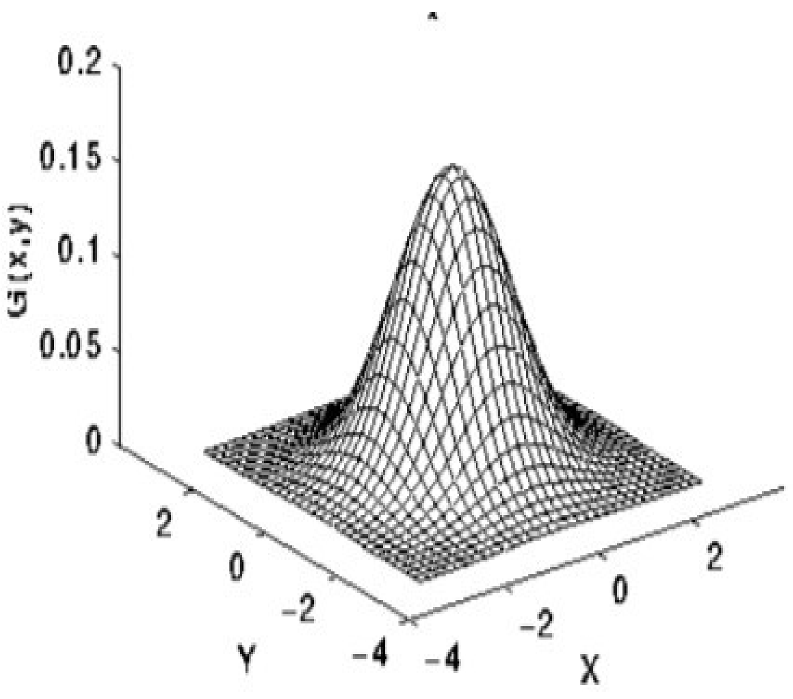
1. Image Smoothing
Spatial filtering can basically be examined under 2 headings. These are low pass filter and high pass filter. Low pass filters are generally used for smoothing. High pass filters are used for edge detection and sharpening. With image smoothing, the difference between neighboring pixels is reduced and the contrast of the picture is reduced. For this reason, it is often preferred for noise reduction. Since image smoothing is done using low pass filters, high frequency components are removed from the image and only low frequency components remain. Therefore, some details in the picture are lost. For example, sudden pixel change is used for edge detection. Since the differences between pixels will be reduced with smoothing, both the overall picture becomes blurred and details such as edges begin to disappear.
Parameters such as filter size directly affect this change.

2. Mean Filter (Averaging Filter)(Box filter)
It is the most basic spatial smoothing filter. It is done by replacing the average of neighboring pixels of the relevant pixel and filter size with the corresponding pixel value. It is generally used to reduce noise in the picture. Since it is a smoothing filter, it is a low pass filter and reduces the contrast in the picture, the difference between neighboring pixels is reduced by averaging. If the neighbor size, that is, the filter size, increases, smoothing and blurring increase. It is implemented by moving the filter over the image with the Convolution process. It is generally the most unsuccessful of other smoothing filters. I will explain these problems on sample pictures.

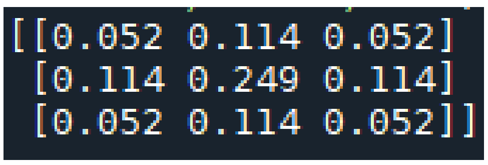
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

3. Gaussian Filter
It is another smoothing filter that removes high frequency components from the picture and reduces noise. It is a spatial convolution filter, which is achieved by moving the filter over the picture. The most important feature is that it is a separable kernel. A 2D gaussian filter can be obtained by applying two separate 1D gaussian filters. In terms of complexity, applying two 1D filters works better than applying a single 2D filter with multiplications and doing the same job. Its filter is not fixed like the mean filter and is calculated using both the filter size and the sigma value. Another feature is convolution with self is another Gaussian. So, applying a single filter with convolving two times with sigma and sigma*sqrt(2) gives the same result. We said that his filter is not fixed. The visual and formula for the calculation are as follows. Here, sigma acts as the standard deviation of the distribution and the mean of the distribution is assumed to be 0.

$$G(x) = \frac{1}{\sqrt{2\pi}\,\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



An example sigma value and its calculation for the filter are as follows (filter size = 3).

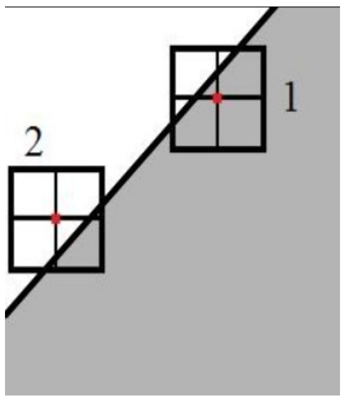


```
[[0.052 0.114 0.052]
 [0.114 0.249 0.114]
 [0.052 0.114 0.052]]
```

4.Kuwahara Filter

It is another smoothing filter. Implementation is different here, when using convolution in mean and gaussian filters by convolution over a kernel image. It is a non linear filter. **Most of the filters used for image smoothing reduce noise while blurring the picture and making details such as edges disappear. Kuwahara smoothes the image while protecting the edges**. This is the main difference from other smoothing filters. It uses standard deviation and mean while doing this. It divides our filter matrix into four quadrants and finds the standard deviation of each quadrant. The sum of the means of the quarters with the smallest standard deviation becomes the new pixel value for us. **Smaller filter values are more effective at preserving edges, although large filter values usually give more abstract output. The answer to the question of how it protects the edges can be explained as follows.**

In a situation where one side of the image is dark or bright, the average value of that portion will likely be taken, since the standard deviation of that portion will likely be lower. In this way, the homogeneity of the region is taken into account and the edges are preserved.



For example, in the picture on the right, filter number 1 will use the average of that quadrant, since the standard deviation of the completely dark part is lower, filter number 2 will use the completely bright quadrant. **In this way, the central pixel will take the average of the most homogeneous (with the lowest standard deviation) region, so that the edges will be preserved even if the picture is blurred. (The edges are the spots where there are sudden pixel changes, just like in the example picture)**

5. Implementation Details

5.1. Mean and Gaussian

Since the Mean and Gaussian functions are basically based on convolution and a kernel navigating over the picture, I have handled both of them in a function called blurring.
The function signature is blurring (image,filterSize,blurringType,sigma=0).
Blurring type decides which spatial filter to use with "mean" or "gaussian". Also, I have assigned a default value since sigma will only use the gaussian value. We don't need the sigma value for the mean filter. Example function calls are as follows.

```
blurring(image, 3, "gaussian",0.8)
blurring(image, 3, "mean")
```

Since we will be working on colored pictures in the function, we had to work with 3 different values, RGB. I extracted the R, G and B values of all pixels in Images and brought them to the image size. Thus, I have 3 images with the same shape and a matrix for you. Contains only R values, only G values, and only B values. Although the kernel was stable for the mean filter, it was necessary to create a kernel with sigma for gaussian. I created a kernel according to the sigma value and filter size given with the function named **calcualteGaussianFilter**. What I did next was to go around the filter with convolution for the R, G and B matrices and calculate the new pixel values for each channel. Then I created a smoothing image by taking the new pixel values from the matrices. Because I didn't do padding, the images shrank a few pixels relative to the filter size.

5.2. Kuwahara

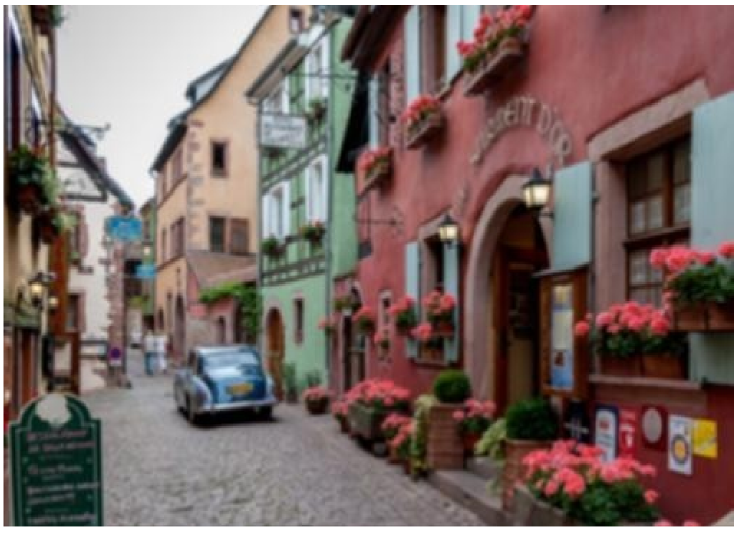
Since we are working on the colored image in the Kuwahara filter just like the previous ones, I removed the R, G, B matrices of the picture. However, after I made the HSV transformation of the picture, I also extracted the V matrix. After calculating the standard deviation over the V value, I found that averaging the RGB pixels of the minimum quadrant worked better. That's why I calculated the standard deviations of the quadrants with their V values and decided on the min quadrants. Then I calculated the average values of the individual quadrants for the R, G, B channels, and found the new pixel values. I created the Smoothing image using these new R,G,B pixel values. Because I didn't do padding, the images shrank a few pixels relative to the filter size. I used **calculateMeanAndStandardDeviation** function for mean and standard deviation operations in Kuwahara, and navigating the filter with convolution was done with my main function, kuwaharaFilter. **I did my math by calling calculateMeanAndStandardDeviation function in kuwaharaFilter**. Example function calls for related functions are as follows

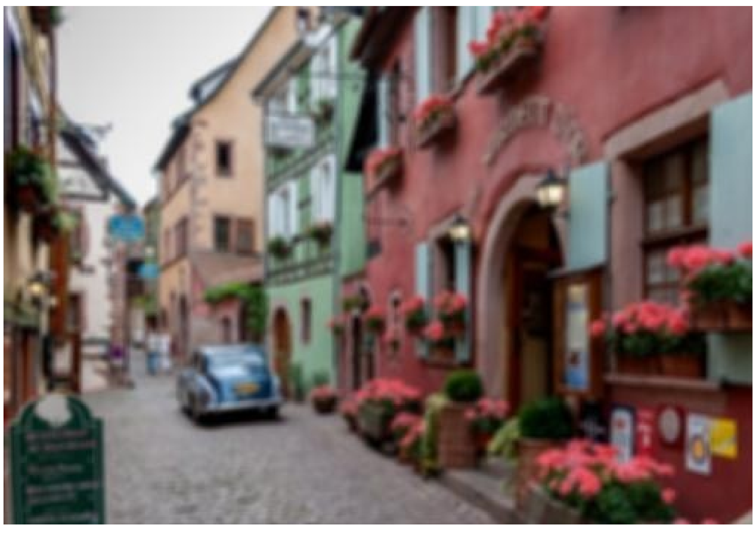
kuwaharaFilter(image,9)
calculateMeanAndStandardDeviation(value,filterSize,colorChannel)

With the calculateMeanAndStandartDeviation function, I calculate my quadrant standard deviation values in an area equal to the filter size in my V matrix. With the input color channel (R, G or B), I took the mean/sum of the quadrant with the minimum standard deviation in a single channel and calculated my new pixel value in the relevant channel.

6.Results of Algorithms

6.1.Mean (original image,from 3x3 to 9x9 kernel size)





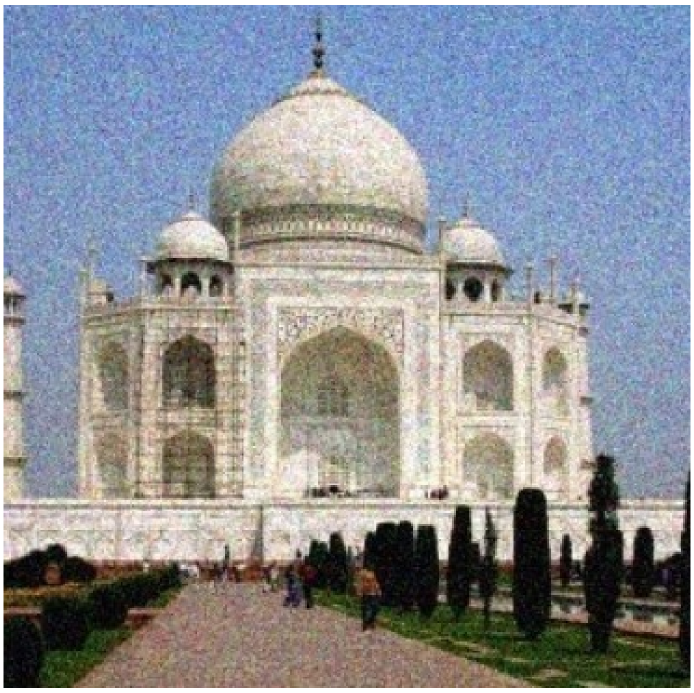
93-0658

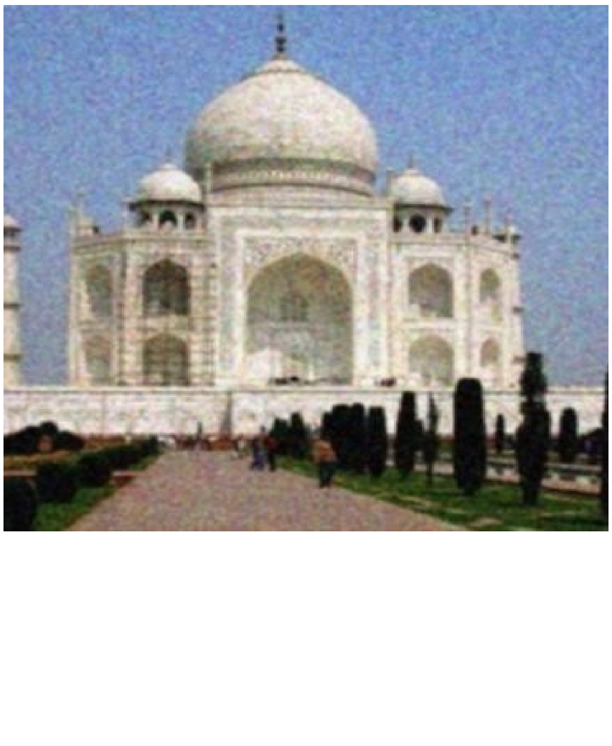
5302
5302
BNSF
BNSF
DASH 9-44CW

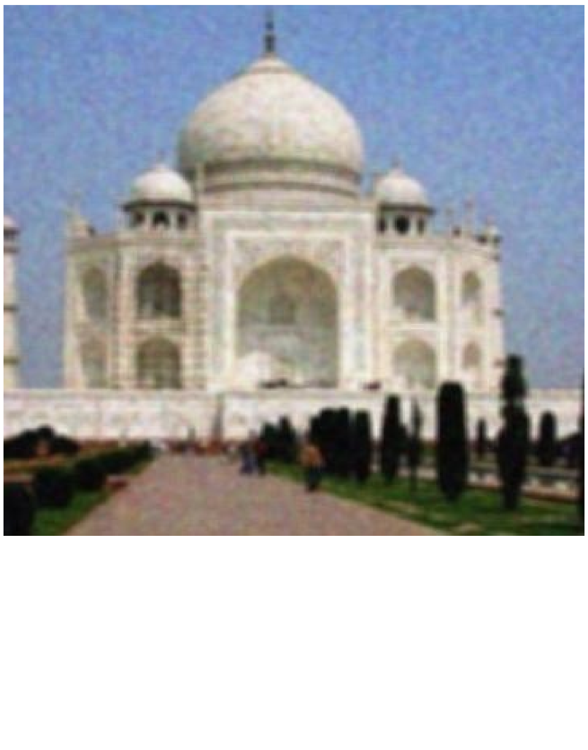
5302

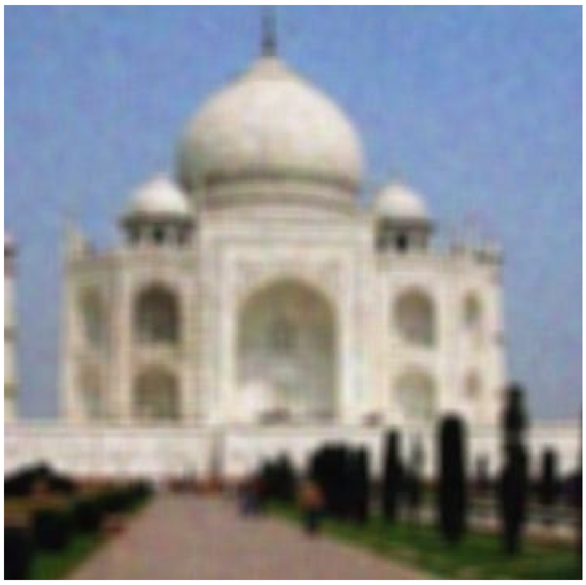
5302

5302









As you can see from the images, filter size enlargement makes the picture more blurry and smoother. Calculating a larger range of neighboring pixels while calculating a pixel causes more detail to be lost. However, we saw in the last picture that it was successful in reducing the noise in the picture. However, wider filter values naturally reduce more noise. **During smoothing, since the farthest and nearest neighbor pixels affect the relevant pixel at the same rate, the picture loses its features (especially the edges).**

6.2. Gaussian (original image,from 3x3 to 9x9 kernel size with different sigma)

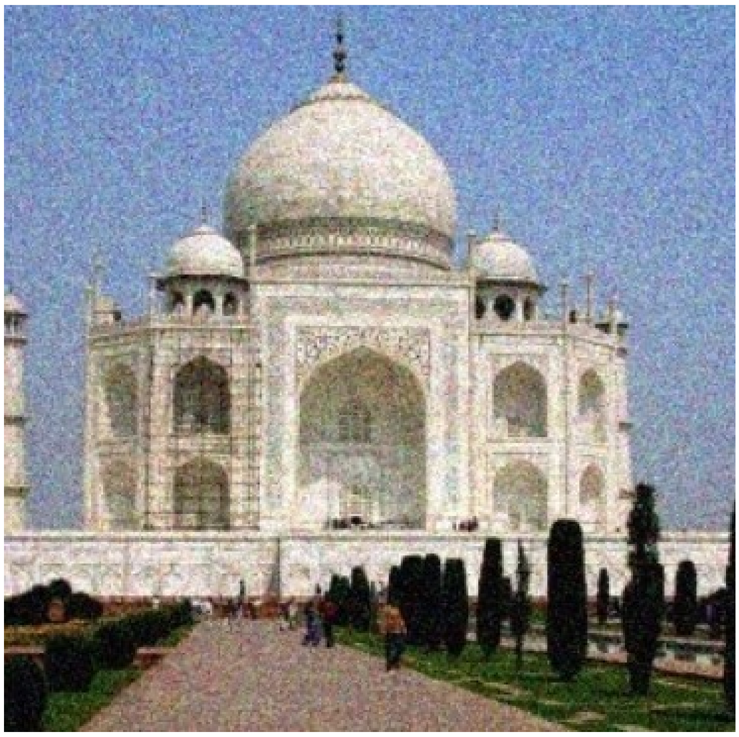
sigma = 0.5

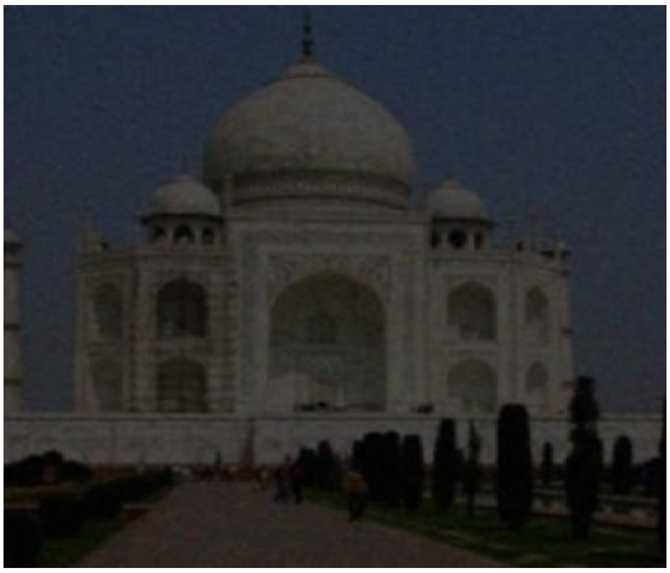
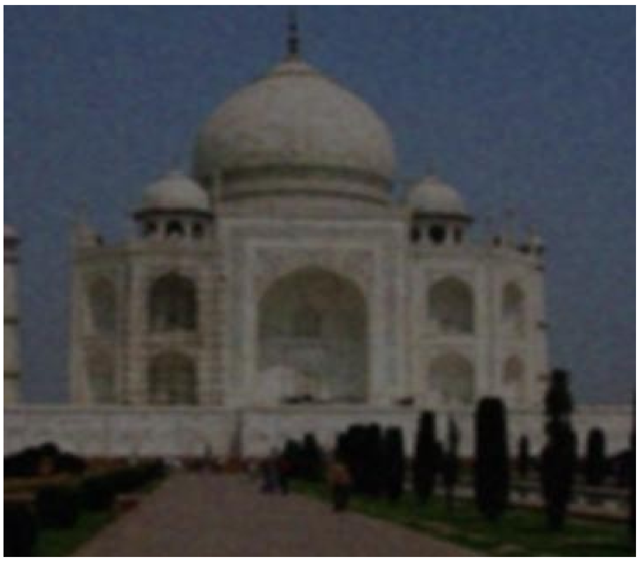
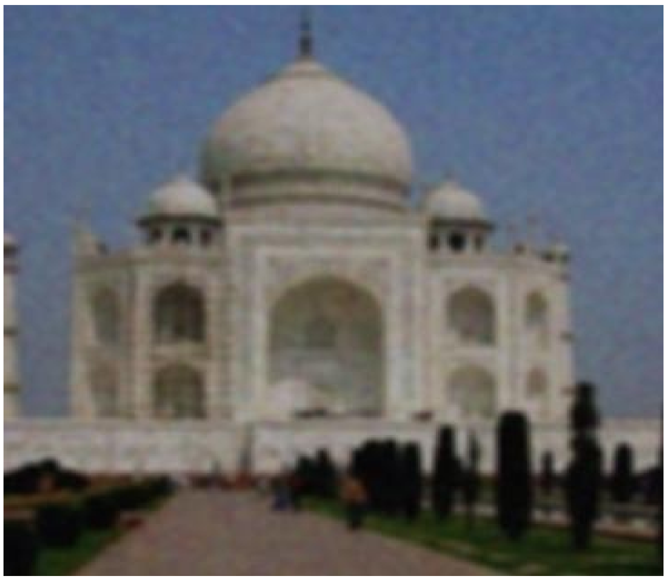
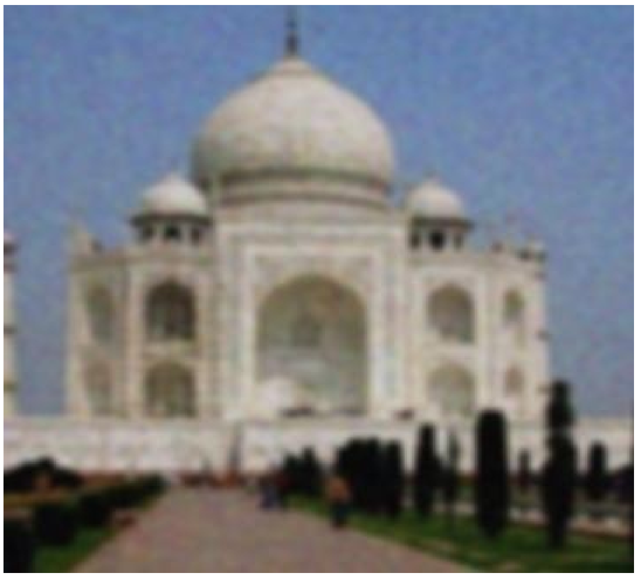
sigma = 0.8

sigma = 1.0

sigma = 2

sigma = 2.5



It showed a good result in removing noises and preserving the overall structure of the picture. The fact that the neighbors at different distances have different weights when processing with the relevant pixel ensured that the picture preserves its general structure even if it becomes blurred. However, at low sigma values, there was no noticeable difference in the outputs, even if the filter size was changed. At higher sigma values, the picture started to get darker. However, with increasing the filter size, the picture started to get a little bright and blurred even more. I think the darker working of the picture at higher sigma values has to do with the generated matrices. Let's try to explain this situation with a few kernels below.

Sigma = 2 3x3

```
[[0.031 0.035 0.031]
 [0.035 0.04  0.035]
 [0.031 0.035 0.031]]
```

Sigma = 0.5 3x3

```
[[0.012 0.086 0.012]
 [0.086 0.637 0.086]
 [0.012 0.086 0.012]]
```

Sigma = 2 9x9

```
[[0.001 0.002 0.003 0.005 0.005 0.005 0.003 0.002 0.001]
 [0.002 0.004 0.008 0.011 0.013 0.011 0.008 0.004 0.002]
 [0.003 0.008 0.015 0.021 0.024 0.021 0.015 0.008 0.003]
 [0.005 0.011 0.021 0.031 0.035 0.031 0.021 0.011 0.005]
 [0.005 0.013 0.024 0.035 0.04  0.035 0.024 0.013 0.005]
 [0.005 0.011 0.021 0.031 0.035 0.031 0.021 0.011 0.005]
 [0.003 0.008 0.015 0.021 0.024 0.021 0.015 0.008 0.003]
 [0.002 0.004 0.008 0.011 0.013 0.011 0.008 0.004 0.002]
 [0.001 0.002 0.003 0.005 0.005 0.005 0.003 0.002 0.001]]
```

As can be seen above, for the same filter size (sigma=2 and sigma=0.5 3x3) low sigma value is less affected by distant pixels that are attached with a higher weight to the relevant pixel. The high sigma value approximates almost all pixels with equal weight. This allows us to obtain a more dark image. With a high sigma value, we lose more detail. However, to solve this, we can use a larger filter size containing the same sigma value (sigma=2, 3x3 and 9x9). In this way, high weights still accumulate near the relevant pixel. It keeps the picture from getting dark and losing more detail.

6.3. Kuwahara (original image,from 3x3 to 9x9 kernel size)

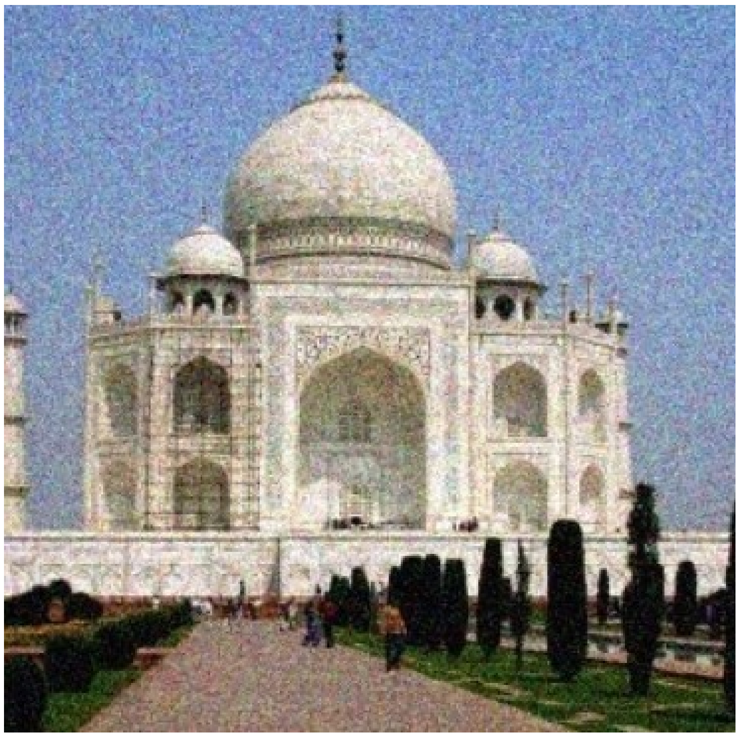
93-0658

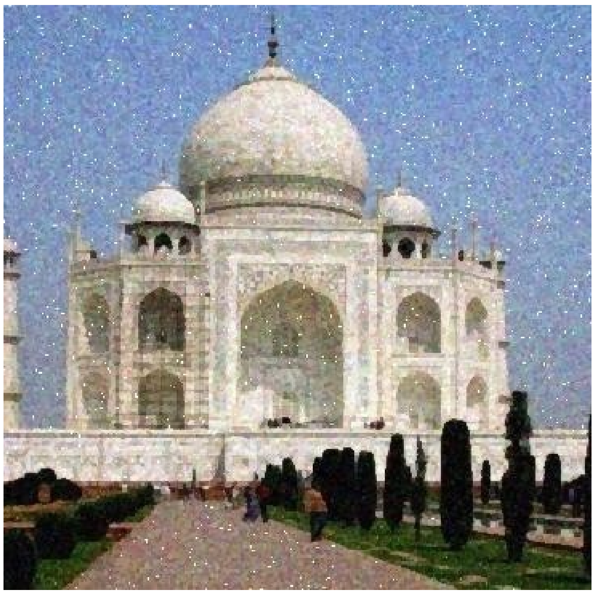
5302
5302
BNSF

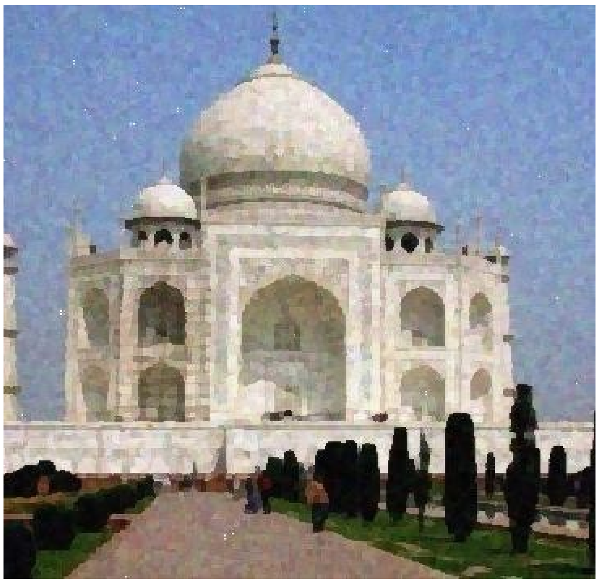
5302

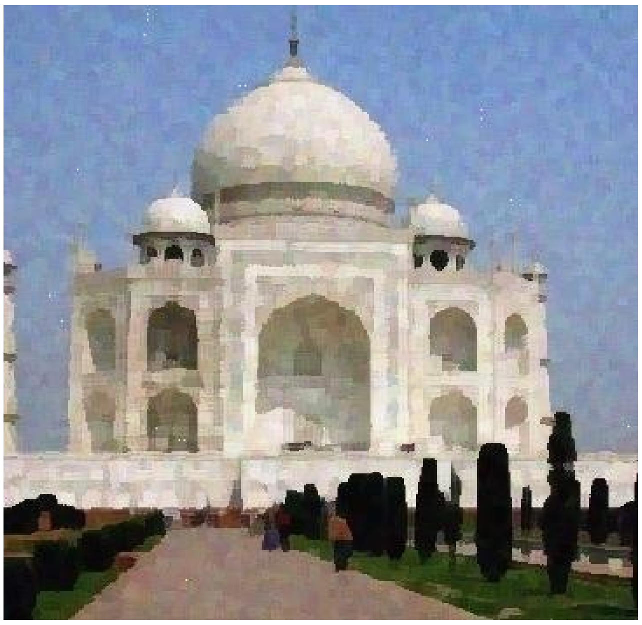
5302

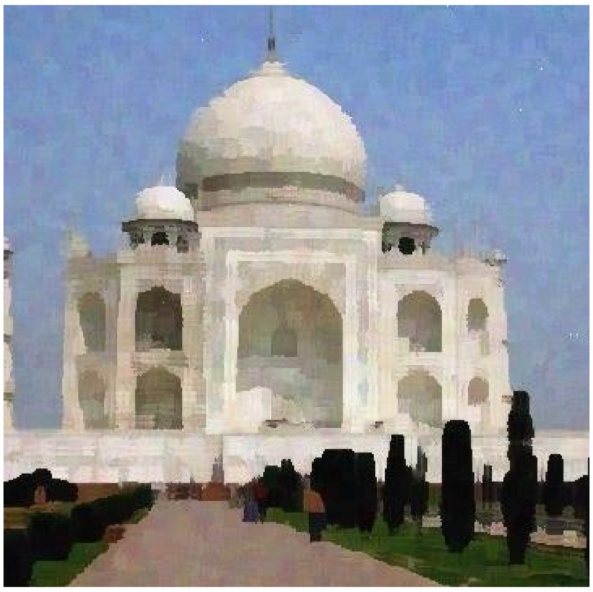
5302











As the filter size grows, the smoothing increases. However, noise occurs in 3x3 filters. Thanks to the standard deviation and mean it uses, homogeneous regions in the picture are preserved and the edges are preserved. However, the picture becomes blurry. The most successful is the protection of the edges. For high filter values, the outputs become abstract and unrealistic.

7. Difference Between Filters

The Kuwahara filter is the most successful filter in preserving the edges and the general structure of the picture and the edges. It provides this by dividing the picture into quadrants with mean and standard deviation. However, Gaussian is also quite successful in this regard. The fact that the neighbors near and far from the relevant pixel have different weights preserve the general structure of the picture. The mean filter fails quite badly in this regard. Especially, as the filter gets larger, the features and details of the picture are lost, since even the neighbors far away from the relevant pixel affect the same weight as the close neighbors.

The Kuwahara filter runs much slower than the mean and gaussian filters as it involves more mathematical operations. Each filter is divided into four and extra operations are performed for each quadrant. This causes it to run slowly.

The Gaussian filter causes a lot of information loss for high sigma values and especially the low filter causes the picture to be darker for you. However, the low sigma value prevents the differences between filter sizes from affecting the output. 3x3 and 9x9 filters give very close results for low sigma values.

All of them are successful in removing noise, where smoothing filters are for general use. Kuwahara protects the edges the most while doing this. We see better noise removing results as the filter size increases.

The reason why the mean filter is the most unsuccessful is that even a very distant neighbor or a value far from the distribution in the filter has the same weight as the other values.

8. Failed Algorithm

The working time of the Kuwahara filter is longer than the other two, due to the large number of processes performed. In Kuwahara, where a filter is convolutioned in the other two, each convolution also performs the operations of dividing the filter into four, mean and standard deviation.

Outputs for 3x3 in Kuwahara filter contain noise. Containing noise is a situation that can be encountered as a result of filters in general. Because the filter size is small, the quadrants get even smaller. Thus, even small areas of the same color affect the result as their standard deviation will be min. As the filter size gets larger, there is not much noise as it is not possible to encounter single color areas of that width.