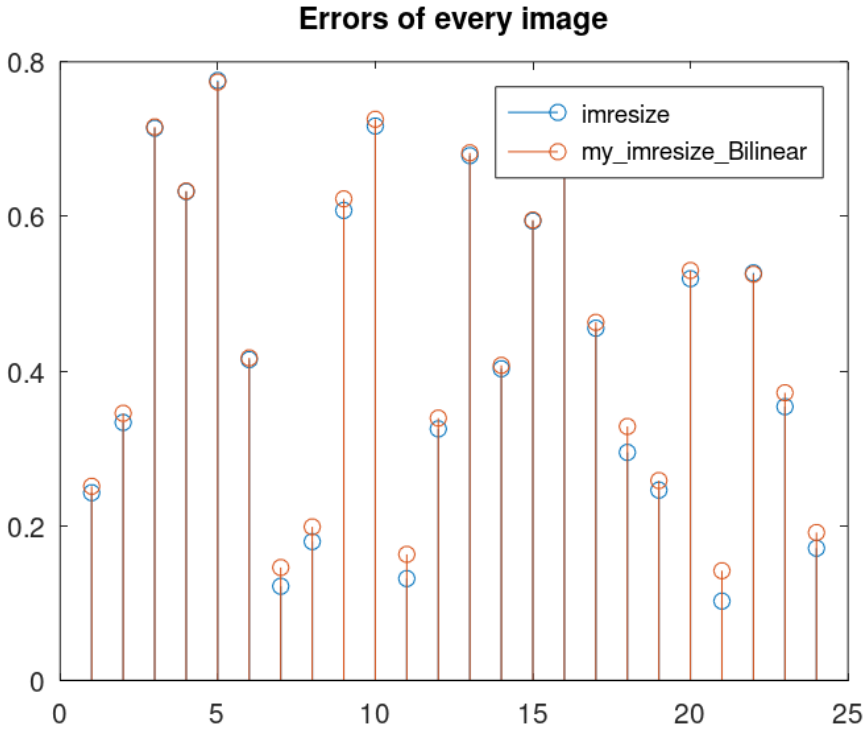

Figure 8: x axis corresponds to the specific image in Demonstrations section

2.2 Local Difference Metric

As we can see, using the Bi-linear interpolation method, the local differences of the reconstructed images are lower than the nearest neighbour method. That means, that the up-sampled images fail to retain the original information of the original images even more, although the errors for those images is lower than the nearest neighbour. By observing the plot with all the local differences, we can see that all the reconstructed images is lower with this interpolation method than the previous one.

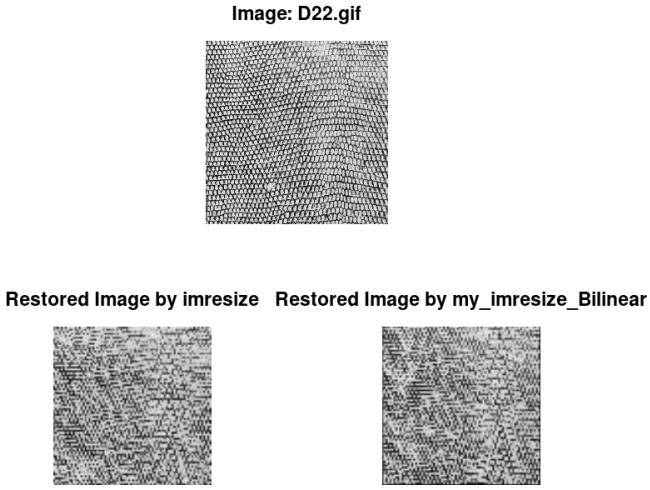

Figure 9: x = 5 in the plot below

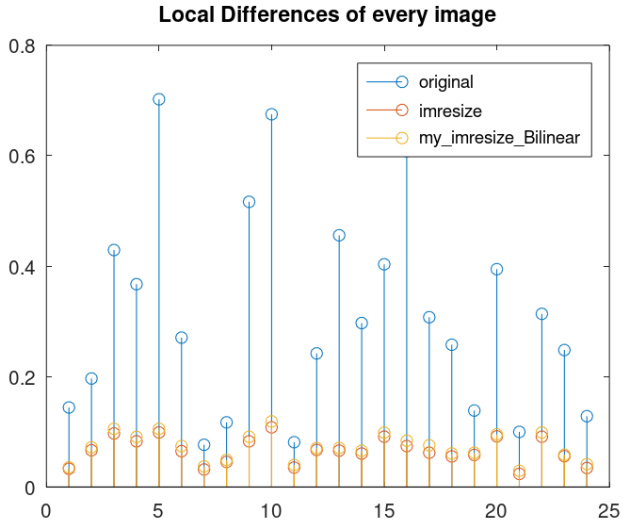

Figure 10: x axis corresponds to the specific image in Demonstrations section

3 Sub-sample and up-sample using Nearest Neighbour and Anti-aliasing

In this section we are going to compare restored images that have been applied anti-aliasing when sub-sampled (using nearest neighbour) and restored images that have not. To make our comparison clear, we will use the same example images as before.

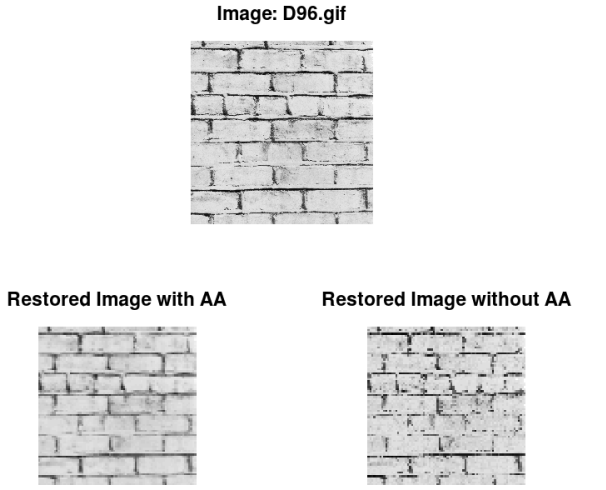

Figure 11: Image with no big details

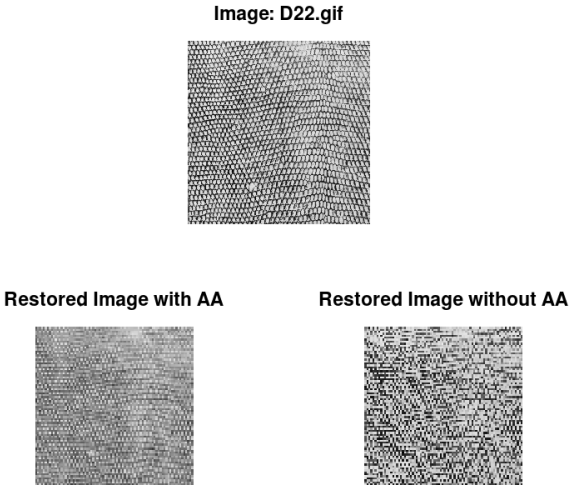

Figure 12: A very detailed image

As one can see, in the images that we have applied anti-aliasing, there is a blurriness but the quality is significantly better than without applying anti-aliasing. This happens because anti-aliasing removes the aliased edges of the images, giving it the appearance of smoother edges and higher resolution. Of course, the texture of the images plays a role on the restoration quality but based on the above plots, we can live without it. We can also show this based on the error and metric plots.

3.1 Mean Approximation Error

As I mentioned before, anti-aliasing gives higher resolution and smoother edges, leading to lower error values. This can be seen on plot underneath. The difference between the restored images with anti-aliasing and the restored images without is (for some images) very high. We should also not forget that the more complex the image is (has a lot of details), the more the error value is going to be very high, with or without Anti-aliasing, as we can see from the figure.

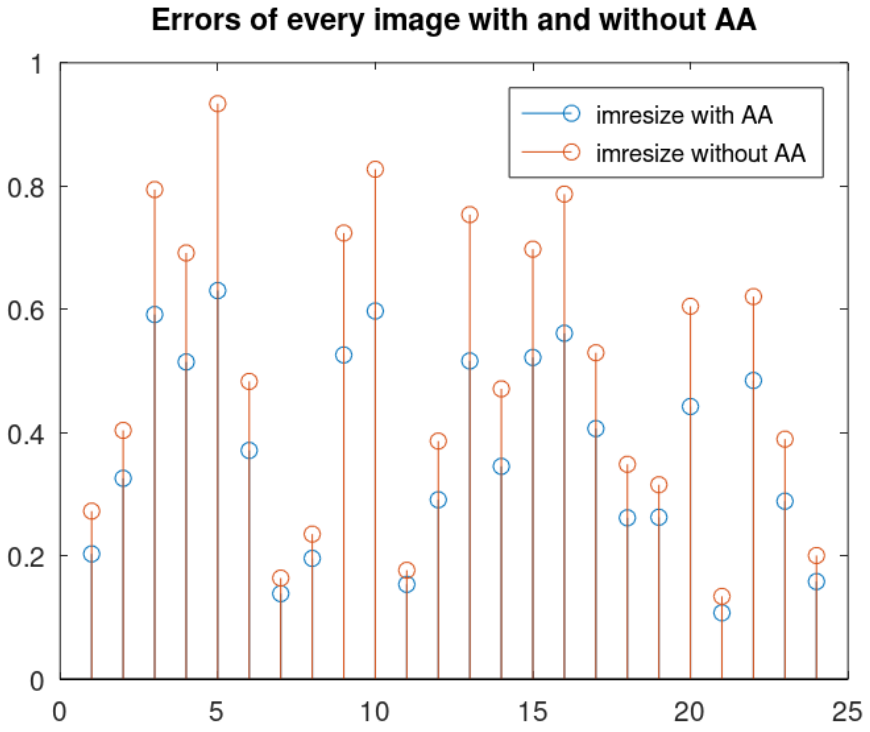

Figure 13: x axis corresponds to the specific image in Demonstrations section

3.2 Local Difference Metric

Using Anti-aliasing on our images, this bluriness that it gives to the reconstructed one is responsible for this low values on the local differences for each image. The errors have significantly decreased, but without applying Anti-aliasing we get a better approximation of the pixels of our original image. By looking at it with our eyes we can say that that's impossible, but mathematically this is the truth :D.

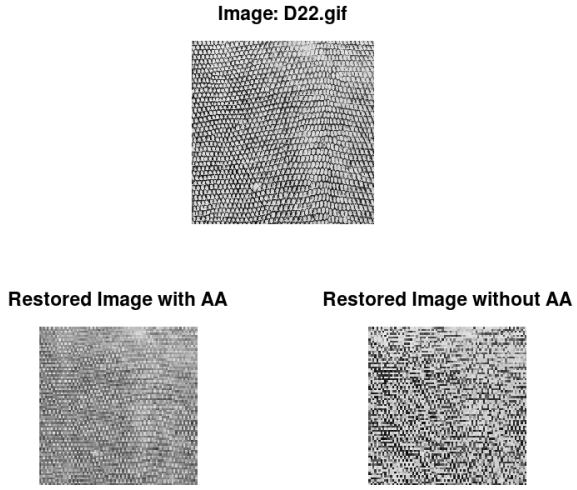

Figure 14: x = 5 in the plot below

Figure 15: x axis corresponds to the specific image in Demonstrations section

4 Demonstrations

The number of the image in the caption corresponds to the number of the image on the x axis of the plots with the errors and the image characteristics.

4.1 Sub-sampling and Up-sampling using Nearest Neighbour

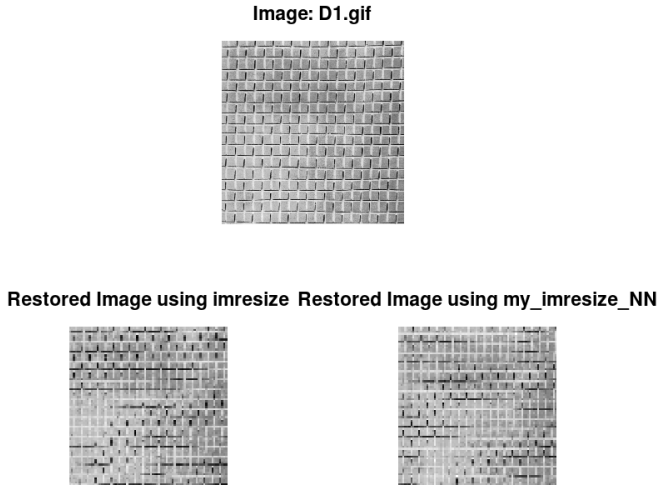

Figure 16: Image 1

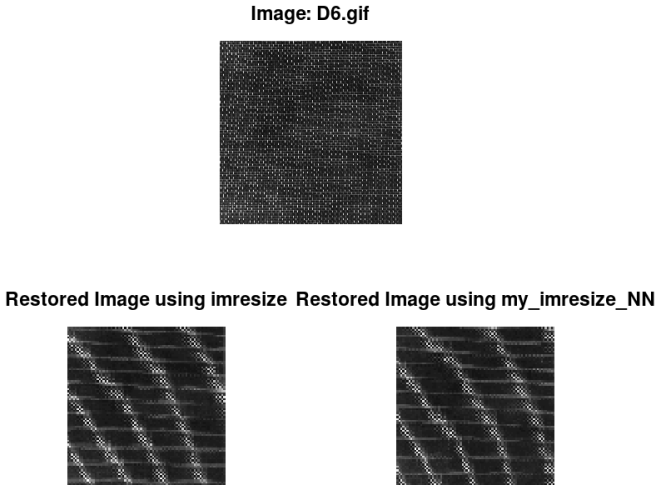

Figure 17: Image 2

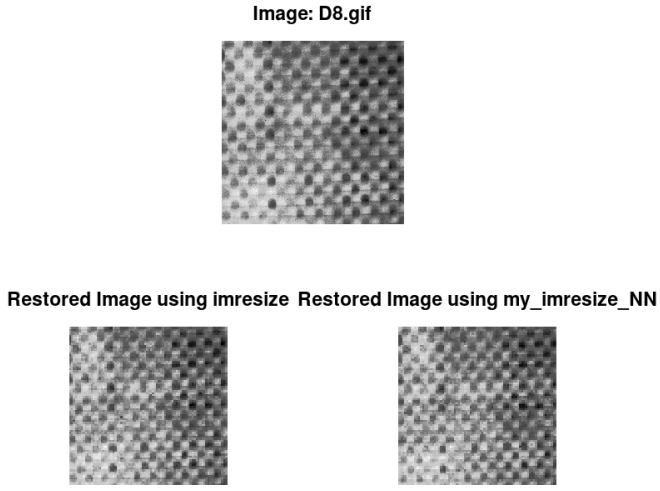

Figure 18: Image 3

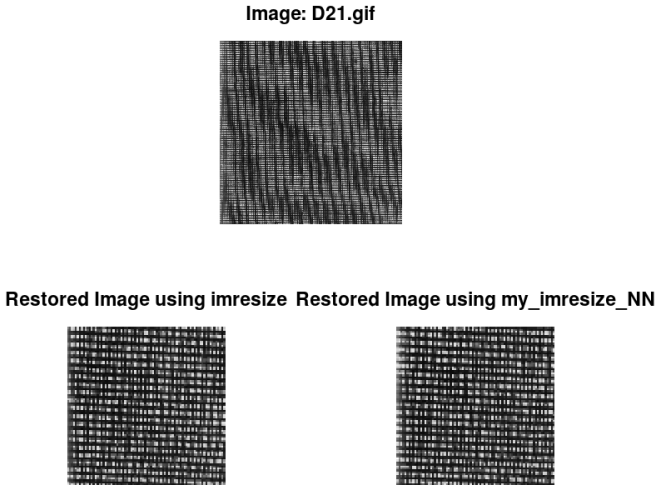

Figure 19: Image 4

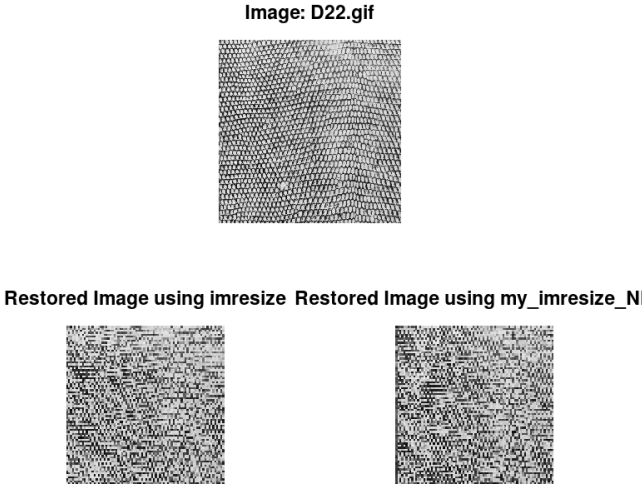

Figure 20: Image 5

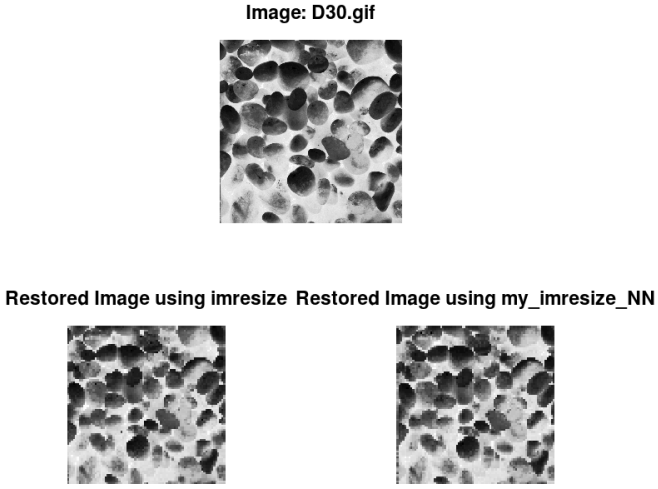

Figure 21: Image 6

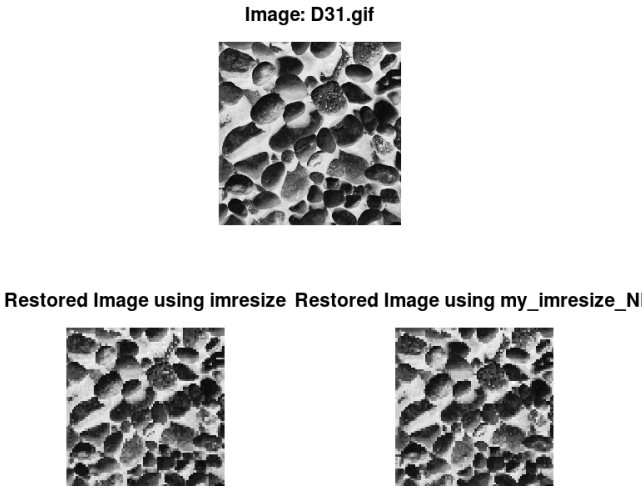

Figure 22: Image 7

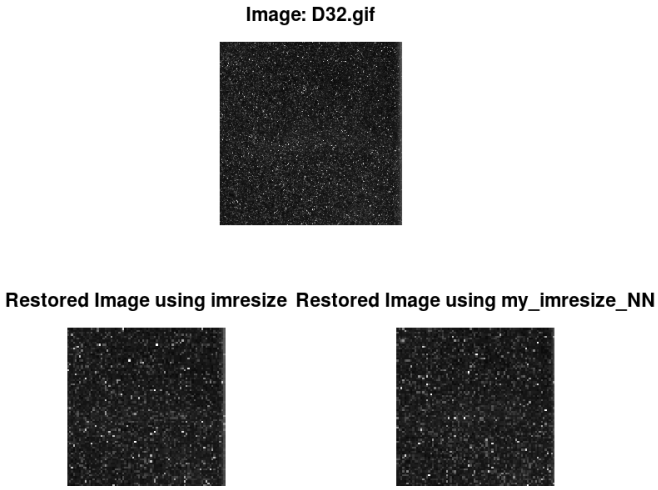

Figure 23: Image 8

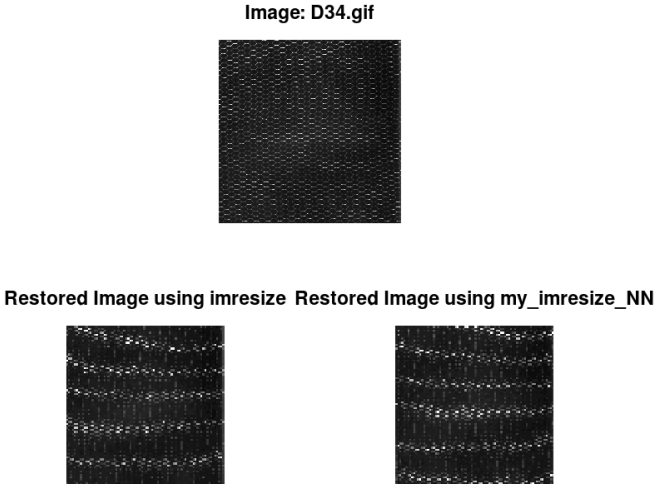

Figure 24: Image 9

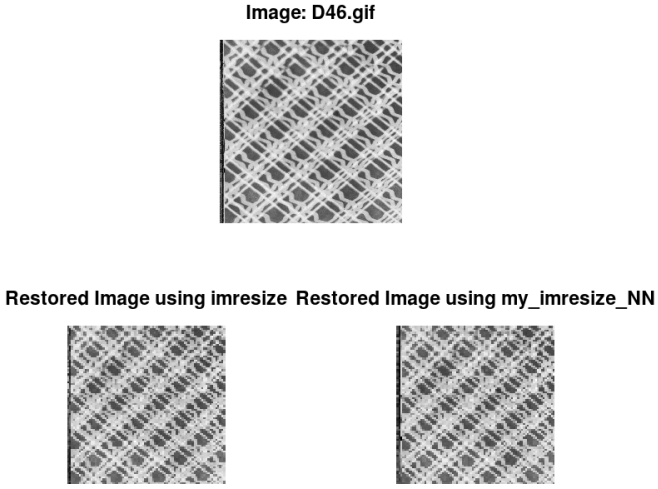

Figure 25: Image 10

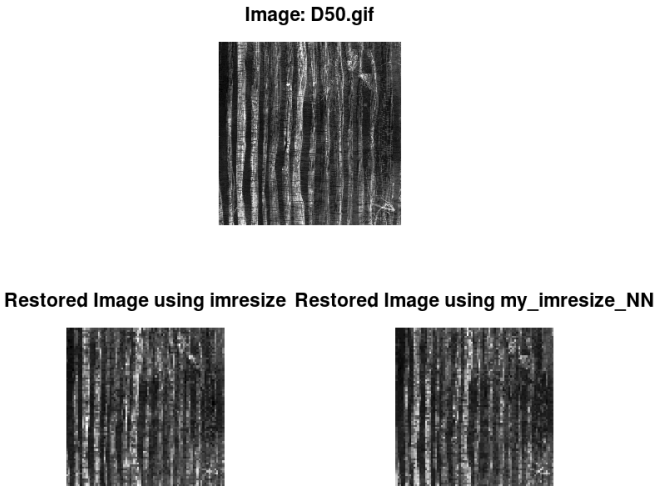

Figure 26: Image 11

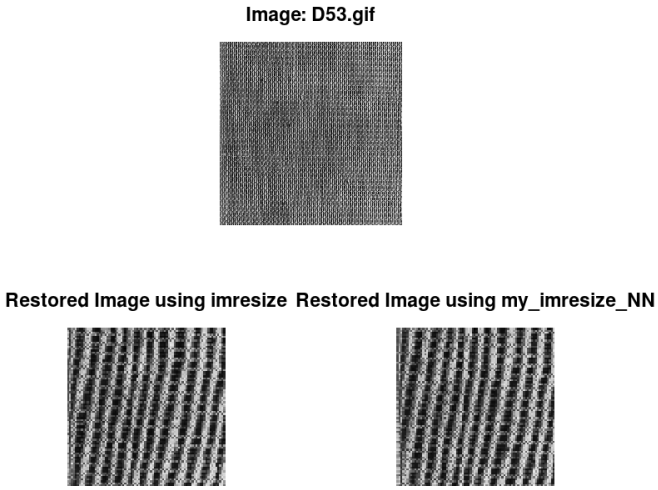

Figure 27: Image 12

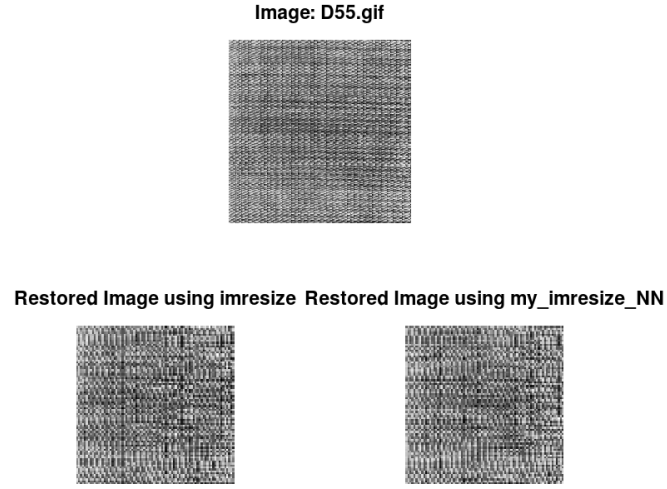

Figure 28: Image 13

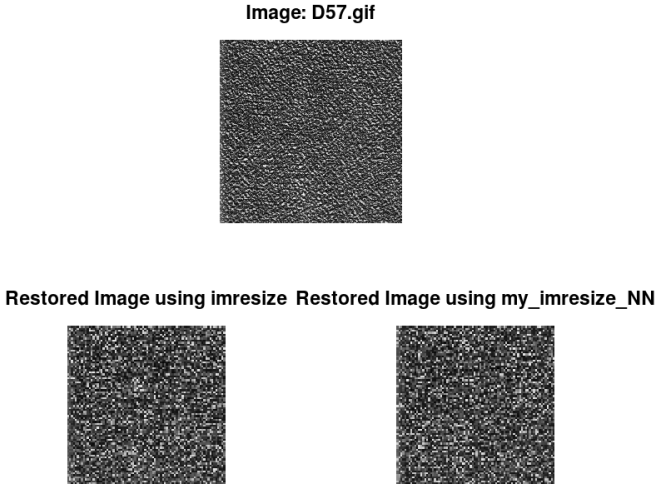

Figure 29: Image 14

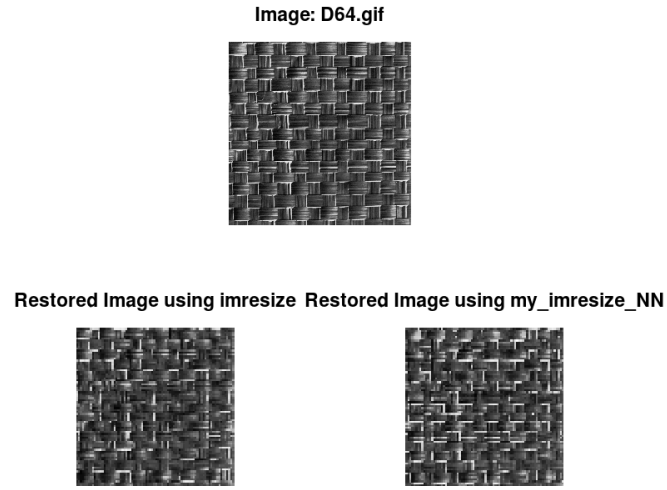

Figure 30: Image 15

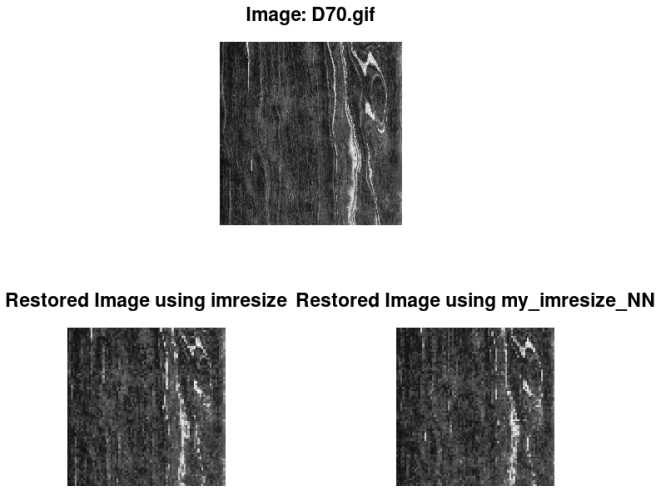

Figure 31: Image 16

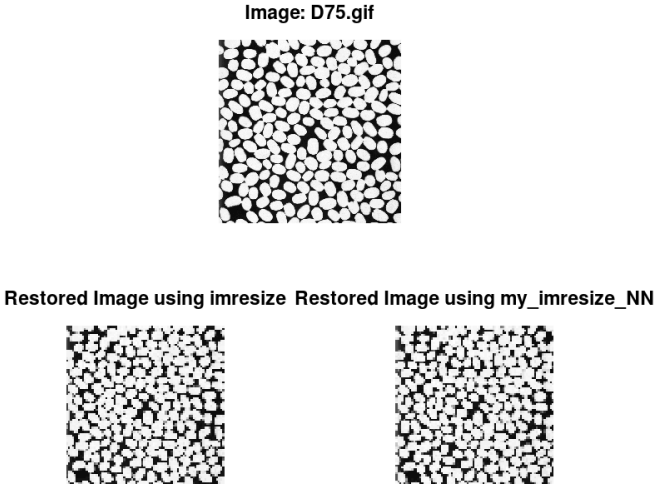

Figure 32: Image 17

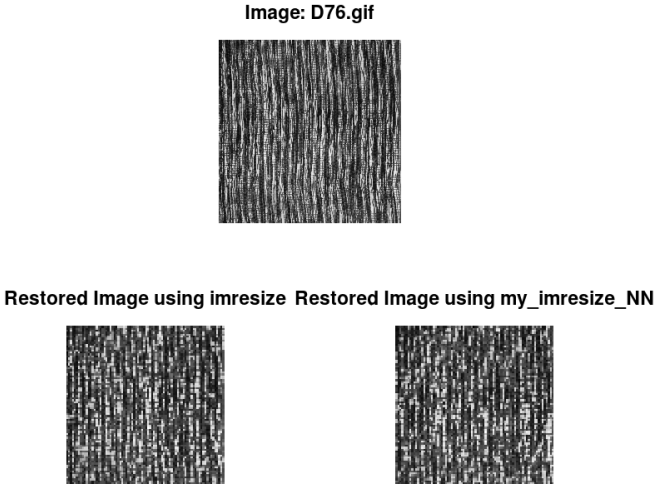

Figure 33: Image 18

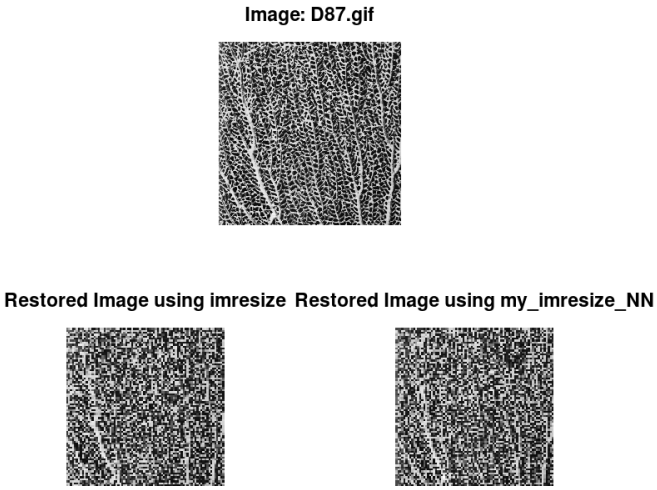

Figure 34: Image 19

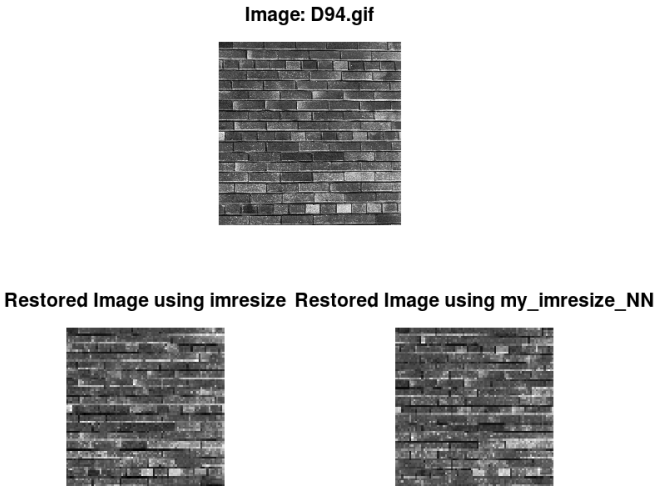

Figure 35: Image 20

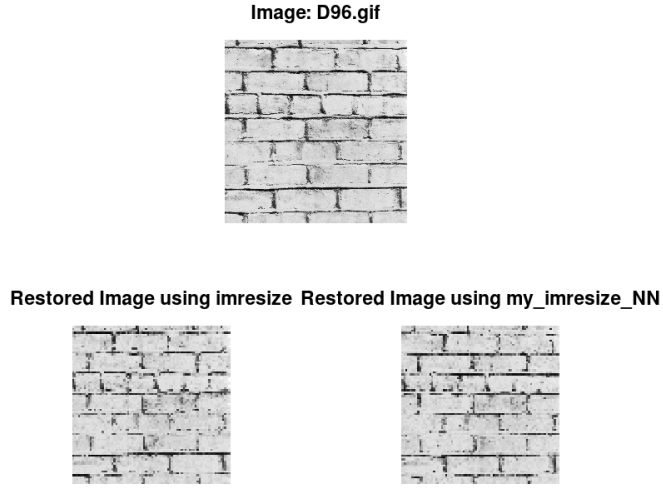

Figure 36: Image 21

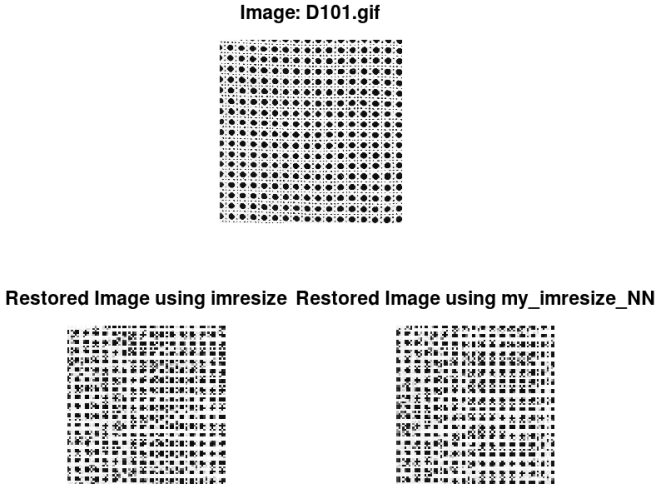

Figure 37: Image 22

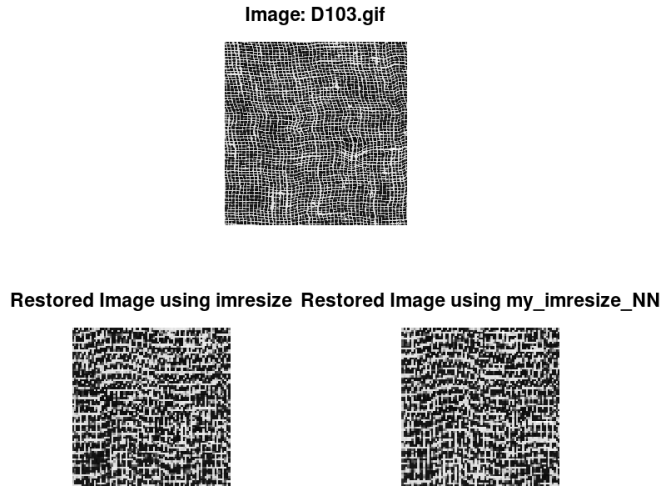

Figure 38: Image 23

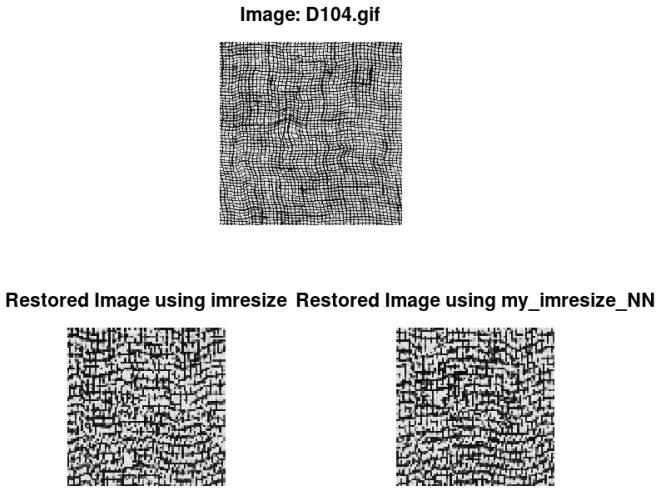

Figure 39: Image 24

4.2 Sub-sampling using Nearest Neighbour and Up-sampling using Bi-linear

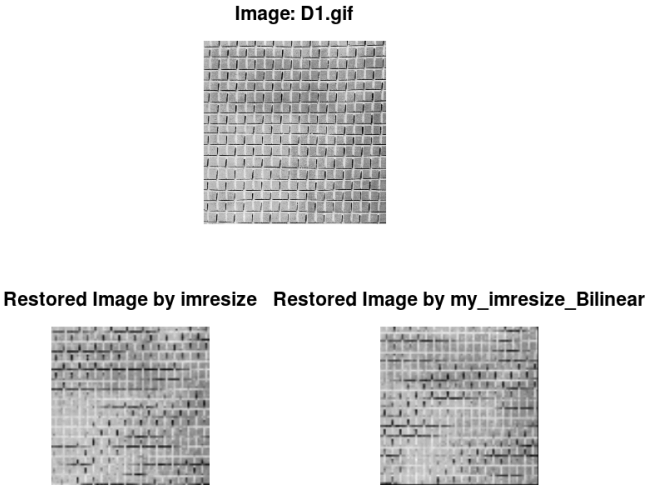

Figure 40: Image 1

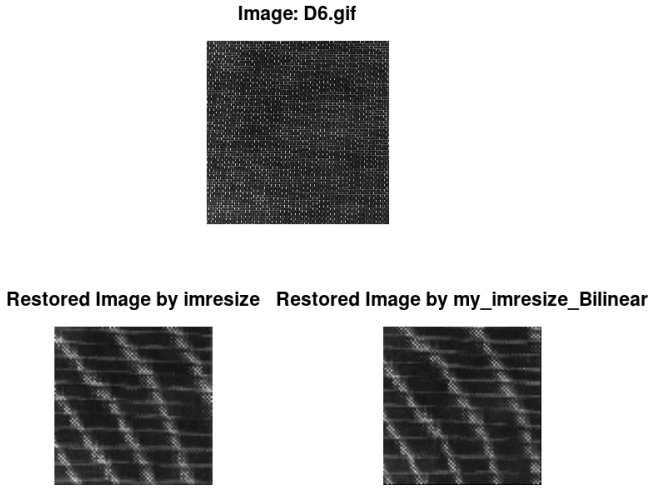

Figure 41: Image 2

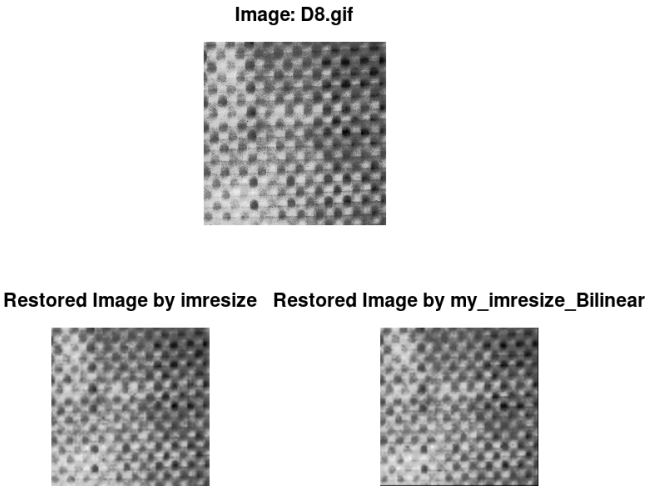

Figure 42: Image 3

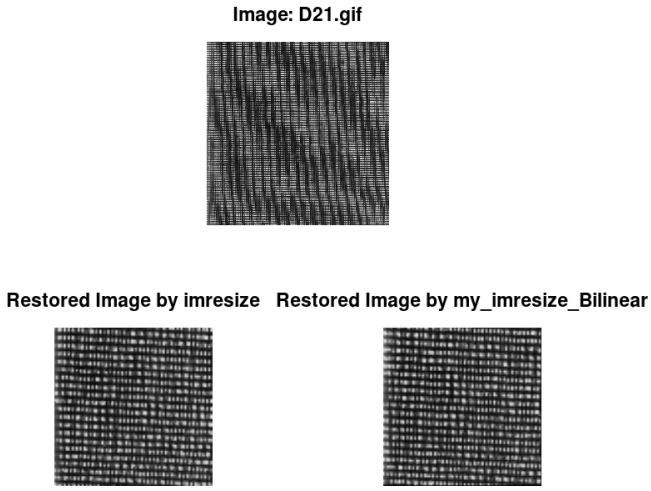

Figure 43: Image 4

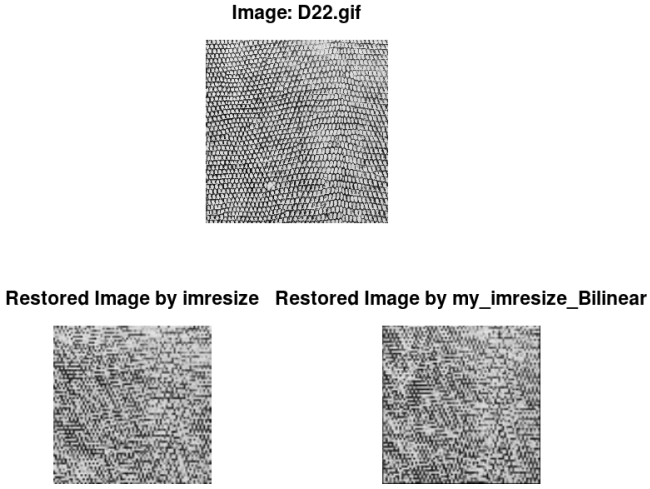

Figure 44: Image 5

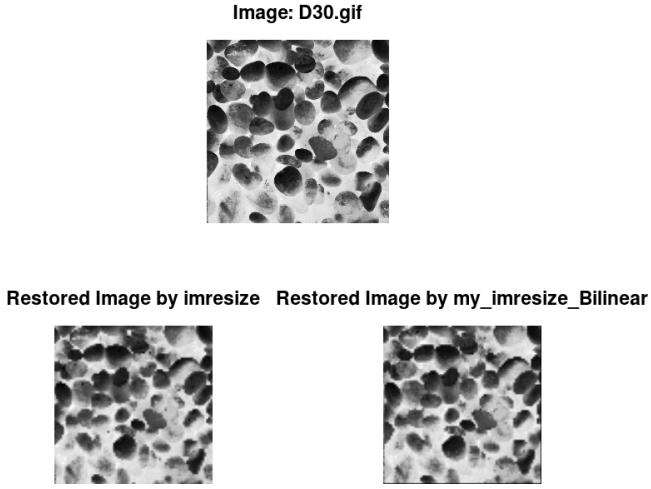

Figure 45: Image 6

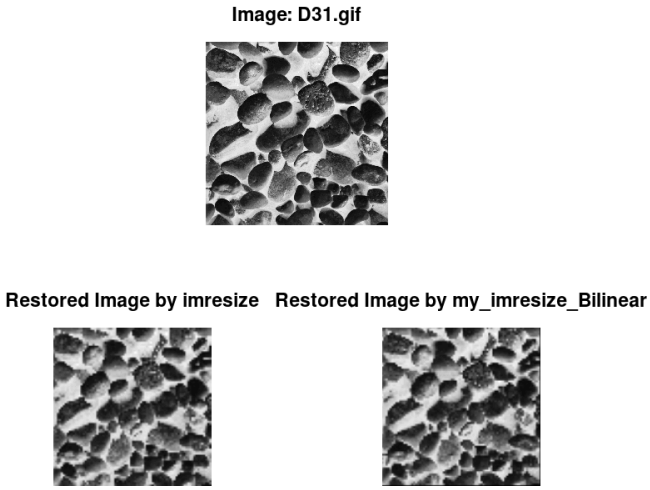

Figure 46: Image 7

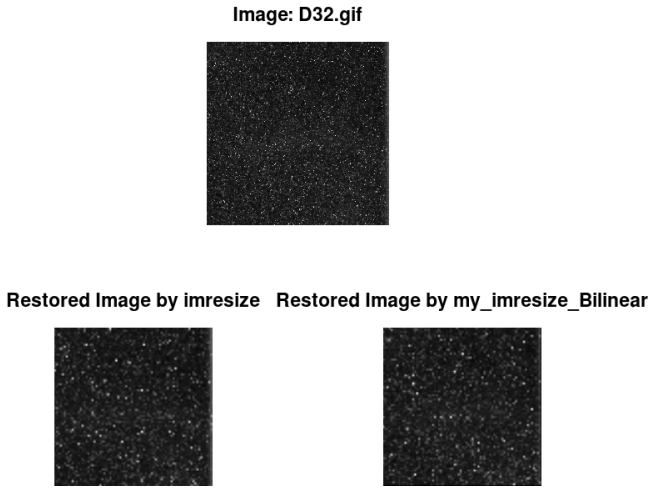

Figure 47: Image 8

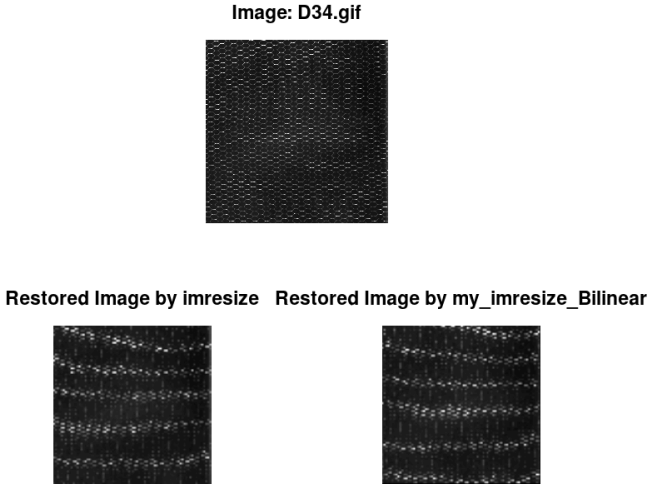

Figure 48: Image 9

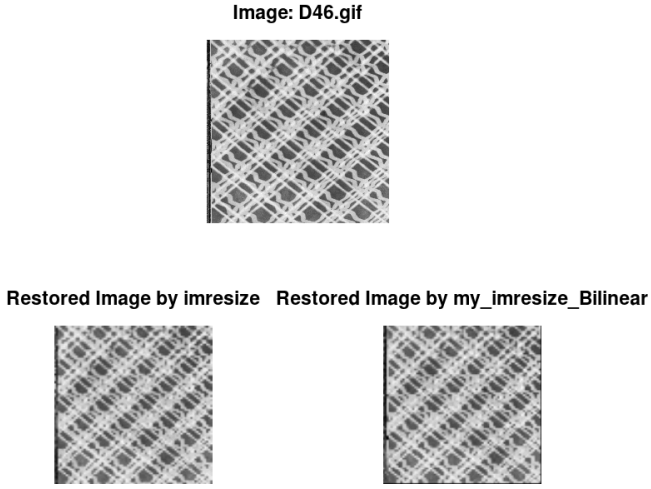

Figure 49: Image 10

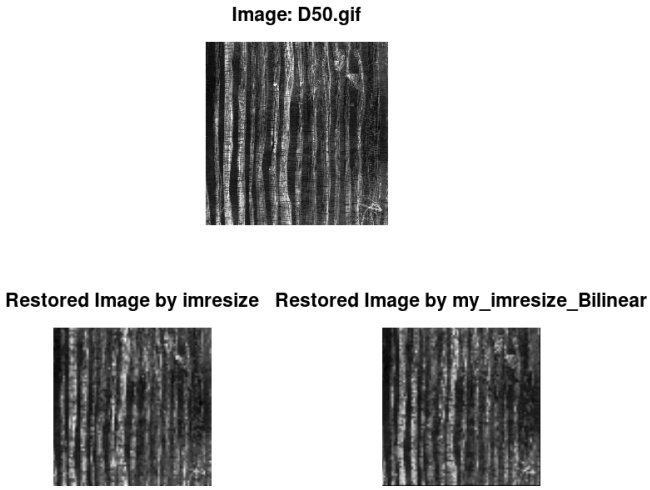

Figure 50: Image 11

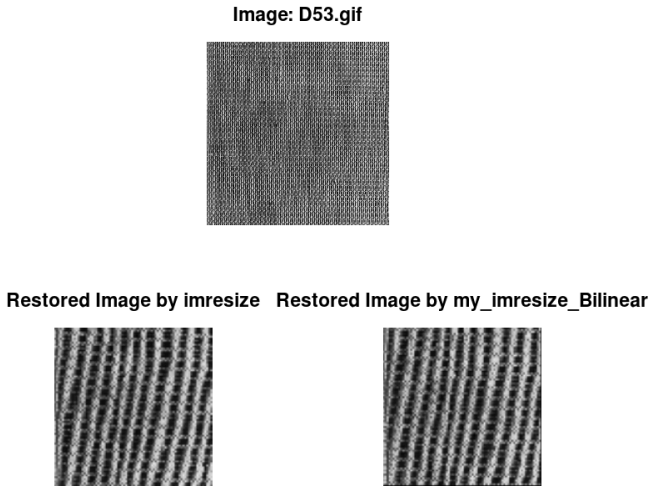

Figure 51: Image 12

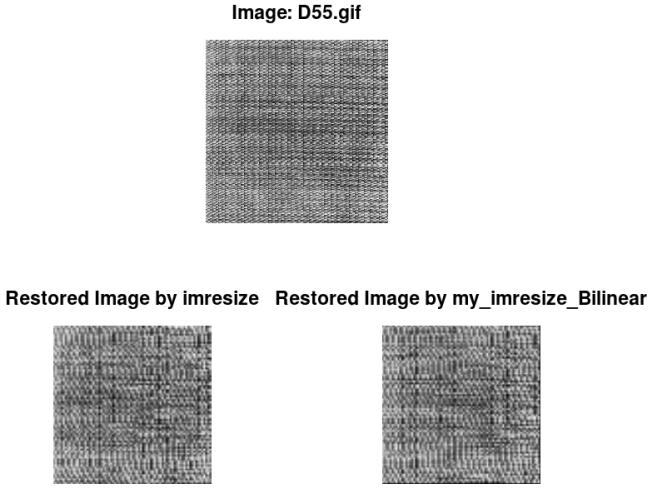

Figure 52: Image 13

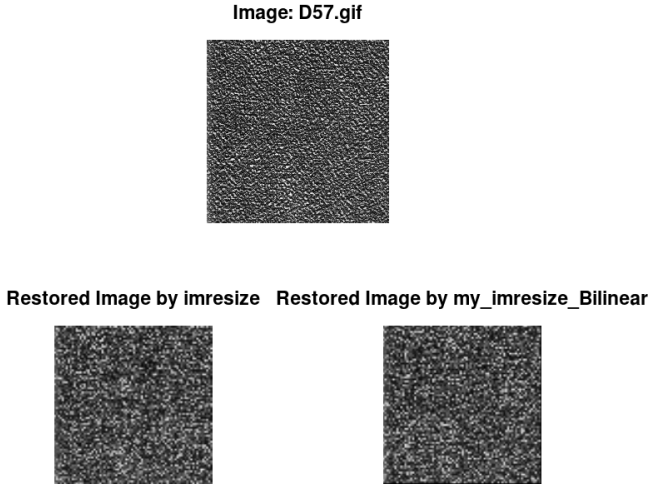

Figure 53: Image 14

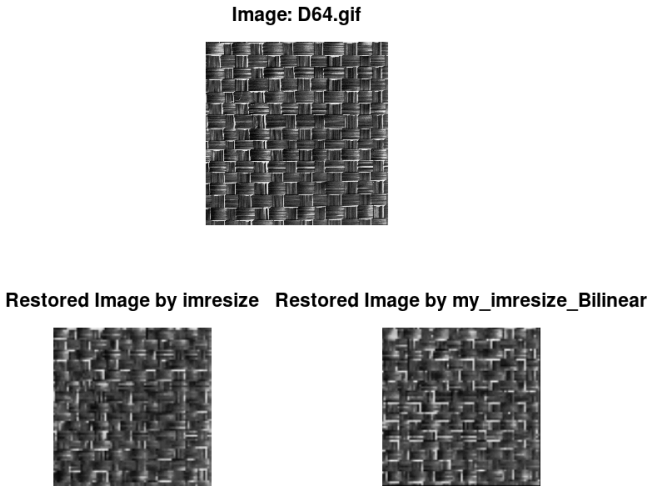

Figure 54: Image 15

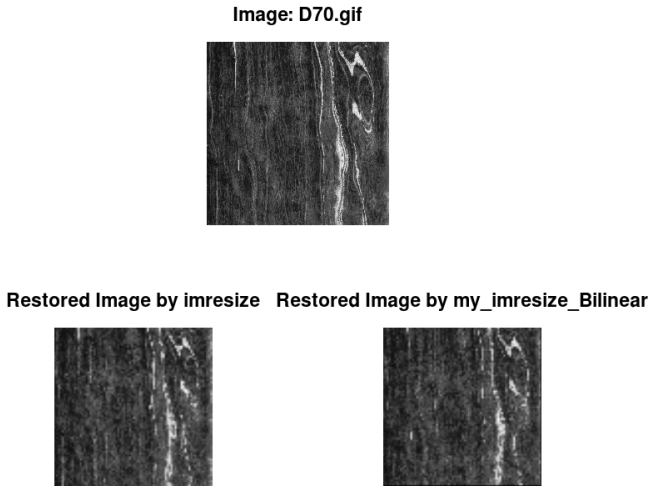

Figure 55: Image 16

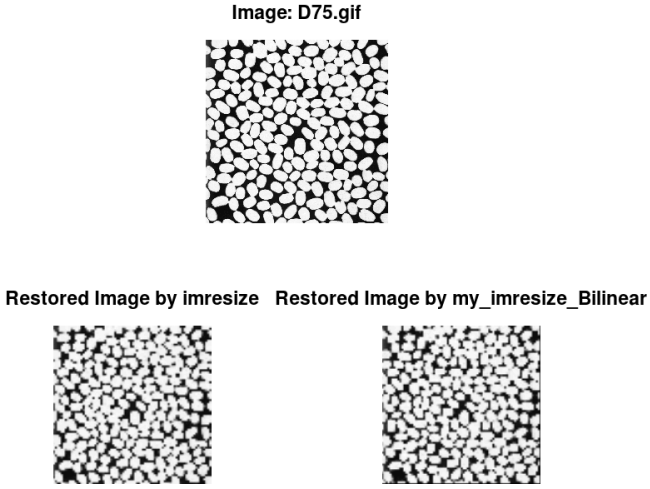

Figure 56: Image 17

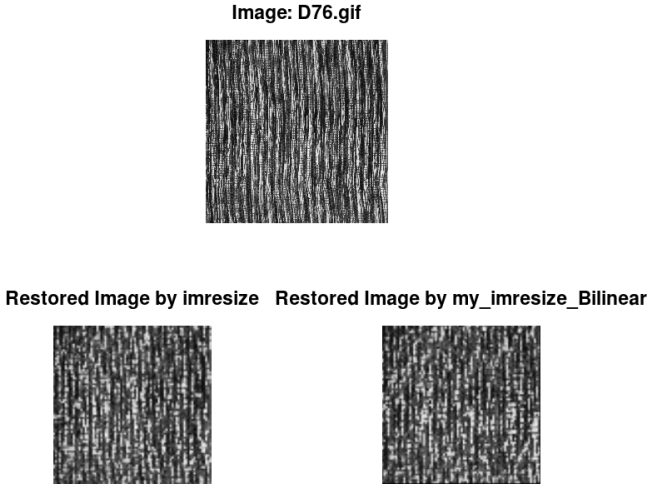

Figure 57: Image 18

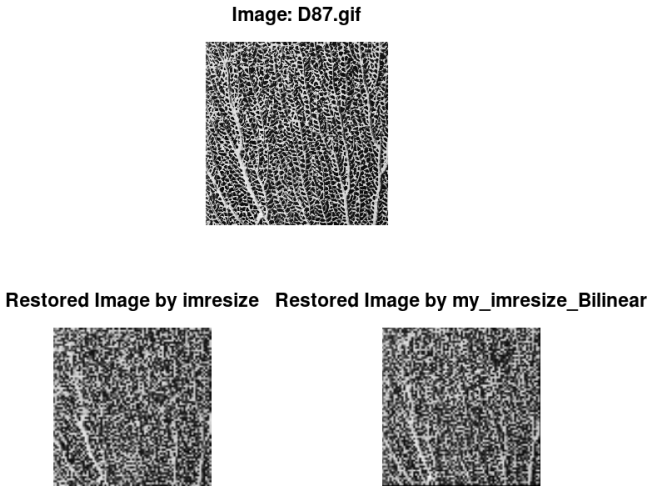

Figure 58: Image 19

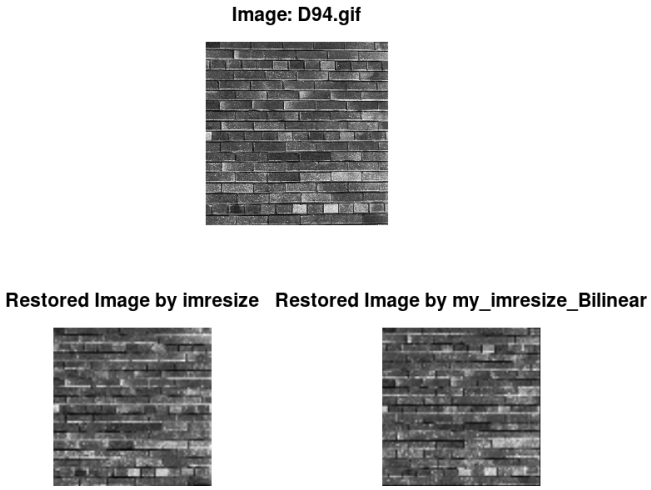

Figure 59: Image 20

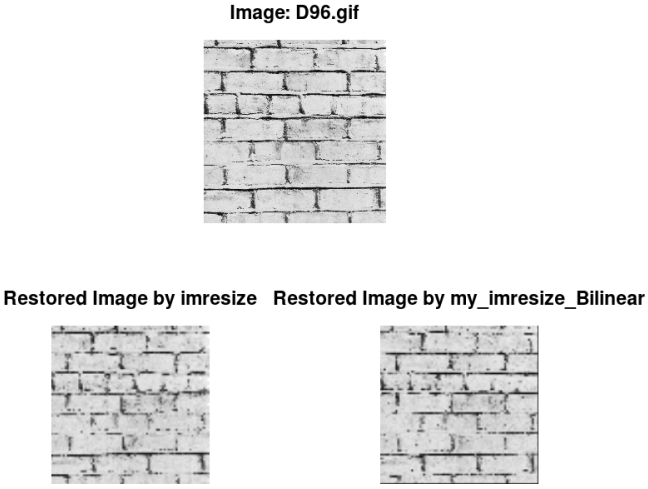

Figure 60: Image 21

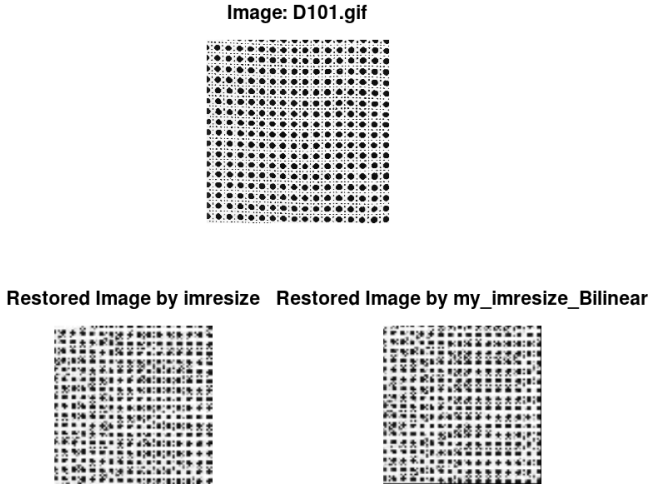

Figure 61: Image 22

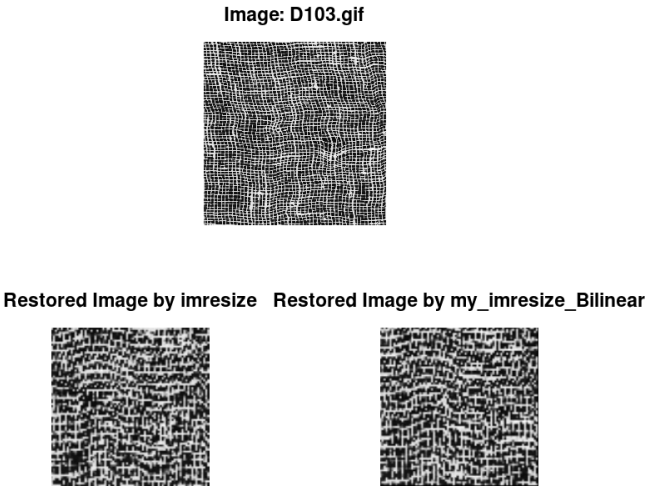

Figure 62: Image 23

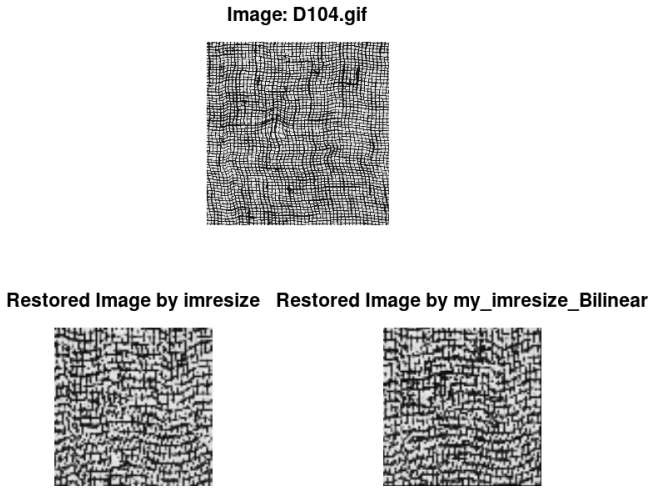

Figure 63: Image 24

4.3 Sub-sampling and Upsampling using Nearest Neighbour and Anti-aliasing

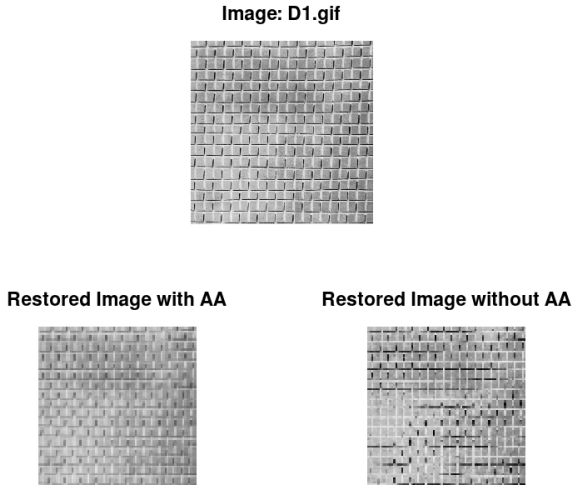

Figure 64: Image 1

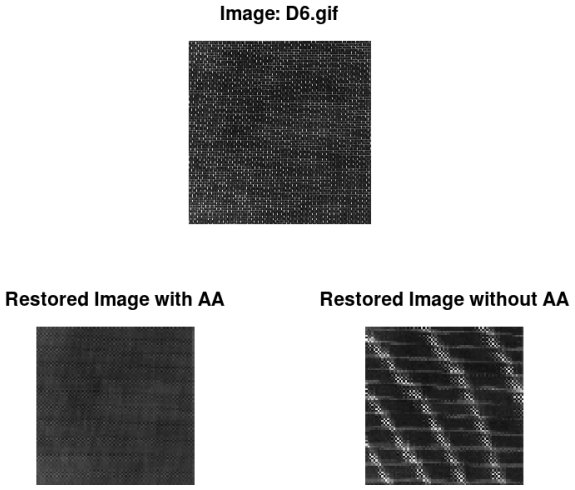

Figure 65: Image 2

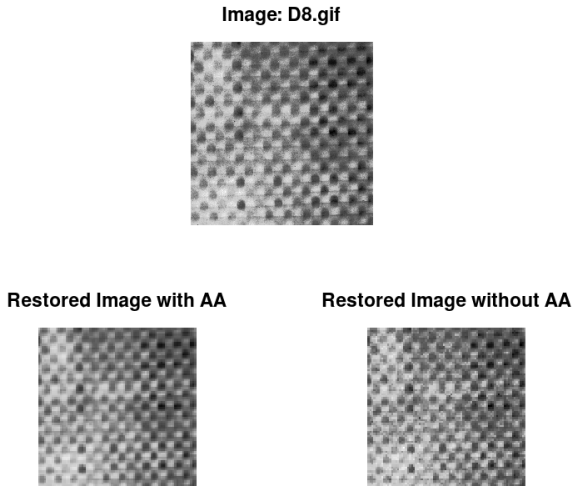

Figure 66: Image 3

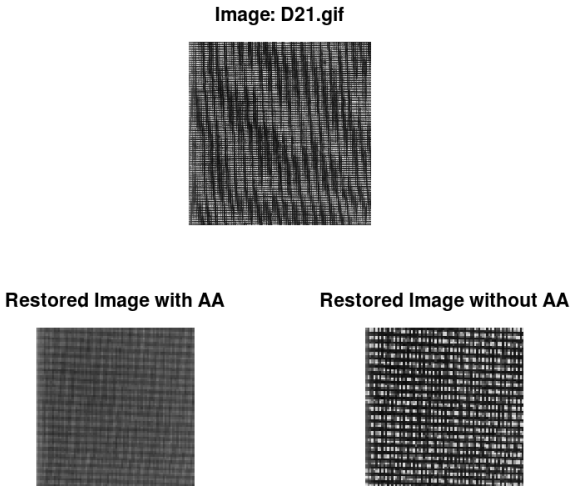

Figure 67: Image 4

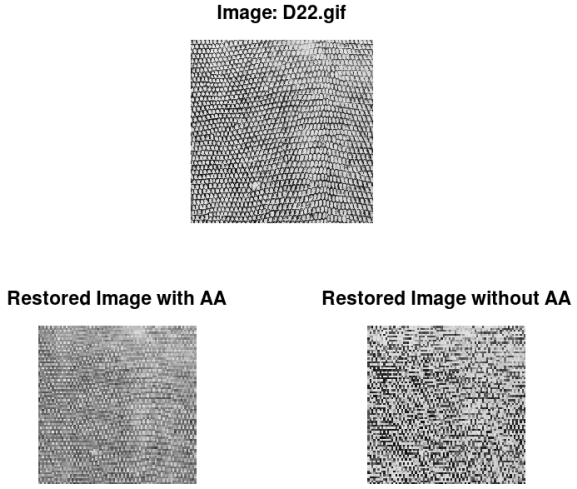

Figure 68: Image 5

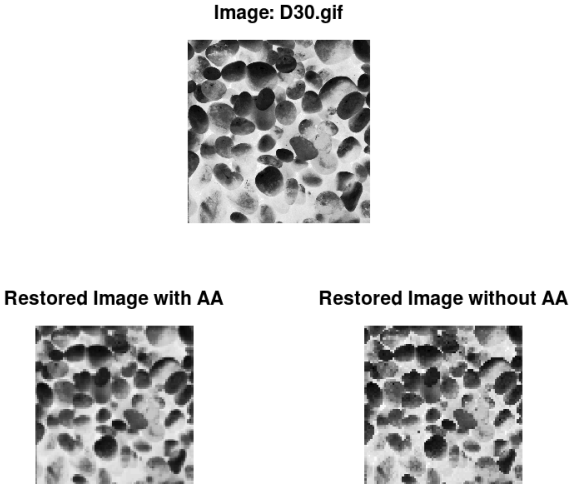

Figure 69: Image 6

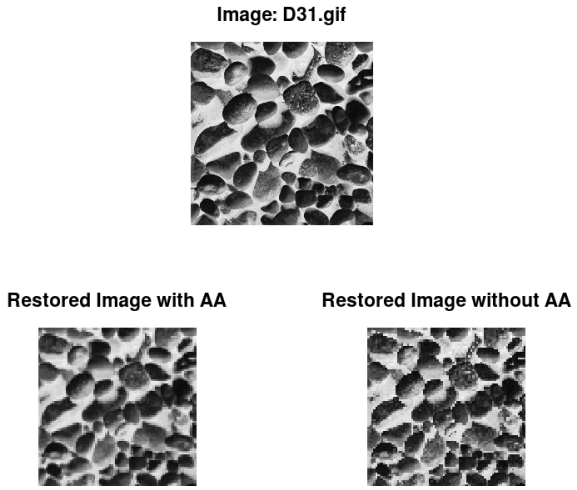

Figure 70: Image 7

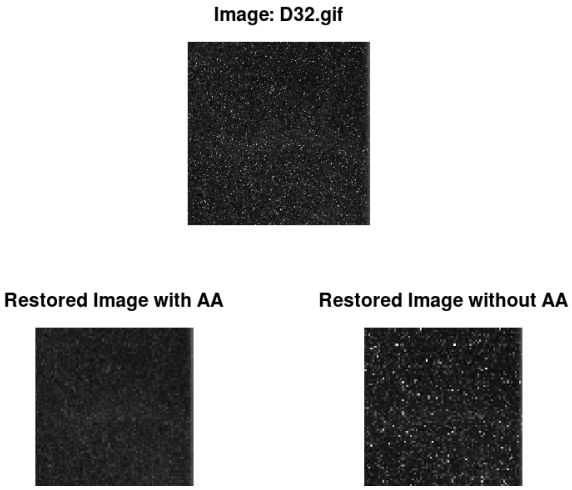

Figure 71: Image 8

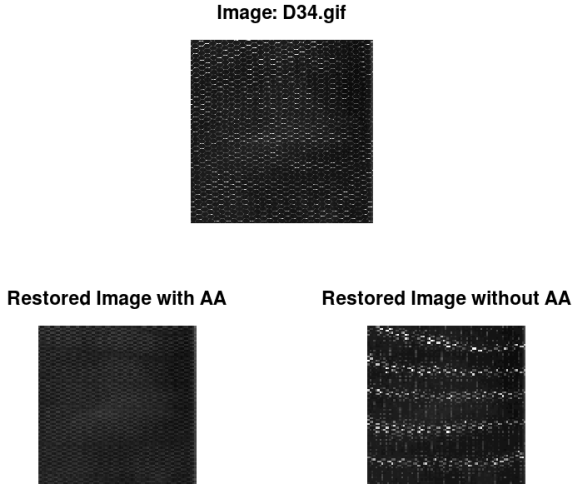

Figure 72: Image 9

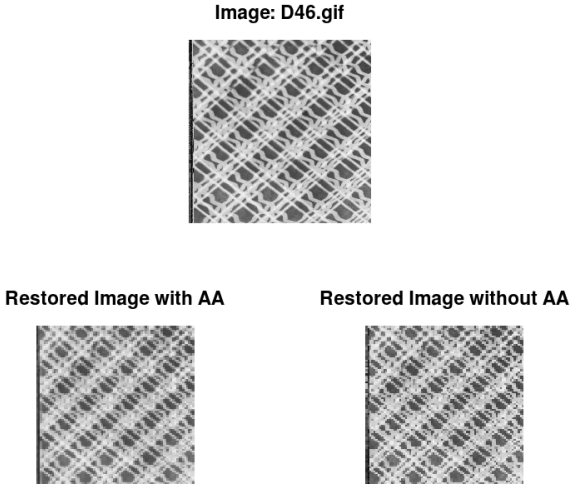

Figure 73: Image 10

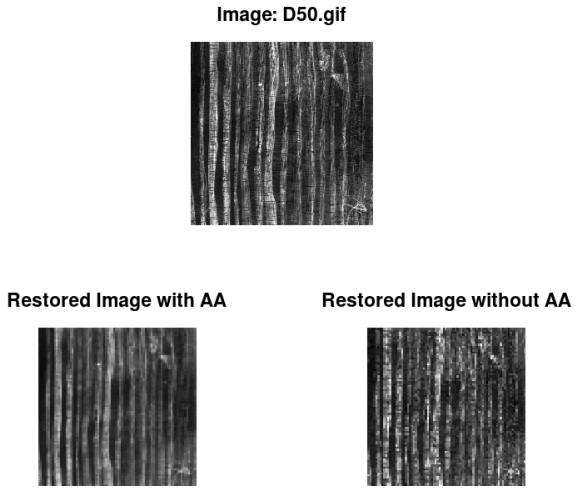

Figure 74: Image 11

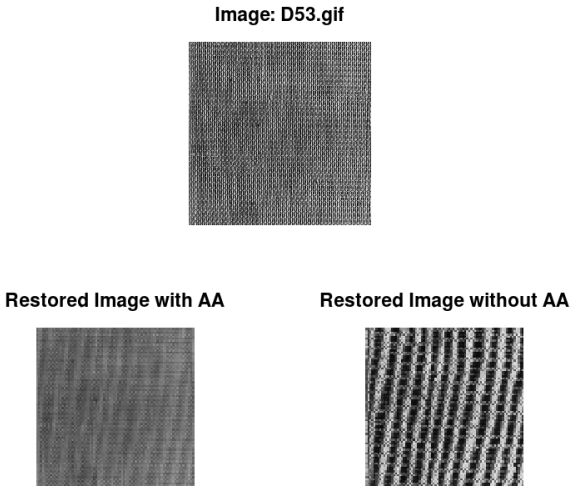

Figure 75: Image 12

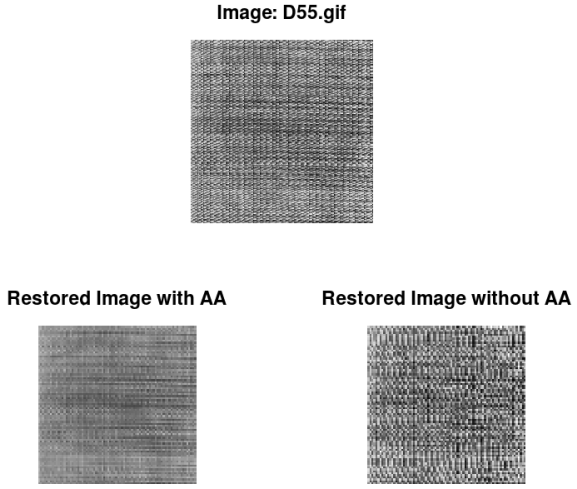

Figure 76: Image 13

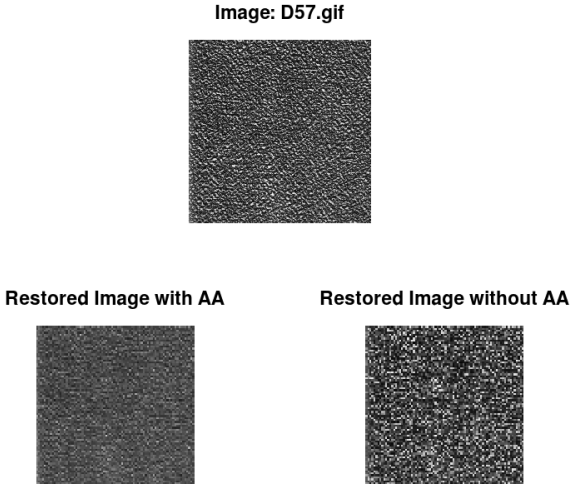

Figure 77: Image 14

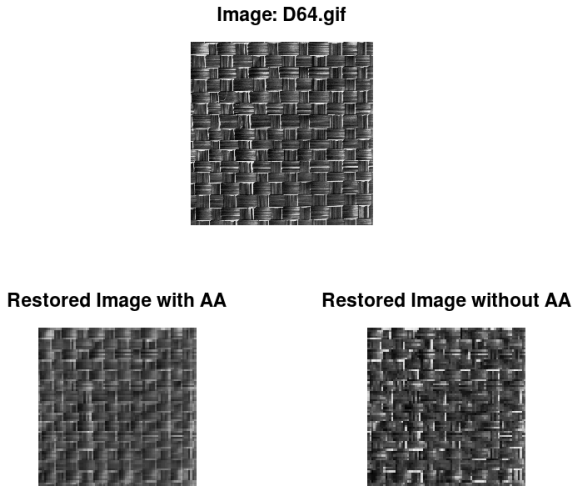

Figure 78: Image 15

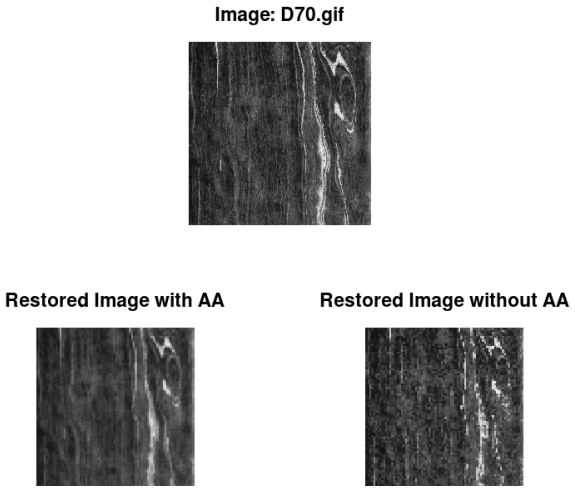

Figure 79: Image 16

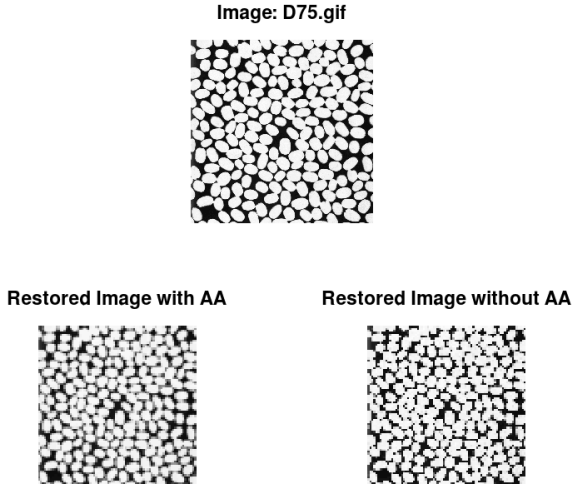

Figure 80: Image 17

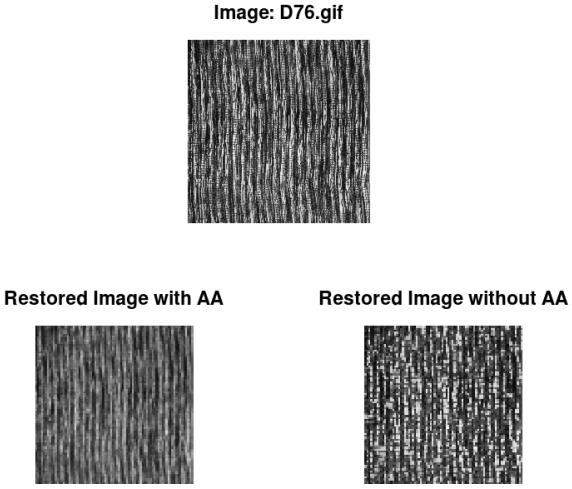

Figure 81: Image 18

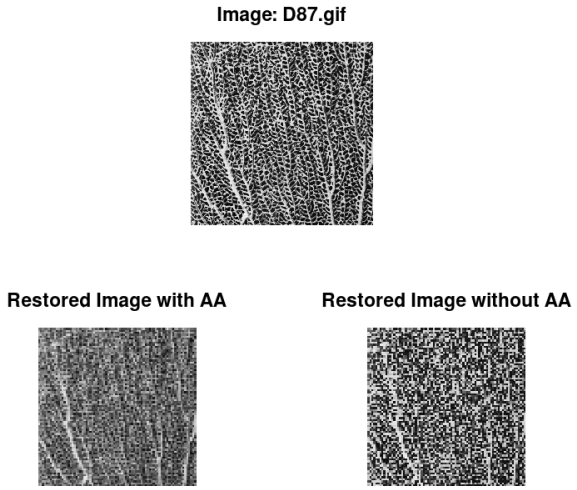

Figure 82: Image 19

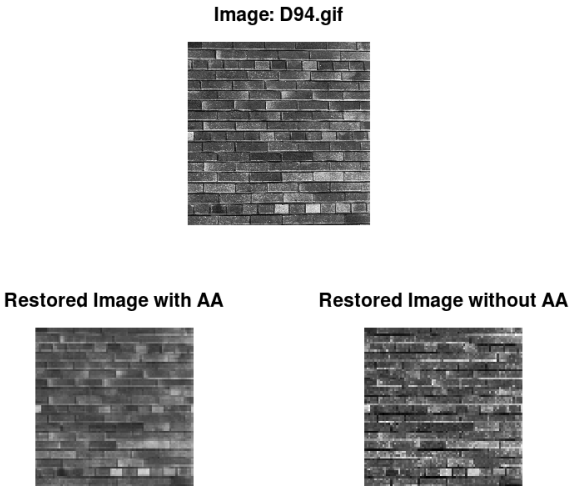

Figure 83: Image 20

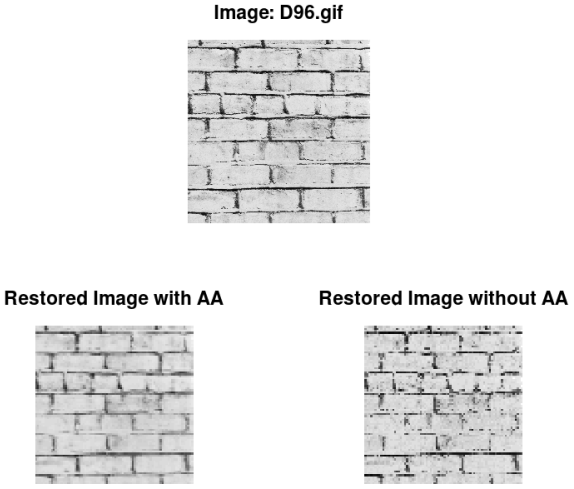

Figure 84: Image 21

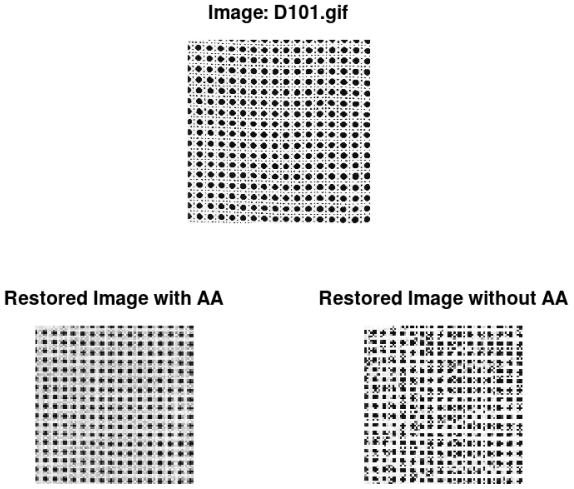

Figure 85: Image 22

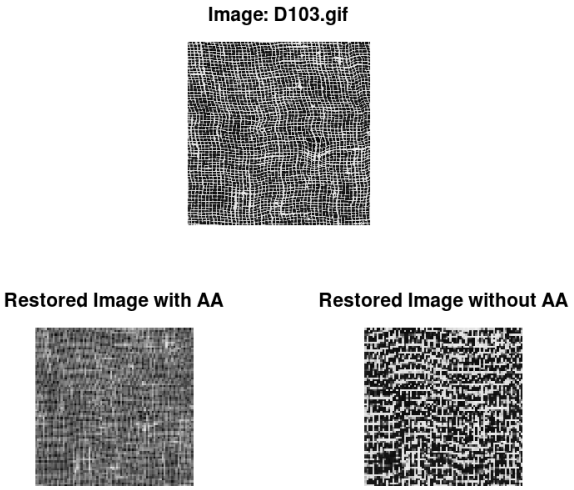

Figure 86: Image 23

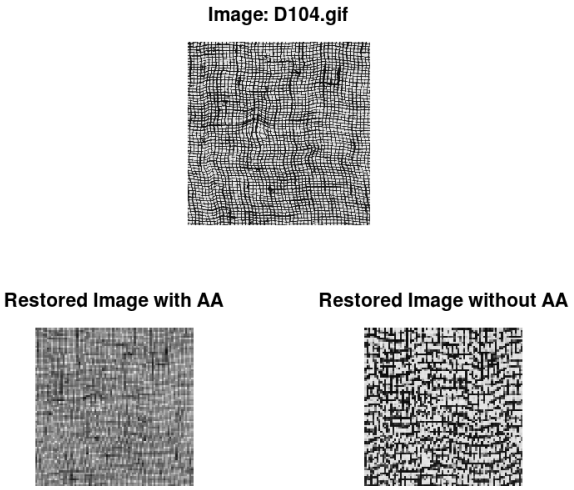

Figure 87: Image 24