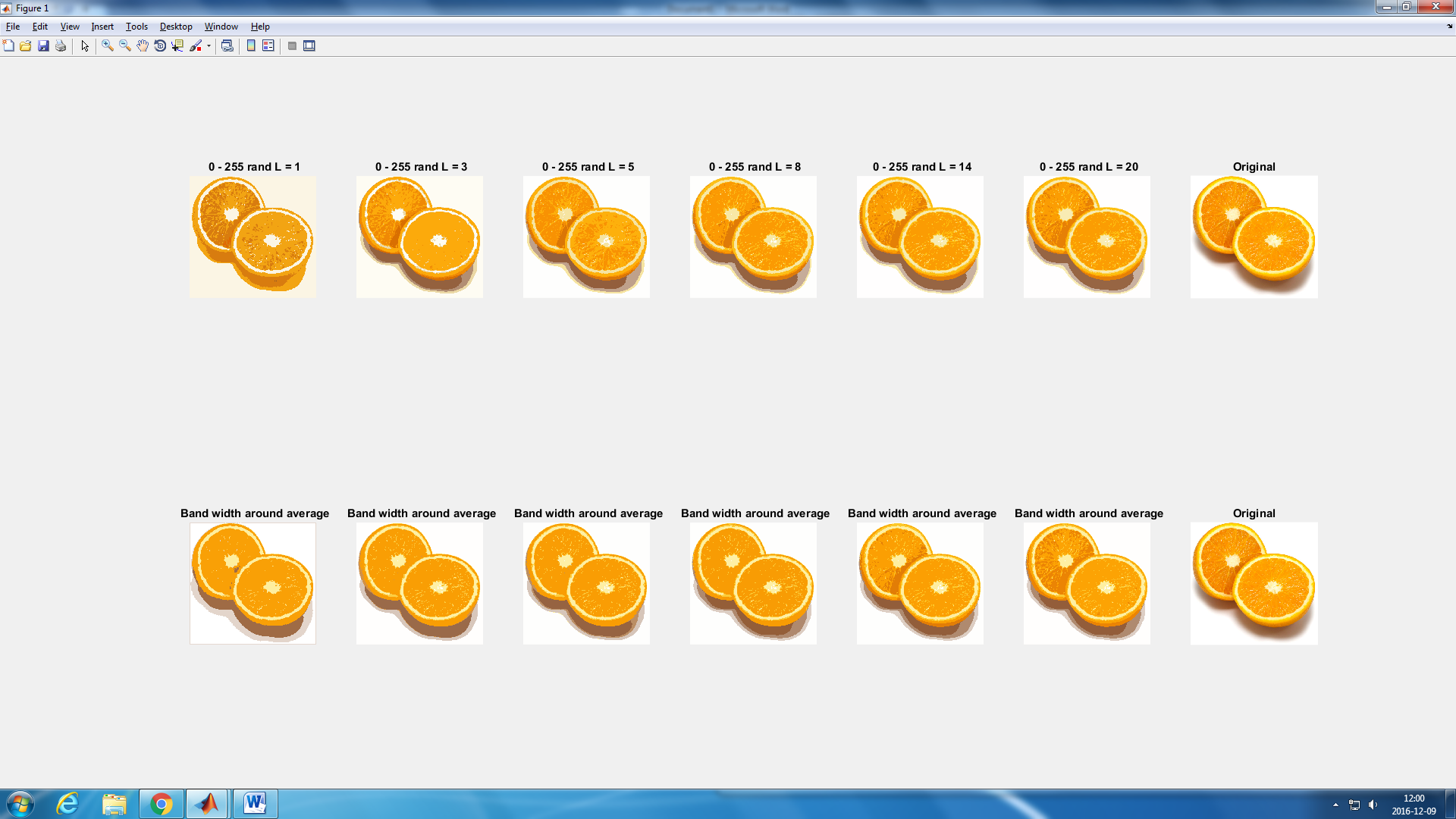
K means clustering

**Question 1: How did you initialize the clustering process and why do you believe this was a good method of doing it?**

The initialization of the clustering process can be done using different methods. In the internet is easy to find several algorithms only to initialize the clustering process. However, in the lecture notes were specified that this initialization had to be random. This type of initialization of the clustering process consists of giving random values to the center of the clusters. However, these random values can be given between different ranges of numbers. For the initialization of the random values I proposed 3 different methods. In the first and the simplest one I just generated random numbers between 0 – 1 and multiply them by the max value possible for a pixel (255). Then, I generated this way random numbers between 0 and 255. The second option was to get the max value of a pixel of the image and use it to multiply the random (0,1) number. This way I generated numbers between 0 and the max value of pixels of the image. The last method was to obtain the mean value of the pixels of the image, and generate the random numbers around this mean value within a bandwidth. So I can generate this way random values around mean\_value +- bandwith.

The first method was the most obvious one; we just initialize the center of the clusters to a random 8 bit color. The second method came up as an alternative. The idea was to initialize the values around the mean values of the image so the convergence could be reached faster. However, I designed several experiments to check if this was true.

First if just compared the outputs of the two methods using different number of iterations. (Figure 1)



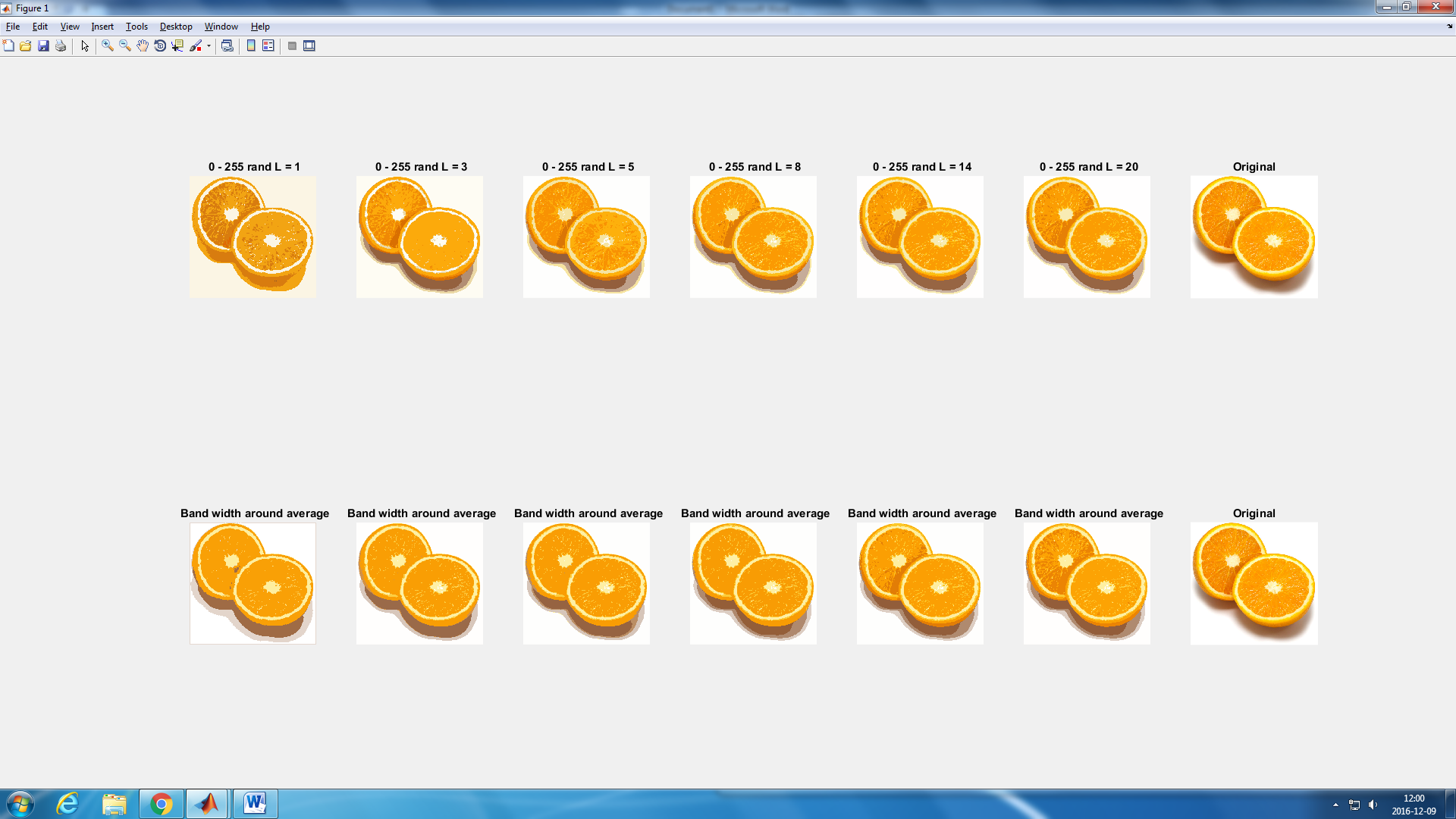


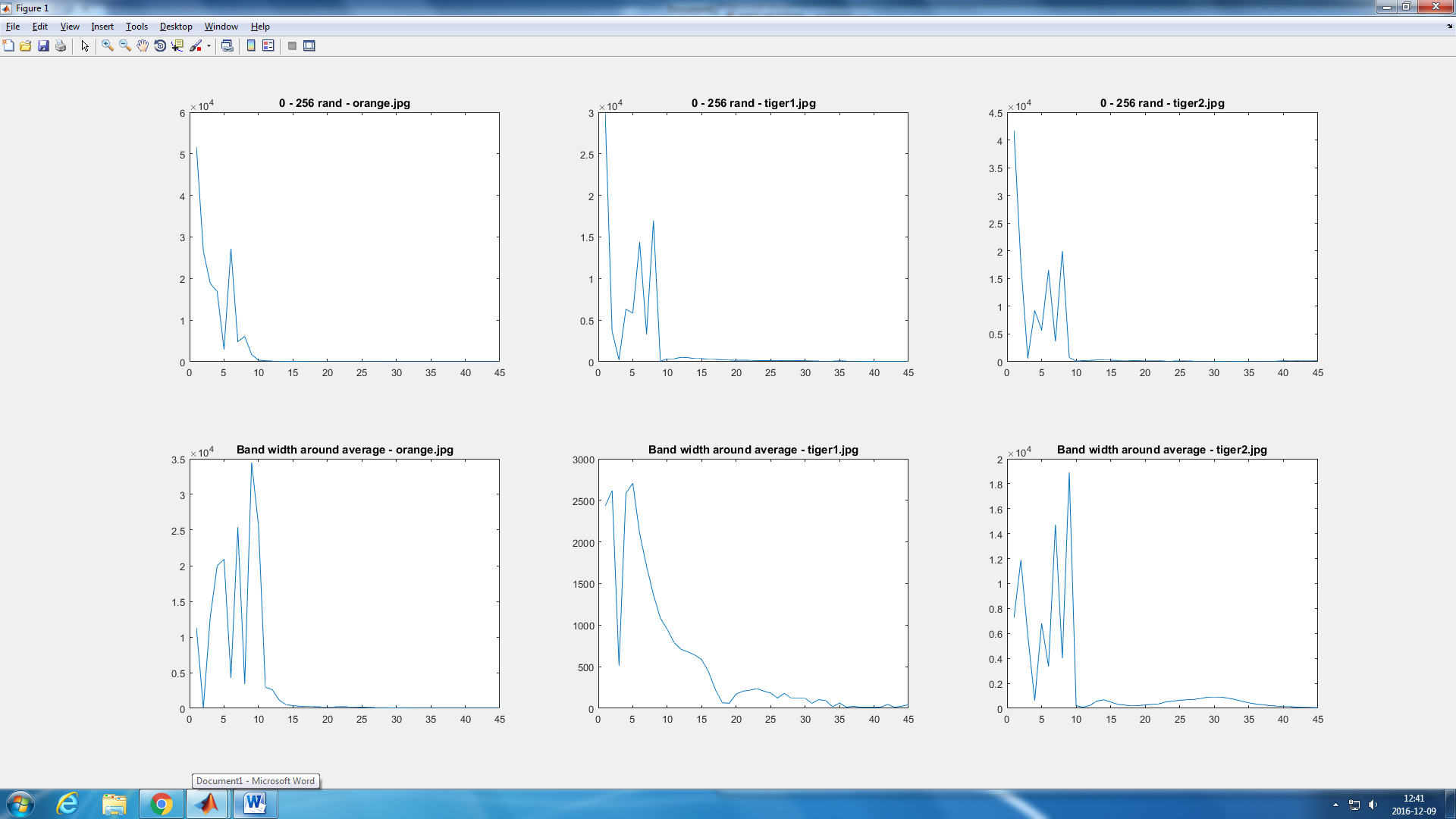
Figure 1: Output comparison between the two initialization methods

We can easily observe that using the first method, it needs less iteration to reach a picture that looks more alike to the original one. However, for highest number of iterations the differences between methods are not huge.

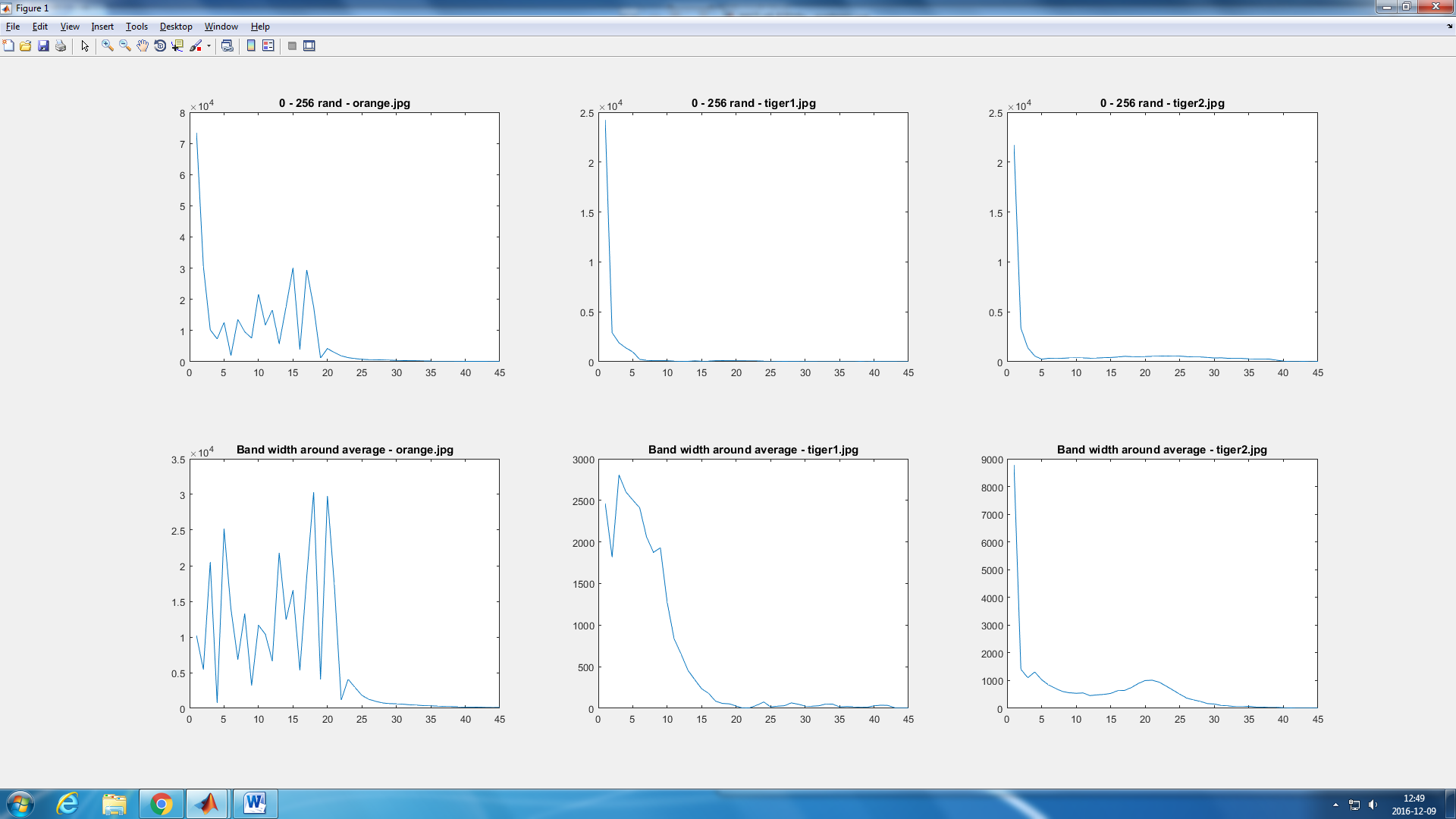
Let’s now study the convergence of both methods. For this, we are going to see the evolution of the difference between the distances between the current iteration and the previous one. This way we can measure how much does the new iteration output looks alike to the previous one. The convergence would be reached when in both the previous and the current iteration the obtained image is exactly the same.

Plotting the evolution of these differences for both methods in different images( Figure 2) we can observe several facts. Notice that we used different seeds for the random initialization to make sure that our conclusions are valid for different possible initial random sequences. First, we can conclude that the convergence is reached, or at least the difference is low enough, after 10 iterations. We can also see that in general, the differences decrease quicker in the first method and they completely converge (difference ≈ 0) using less iterations.

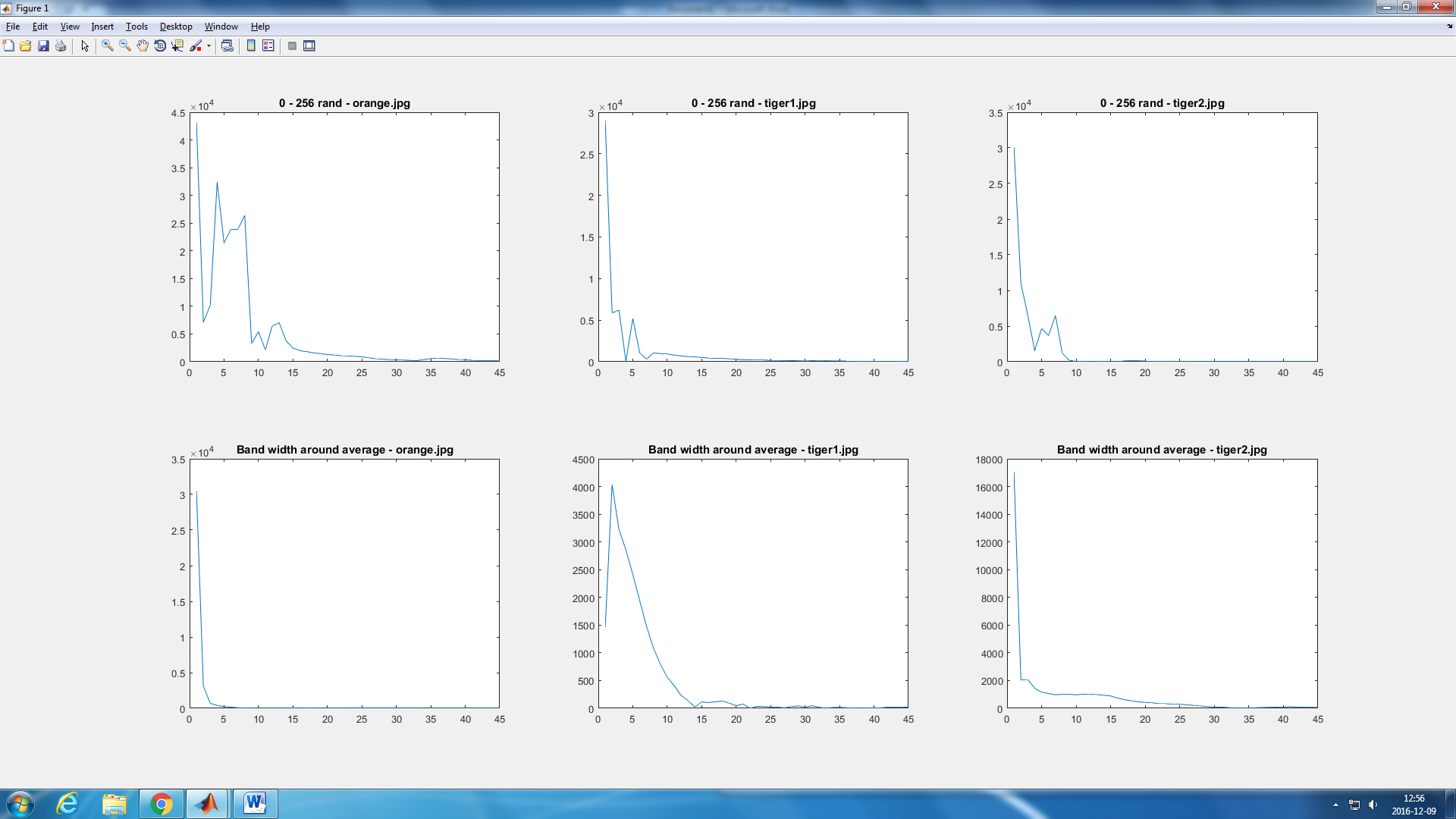
After these experiments we can conclude that the first method is more optimal than the bandwidth method. This makes sense as the range of possible colors is higher and therefore, the pixels will be more spreaded between the different clusters during the first iterations and therefore, the convergence will be reached quickly as the update of the clusters centers would be more effective.



(a)



(b)



(c)

Figure 2: Distances differences plot for 3 different images. 0-255 method (top) bandwidth method (bottom). Different seeds where used (a) 14, (b) , (c)

**Question 2: How many iterations L do you typically need to reach convergence, that is the point where no additional iterations will affect the end results?**

In the figure 2 we can see how the difference between the previous iteration output and the current one evolves during the iteration process. We can use this figures to conclude that, for both methods, the convergence is reached (or the difference is low enough) in around 10 iterations. This value can change depending on the image and the seed used, but almost in every case, it converges in about 10 iterations.

We can observe in the figure 3 that now differences can be identified for the first method.

**Question 3: What is the minimum value for K that you can use and still get no superpixel that covers parts from both halves of the orange?**

To see what is the minimum number of clusters that we need to get no superpixels that covers parts from both halves of the orange we used the best possible parameter of L and initialization method. As we said, the best combination of both would be to use the 0-255 initialization method and 10 iterations. With this parameters, we computed the algorithm for different numbers of clusters. We also printed th boundaries of each cluster (figure 4, in red) to help us figure out what the minimum needed number of K is.

Observing the figure 4 we can see that for K = 10, we don’t have any superpixel shared between both halves of the orange.

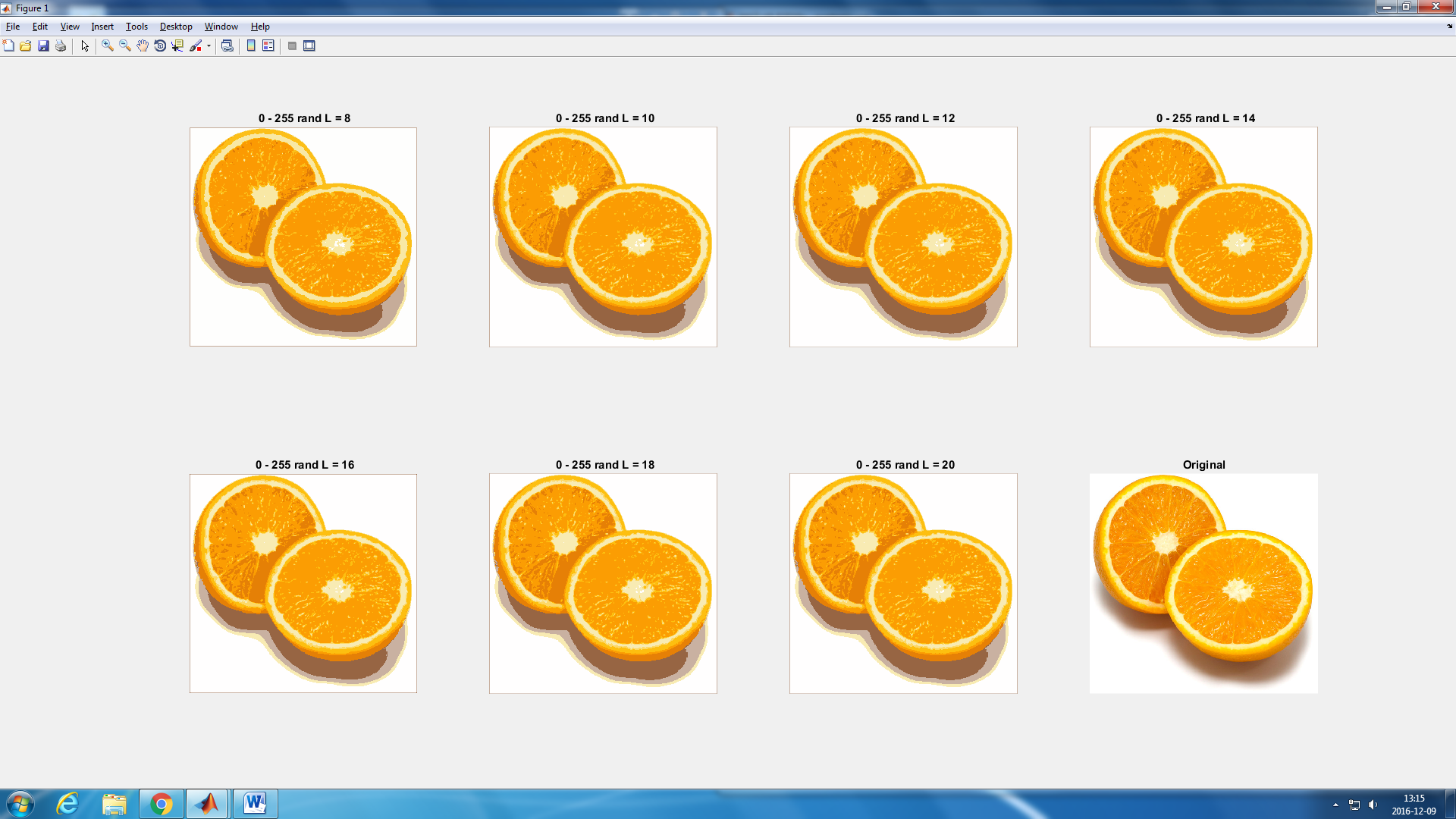


Figure 3: output for the 0-255 initialization method using different number of iterations

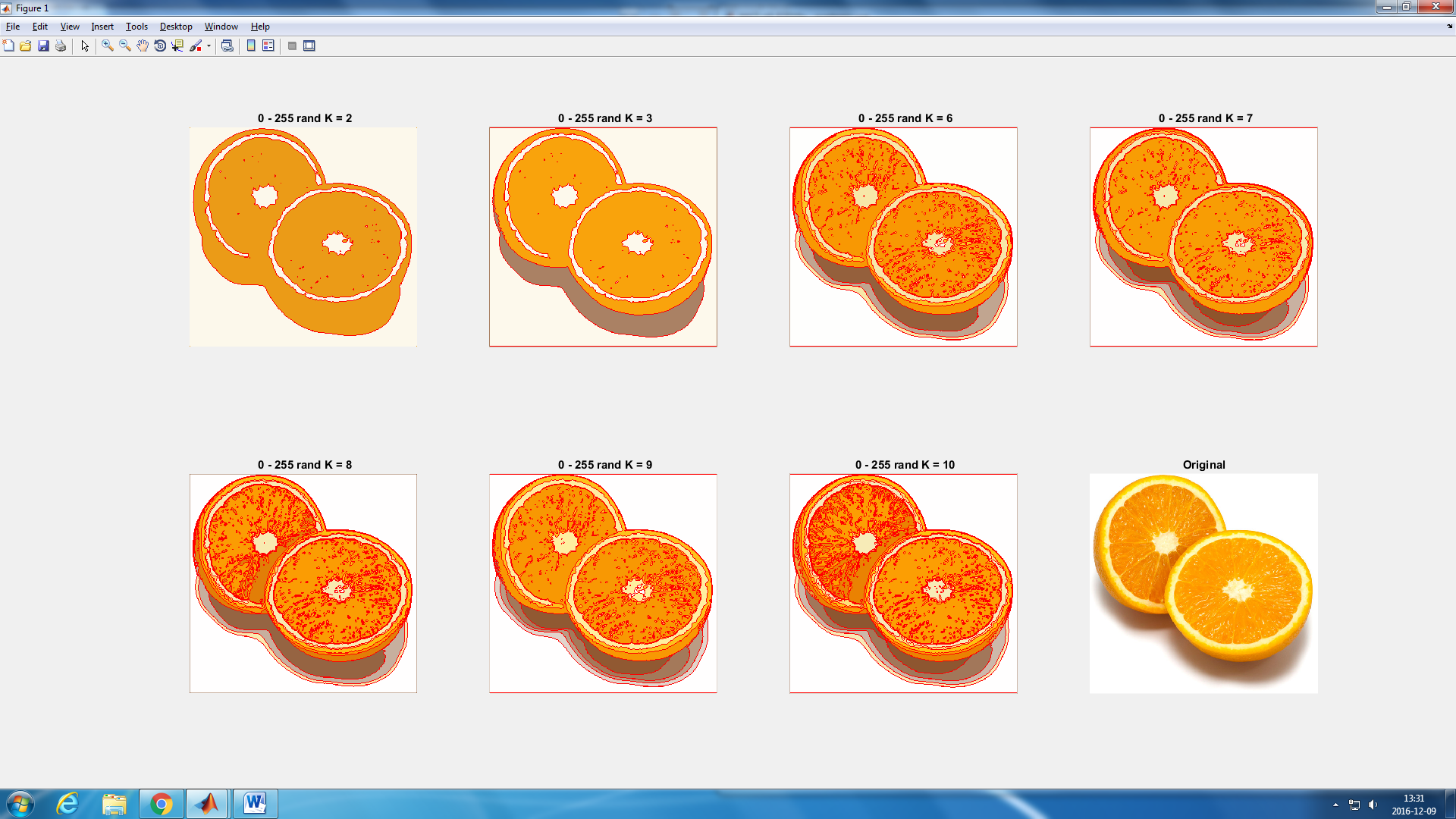


Figure 4: Output for different number of clusters using the 0-255 method and 10 iterations

Question 4: **Try using parameters suitable for orange.jpg and see how these affect the tiger images. What needs to be changed in the parameters to get suitable superpixels for the tiger images as well?**

For answering this question, we are going to set the number of iterations to 10 and we are using the first initialization method as we concluded that the combination of both is the best possible. We can play then with the number of clusters and the pre-smoothing of the image.

If we use the same parameters as in the orange example, we get the following image:

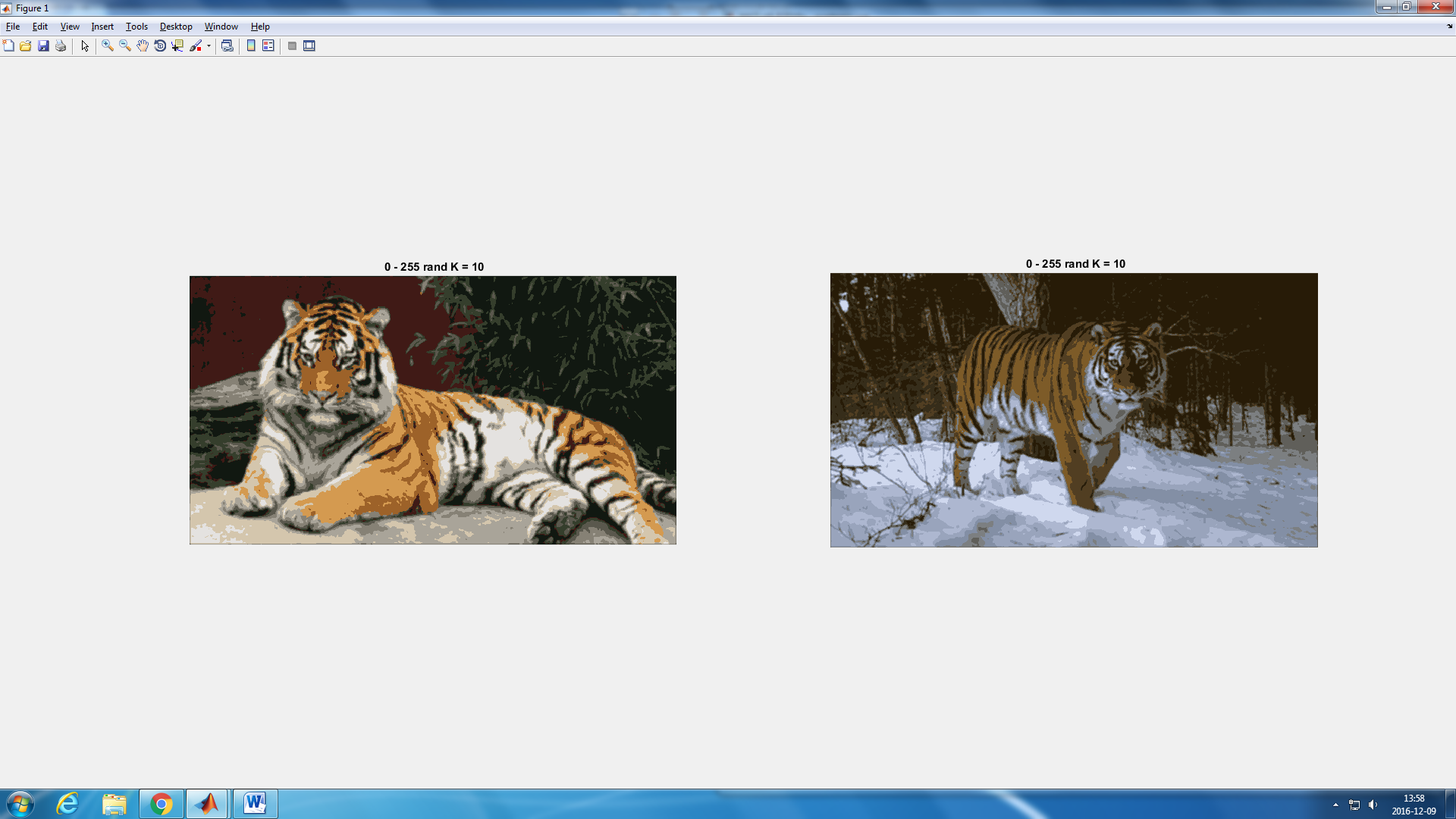


Figure 5: Output after applying the algorithm to tiger1.jpg and tiger2.jpg with the same parameters as we used for the orange .jpg example

We can observe that, while on the tiger1.jpg pic the parameters work well as we don’t have shared superpixels between the main objects in the image, on the tiger2.jpg image they don’t work as well. This is due to the low difference of intensity between the pixels of the tigers back and the background. We have then shared superpixels between the background and the tigers back.

Let study the image tiger1.jpg first. Applying the algorithm with the same parameters and only changing the number of clusters we get the outputs shown in the figure 6. We observe that for K = 7 ( 7 clusters) the tiger and the bush are easily don’t share any super pixel between each other or the background.

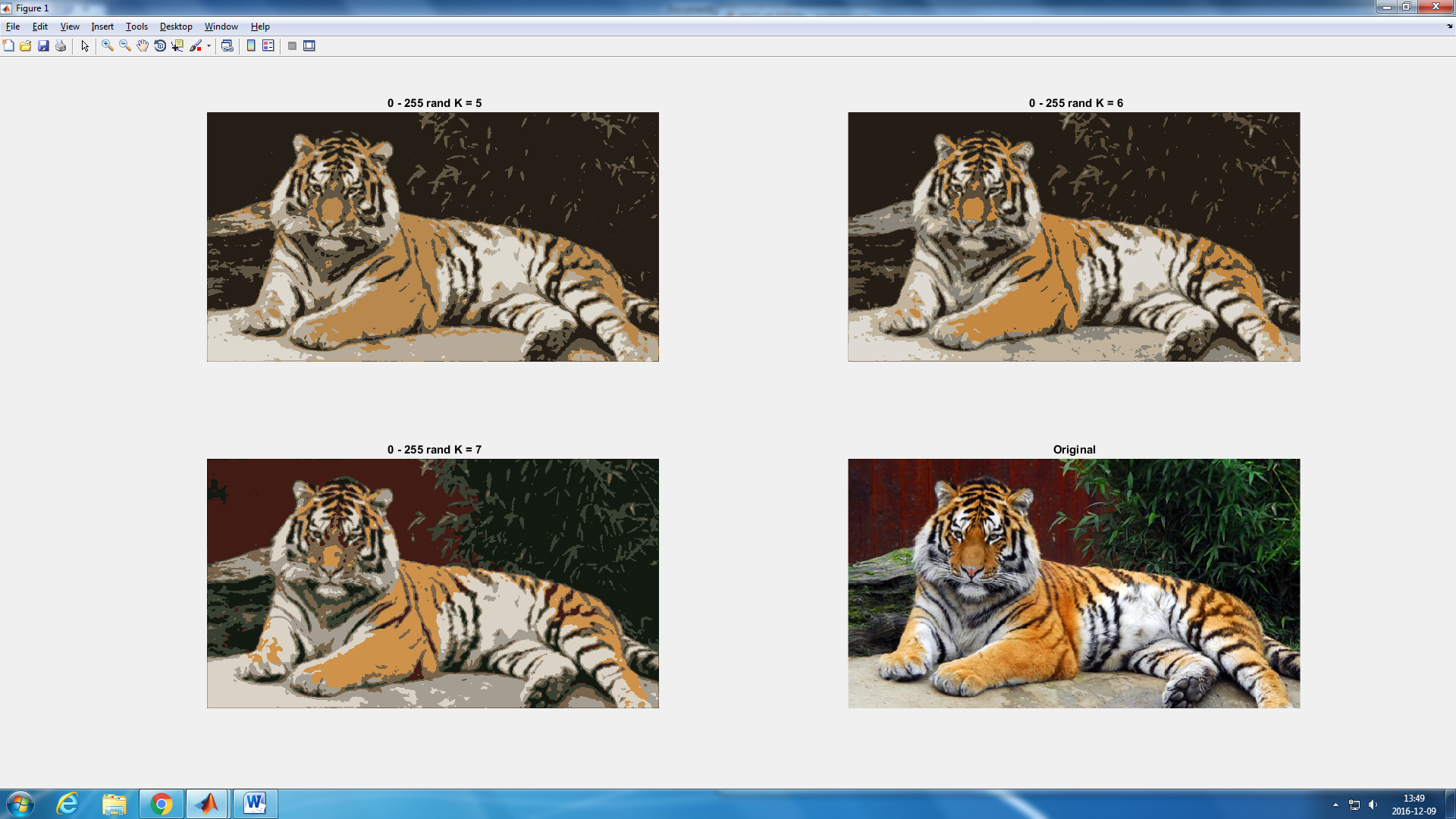
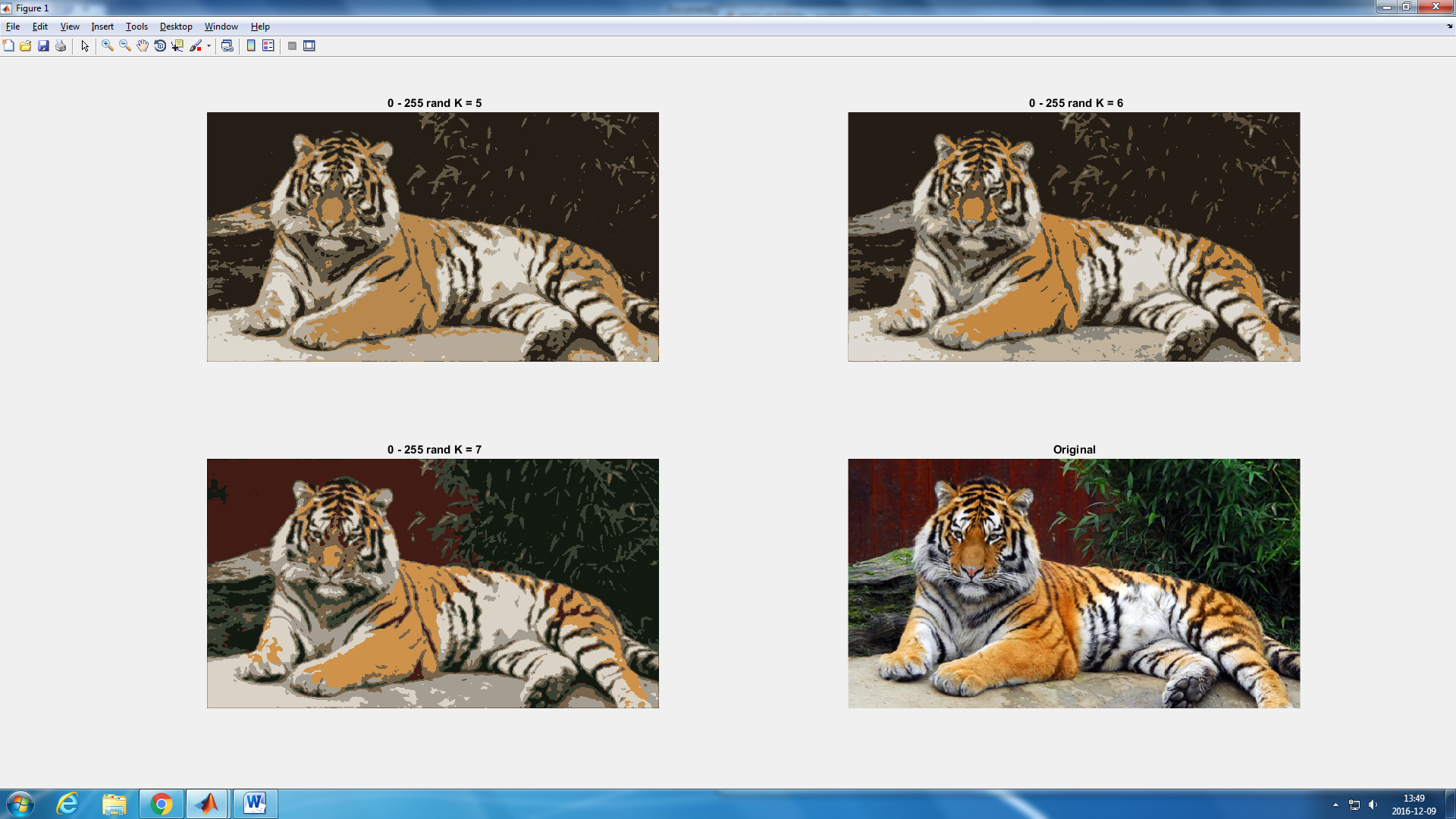


Figure 6: Output after applying the algorithm to tiger1.jpg with different values of K.

The parameters, however, are not working well on the image tiger2.jpg, as we have shared superpixels berween the tiger and the background. We can play with the value of two parameters to improve the performance of the algorithm: the smoothing and the number of clusters. Let see first how the smoothing affects the output. We are going to use the same set of parameters that we used for the orange image changing only the value of the sigma that we used to pre-filter the image.

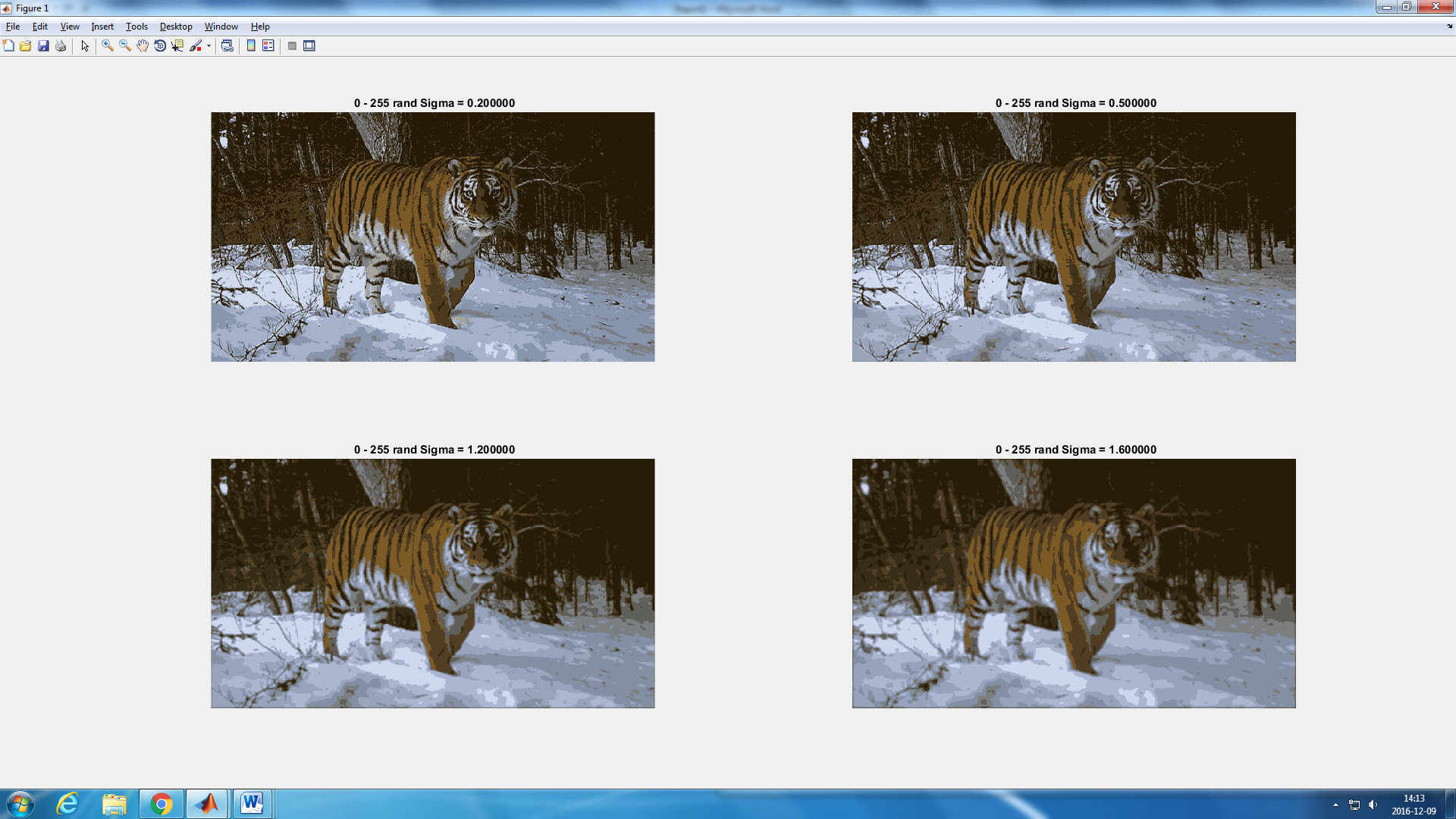
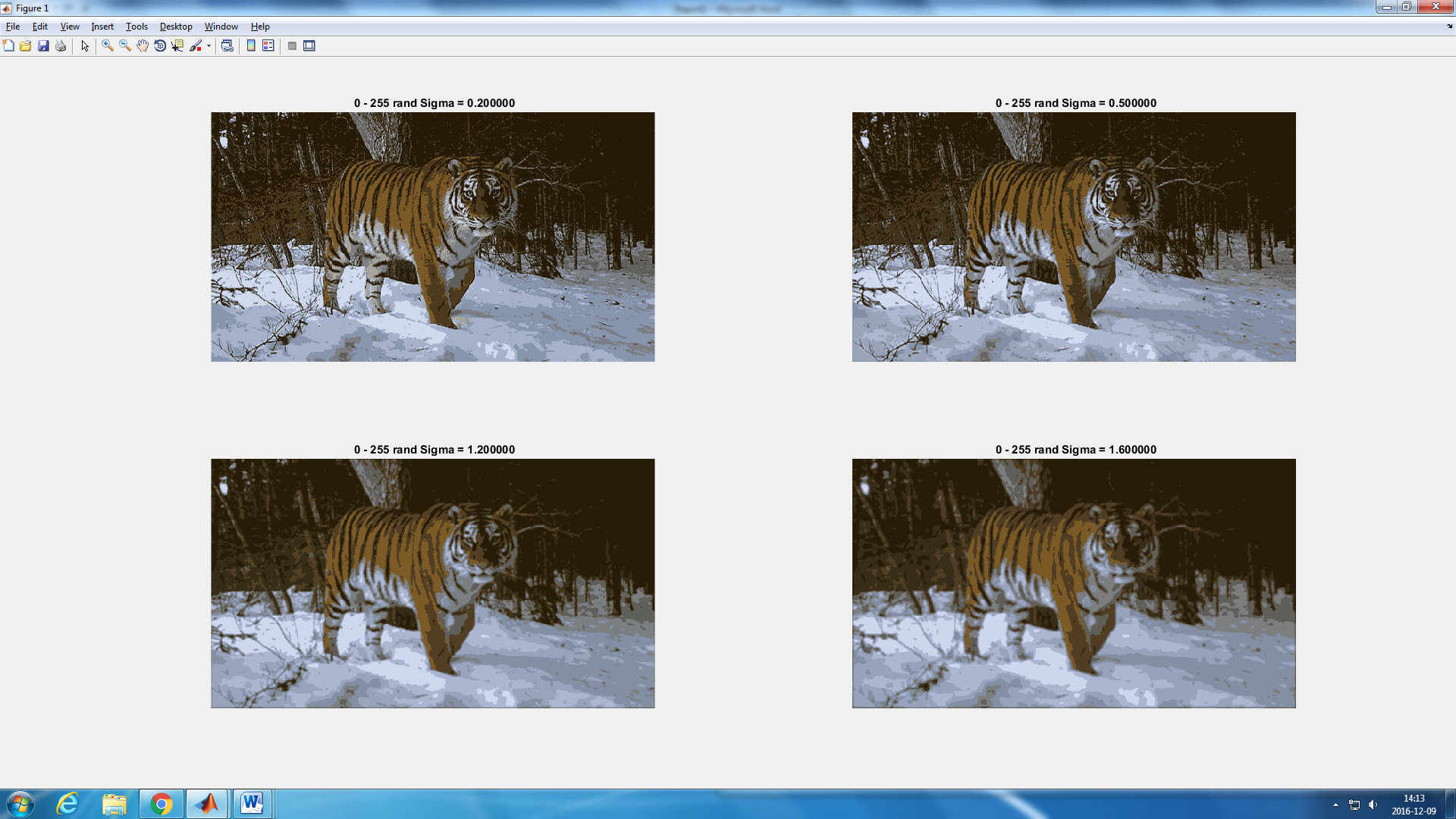


Figure 7: Output for different values of sigma for L = 10 and K = 10. Sigma = 0.5 (left) and sigma = 1.6 (right)

In the figure 7 we can observe that the superpixels are closer to be separated using lower values of sigma rather than higher ones.

Using lower values of sigma (0.5) we are now trying to get the optimal value of K in order to separate the superpixels of the background and the tiger. In the figure 8 we can see the difference between using different K. We can conclude that to completely separate all the super pixels, we need a high value of K. Even for 20 clusters the background and the tiger are not fully separated (figure 8)

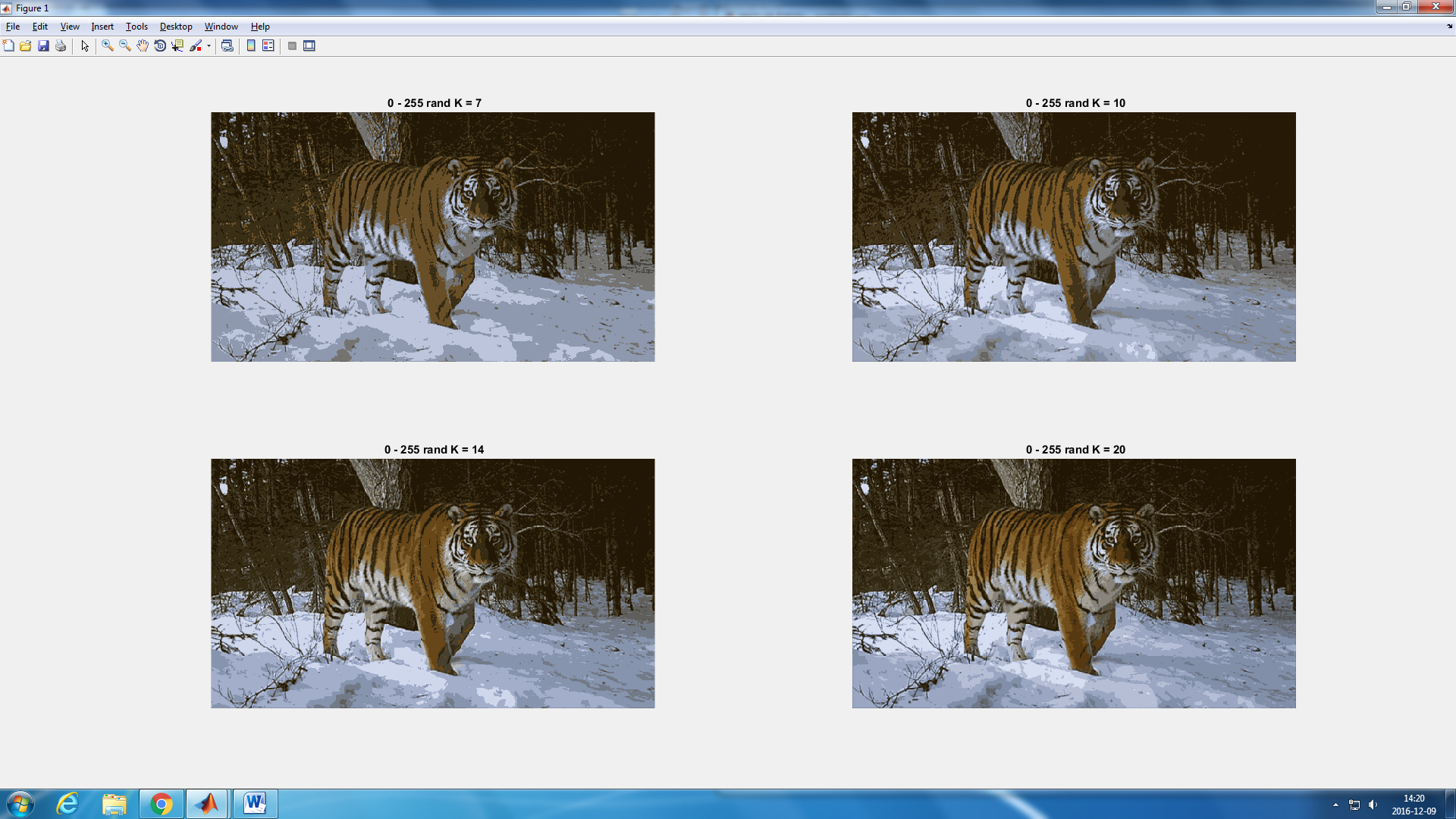
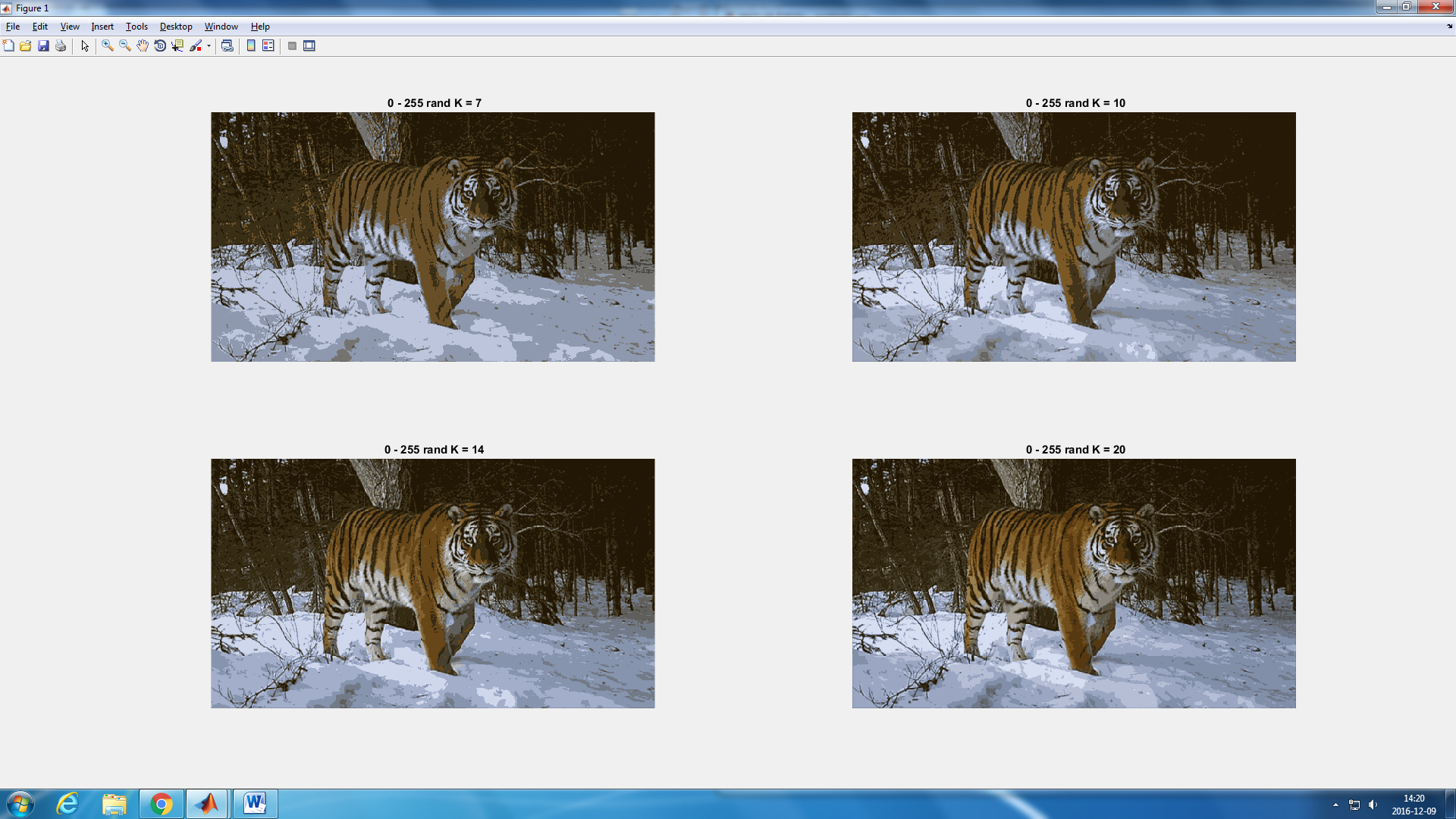
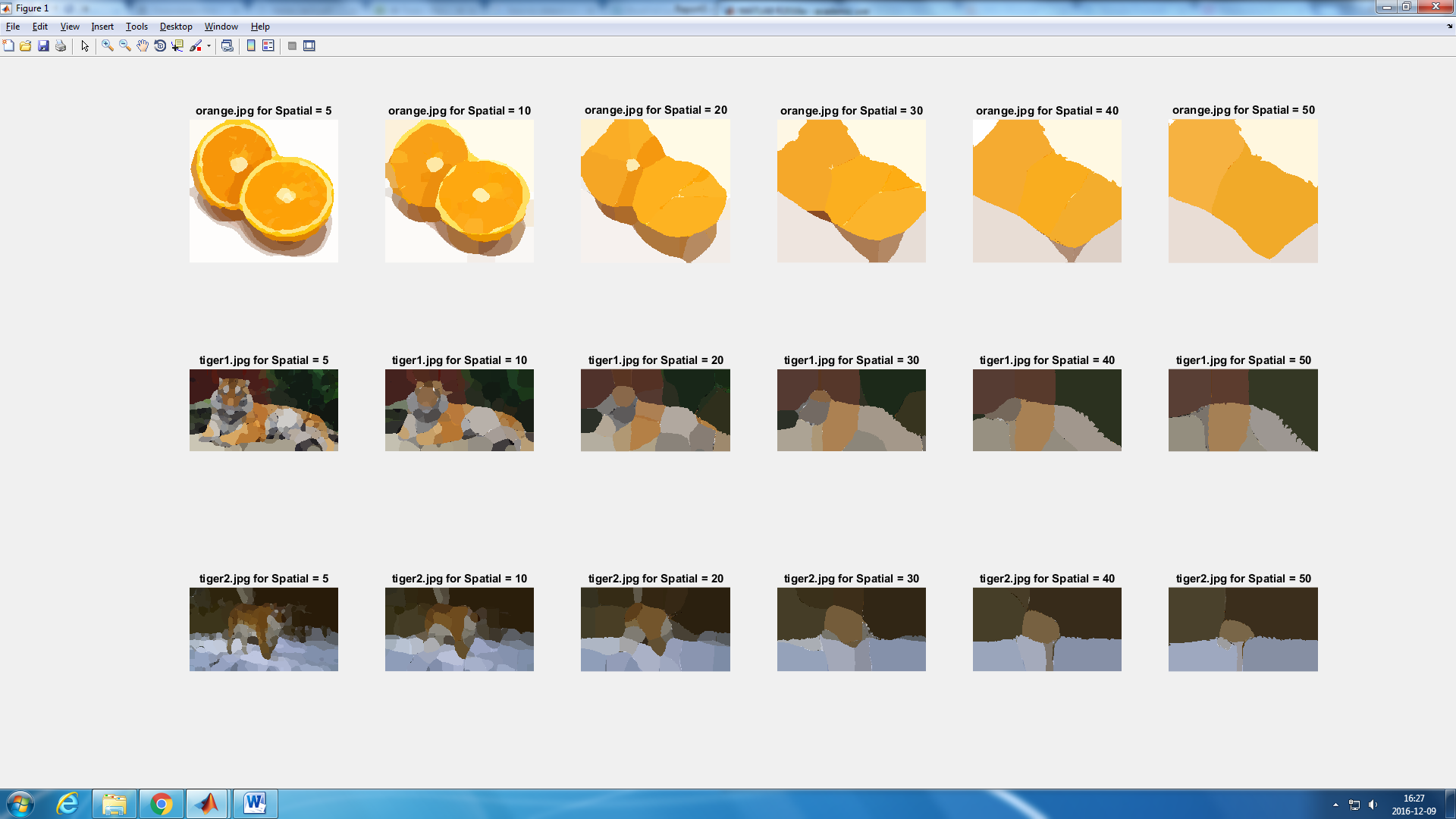
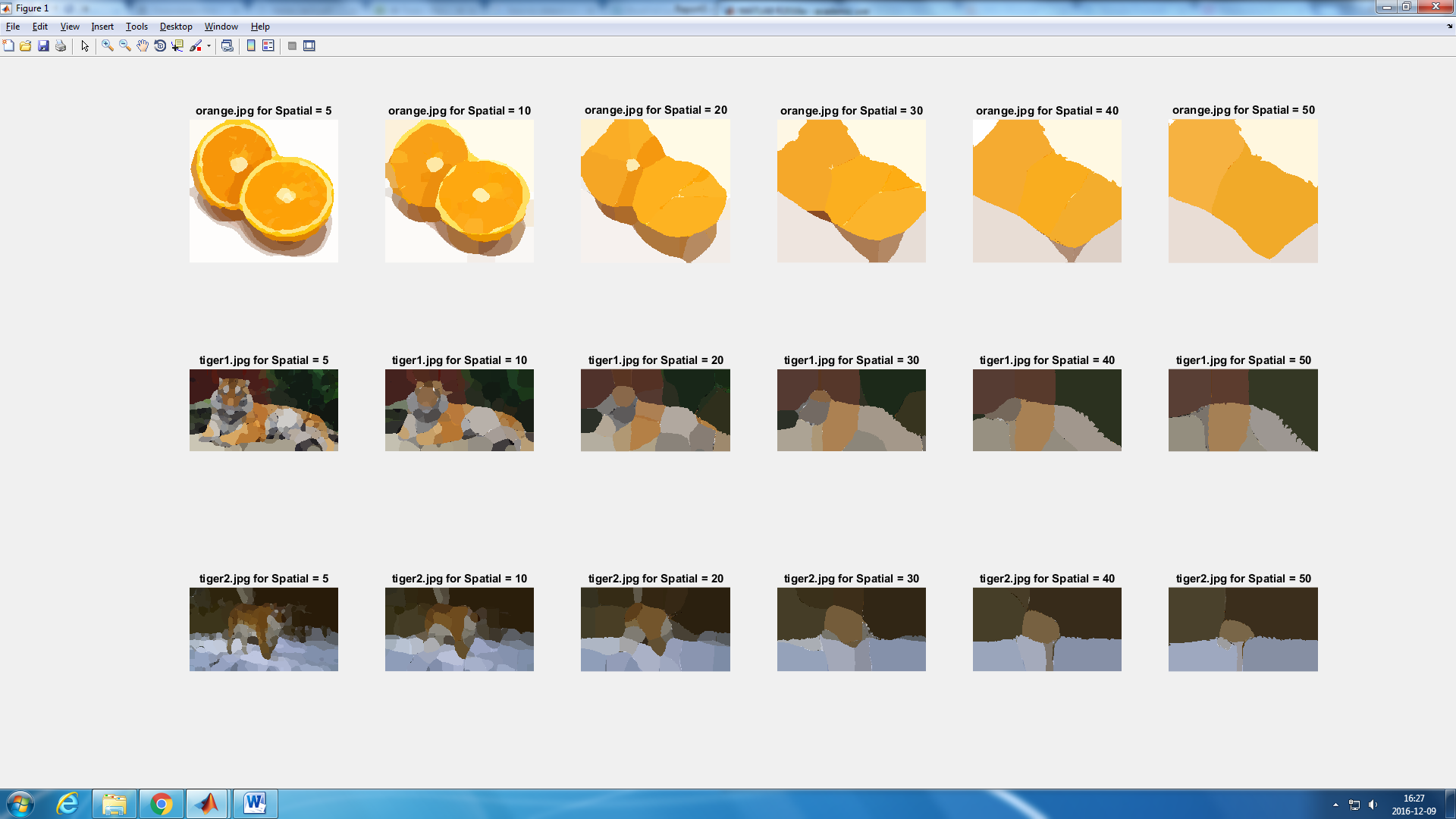


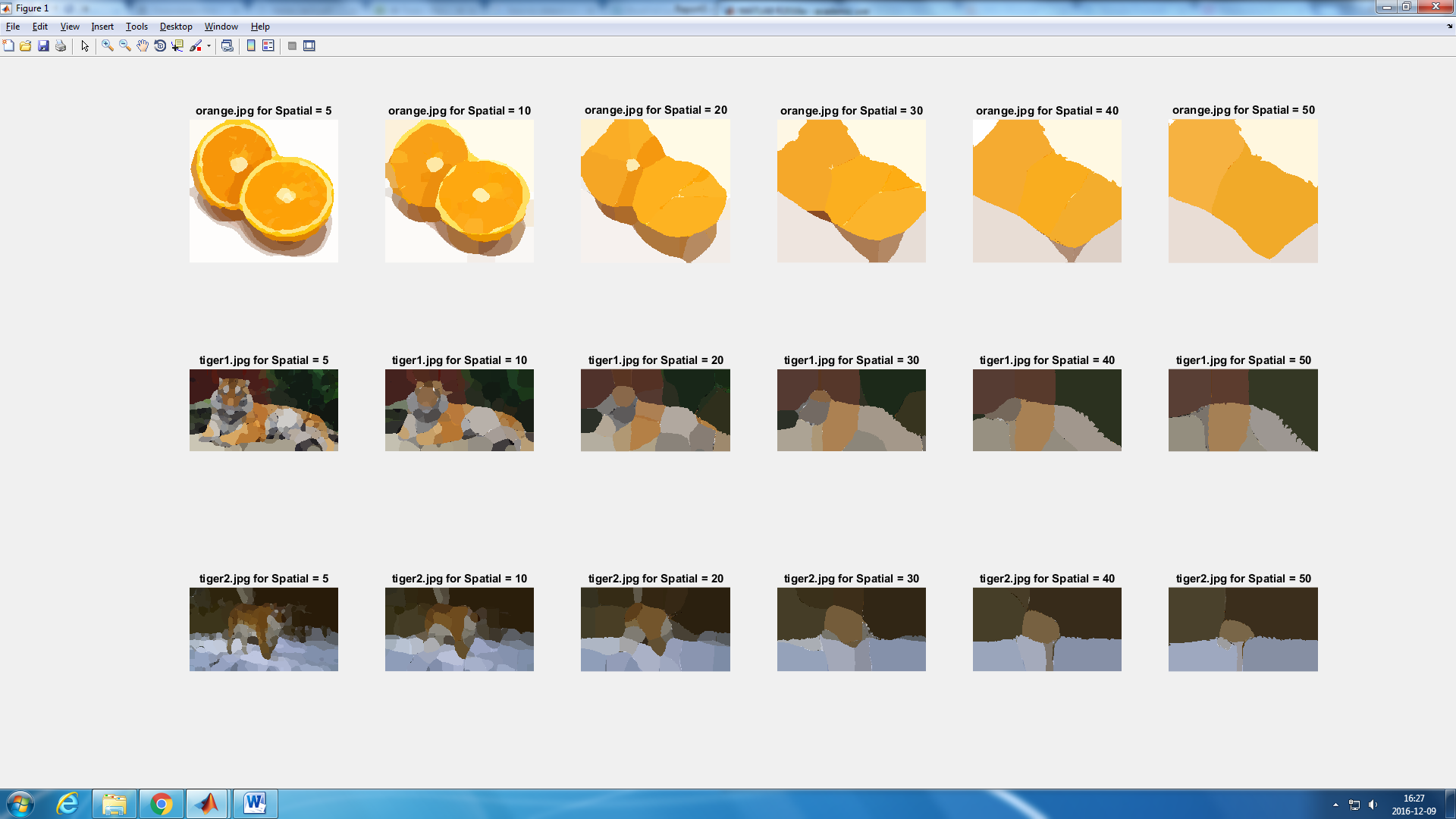
Figure 8: zoomed in output for K = 10 and K = 20 respectively

**Question 5: How do the results change depending on the bandwidths? What settings did you prefer for the different images?**

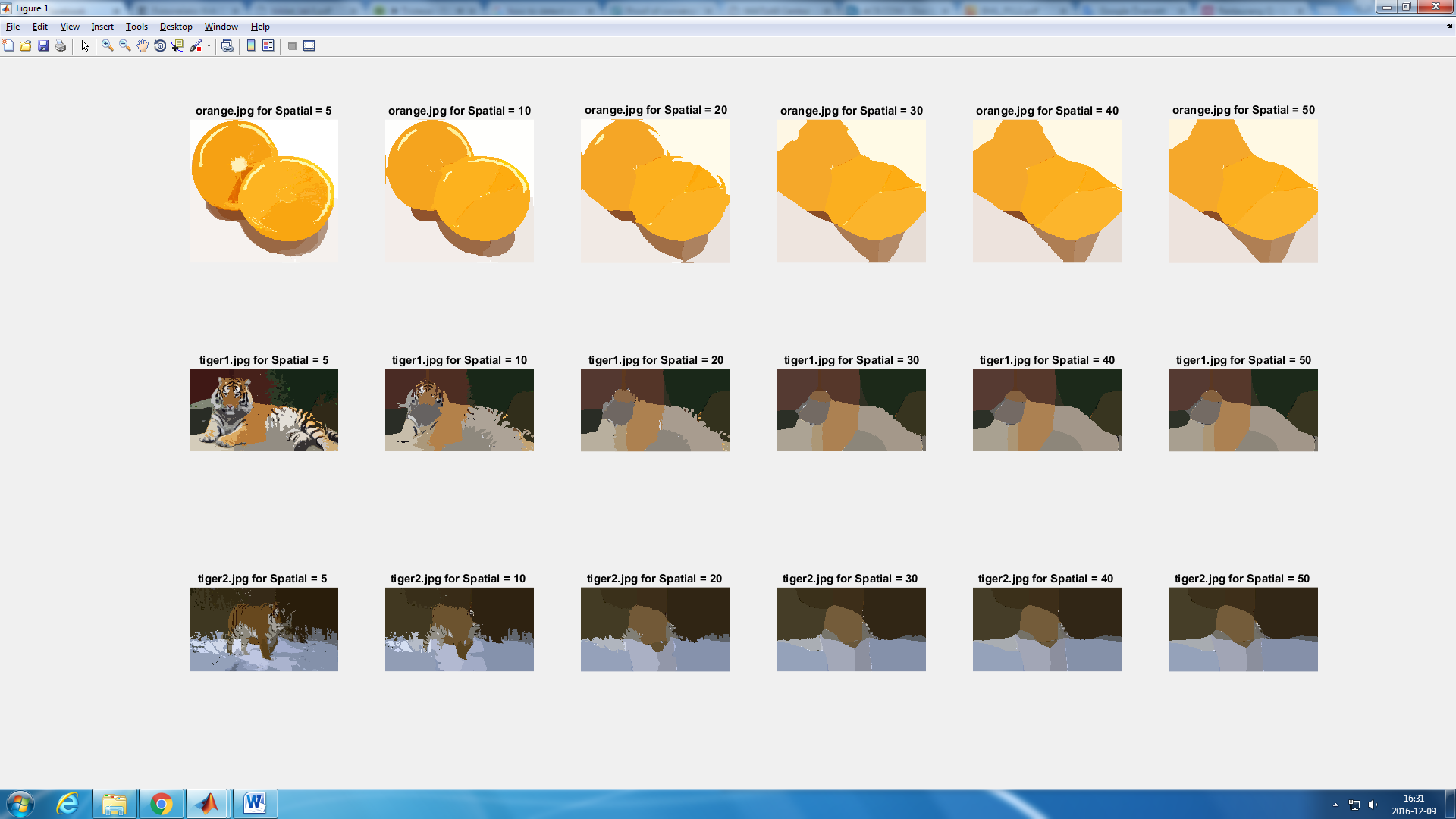
**Question 6: What kind of similarities and differences do you see between K-means and mean-shift segmentation?**

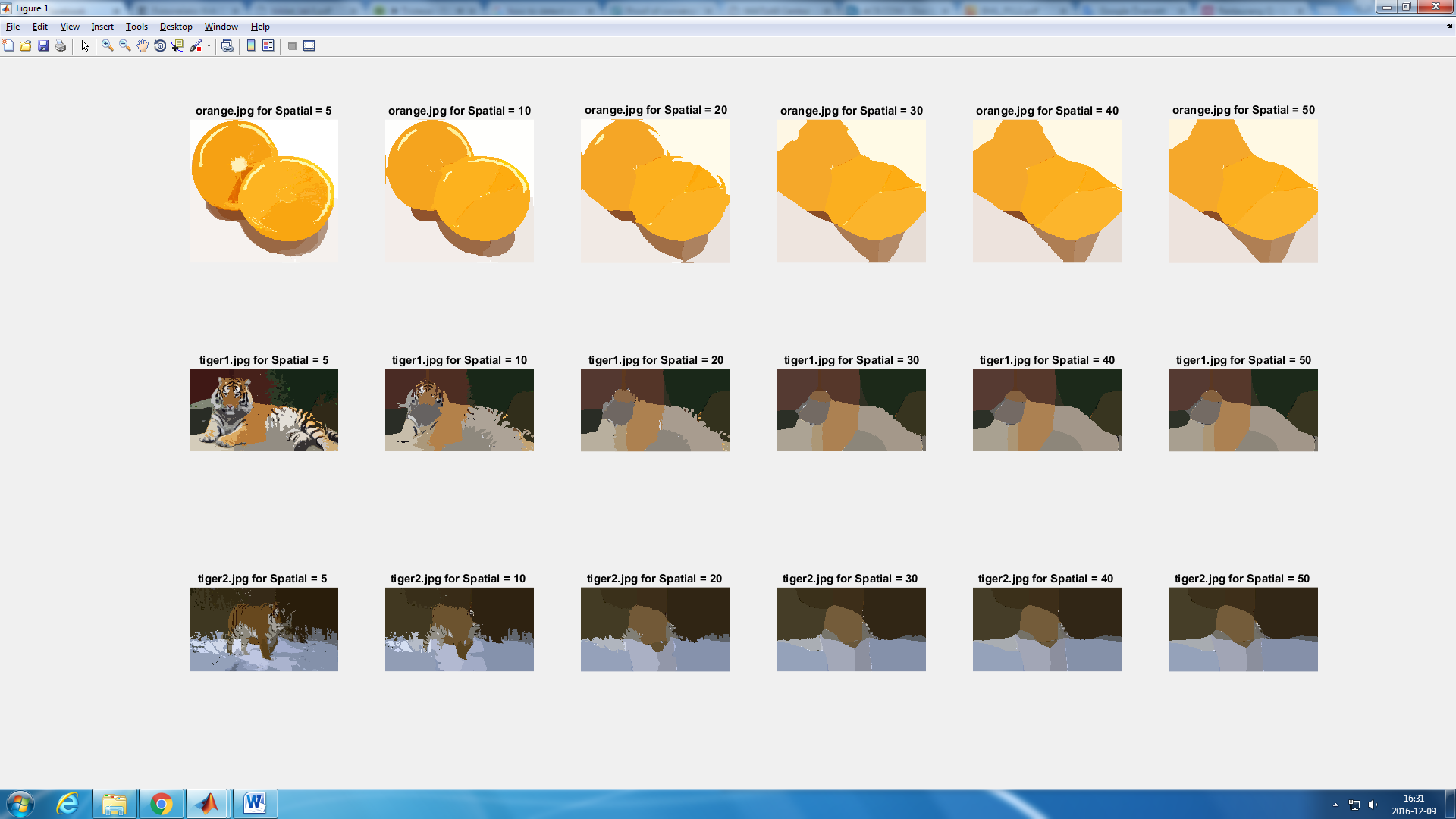


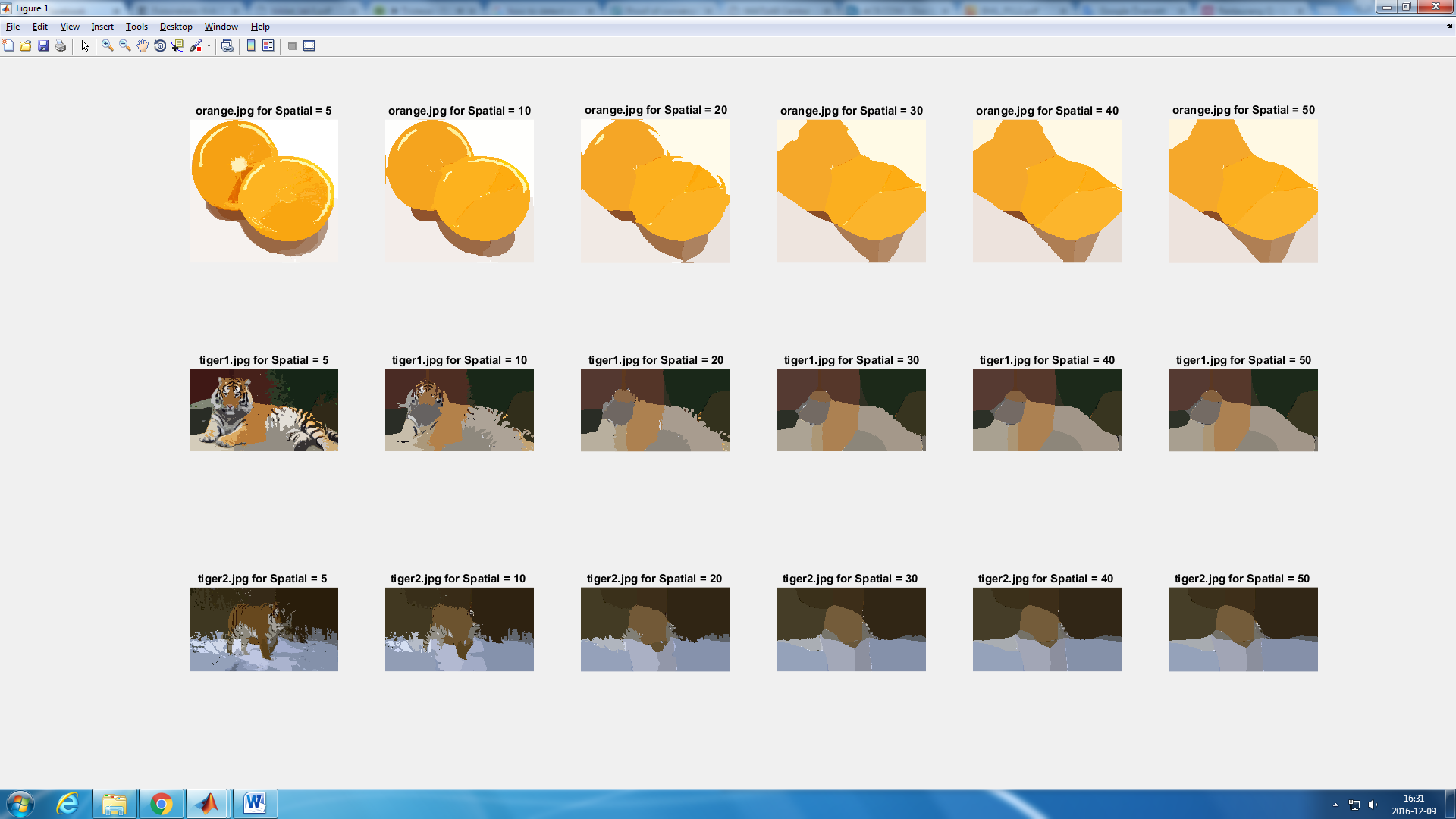




**Spatial**







**Color**