**VU 183.585
Computer Vision
Exercise Course Part 1
WS 2016/17**

*Nikolaus Leopold, 1327344
Alexander Reznicek, 1125076
Patrick Wahrmann, 1327120*

All three assignments are implemented as functions. For testing just call the function assignmentx(parameters) with the parameters (the image-filenames and for k-Means parameters k and d, there is no real error detection for wrong parameters implemented!). Further information about the functions can be seen in the header (i.e. help assignmentx).

Assignment 1: Colorizing Images

The program reads in all three color channels of an image and aligns the G- and B-channels to the R-channel. This is done by checking exhaustively over all displacements in a 30x30 pixel region wherein x and y direction the two color channels are rotated (shift by -15 to 15 in both dimensions and copy the pixels shifted out on the other side of the image). Because of this rotation and no cutoff of the border the image has in the worst case 15 pixels on each border as erroneous region, where the three channels have different information due to displacement. Although tested with corr2 we use for the correlation computation our manual way which implements exactly the formula of the assignment. This works pretty well, as shown in the example picture below:

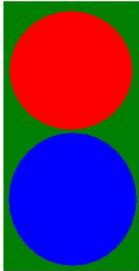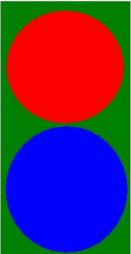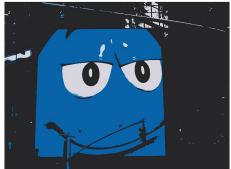**Assignment 2: Image Segmentation by K-means Clustering**

The image segmentation is done by nearly the exact algorithm given in the assignment, hence it will not be described in much detail. The random centroids as starting points were taken by using an uniform distribution from 0 to 1 for each scalar (hence k*D of these has to be taken where D stands for 3D=3 or 5D=5 case). Actually this gives us some problem which will be mentioned further in the experiments. As ratio for reaching the convergence thus terminating the algorithm 1.01 was chosen, which is subjectively a good compromise between iterations and clustering quality. Around the given algorithm with the parameters described above the algorithm was optimized for matrix operations. This lead to a pre- and postoperation where first the image is stuffed into an optimized matrix form for clustering and at the end the result is taken out of it to colorize the outcome image with centroid colors at the right places. But in one point the algorithm differs from the specification from the assignment: The new centroids are only computed for those which have at least one data point assigned to it. This behavior will be also discussed further.
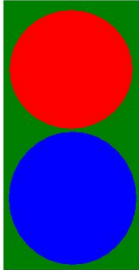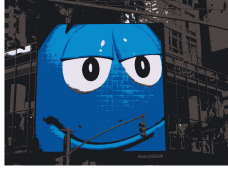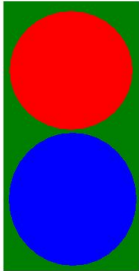Let's go to the experiments:

- *Show the results for all images in the case of 3D data points as well as 5D data points (using a fixed value of K). Discuss the results. Which data representation is better in your opinion?*

The experiments were taken for K=3 and K=5.

K=3:

	simple	future	mm
3D			
5D			

K=5:

	simple	future	mm
3D			
5D			

For both K the same conclusion can be made. In an intuitive way the consideration of the spatial information additionally to the color gives a bonus for near points and a penalty for far away points thus leading to more centralized color clusters. Let's verify this idea with the images:
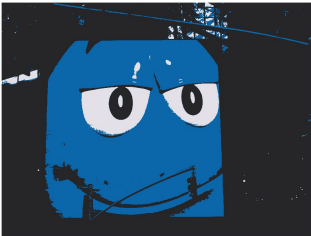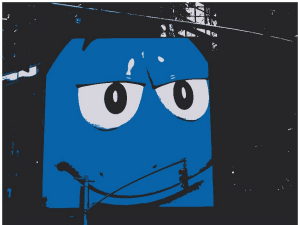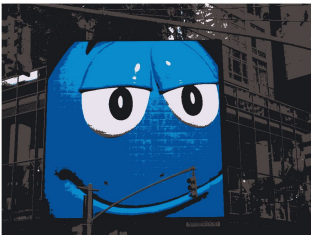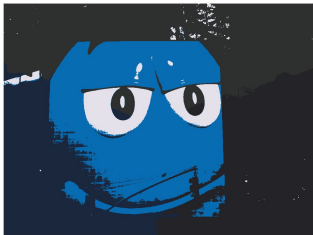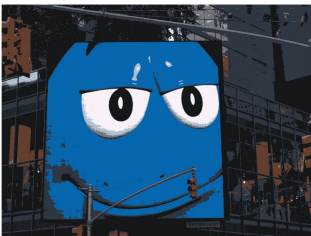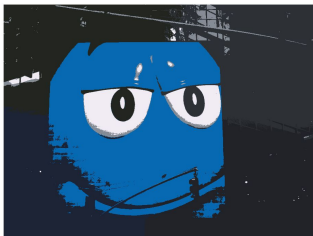
For the simple.png this clearly has no effect because K=3 can already cluster 3 colors so the spatial difference to the points don't change much.
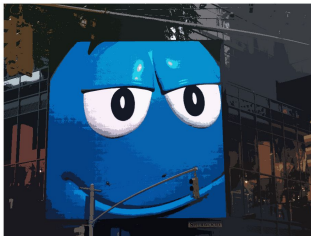
For the future picture exactly the opposite of the assumption is seen: the man in the front is more detailed and with more colors seen in the 3D-case. A possible answer to that curious behavior is that the image is a noisy cartoon where only a few colors are used but in more than one saturation for shadowing. So the clustering is oriented due to noise to information that is not really here (like the two background red colors in the K=D=5 case) leading to a lack of available centroids to show i.e. the shadows or the hair of the man.

mm.jpg witnesses the assumption. In the 5D-case the M&M on the screen is more defined against the background: either it is more detailed f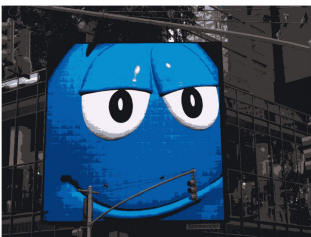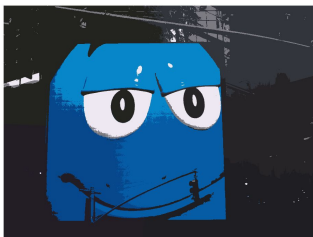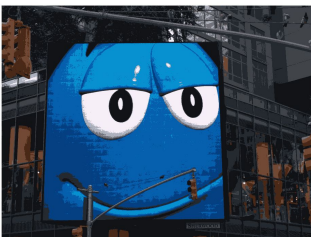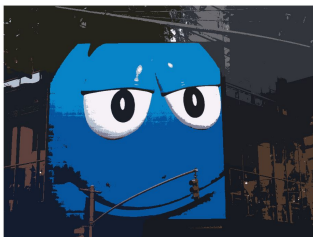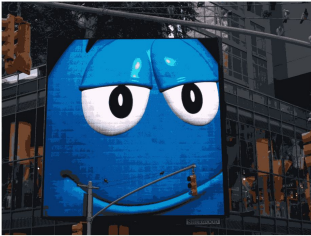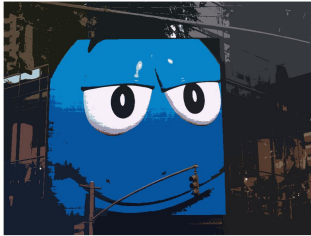or K=3 or it is much more elevated for K=5. So it clearly clusters the close regions with the same color much better than in the 3D case.

But which is better, 3D or 5D? It depends. In our opinion there are two use-cases for the two representations. First there is the case in which more of the structure should be seen. This is much more the case for the 3D-representation. But in the use-case where not the whole image should be seen (or to answer the question for higher k a more comic-like look is preferred) the 5D-representation suits more. It also gives a better image segmentation as it shows only parts of the image, so in terms of image segmentation 5D-representations are better. In a subjective manner for the human eye 3D is better since the structure of the image can be seen even with low K which means the image is much more simplified to the essential parts of the image.

- *Apply different values of K to the image mm.jpg and show the results for both 3D and 5D data points. Interpret the results.*

	3D	5D
K=3		
K=5		
K=7		
K=10		
K=14		

K=20		
K=50		
K=100		

The images leads to one easy conclusion: more K means more detail. Because the cluster centroids are the only colors the image has K stands exactly for the number of colors that are maximal available. This also answer the question why the detail is higher: more colors are more information. For low K just a few more colors can achieve a much more detailed image. But it has to be clearly stated that this holds only in a narrow band. In the comparison from 3-10 the difference is very high but for the much bigger interval 10-100 the difference is lower although there is some difference. In higher K the 5D-representation also shows the cartoon-like look which reminds of cel-shading in computer graphics.

The conclusion that can be given is to choose K wisely and not very high. Too low values give too less information but even low 2-digit Ks give enough information that is not much less than much higher Ks. And clearly for simple images (namely simple.png in the assignment) K don't need to be higher than the number of colors.

- *Where do you see - based on your results - the strengths and the weaknesses of the method?*

K-means is an simple algorithm which segment images by colors fast. Due to the iterative approach in exchange to the speed it can lead to arbitrary precision but gives a very good clustering after just a few iterations in our tested cases. A weakness of k-Means, the need of knowing k, is not a problem in our case as we know a boundary for a good k and choosing k one too low or too high does not have much impact in the right region.

But there is a big disadvantage. As in the description explained there is one change in the implementation in comparison to the assignment. The problem comes from the uniformly taken random centroids at the beginning. Because of the randomness the centroids can have the same position or in case of not using the whole color spectrum centroid can be more far away from all points than some other centroids. Both leads to the same result that one or more centroids aren't assigned to any of the data points. In that case K is lower than actually specified as the not assigned centroids don't contribute to the outcome. To not escalate the situation more not assigned centroids don't get a new centroid position (because then all centroids that are not assigned would have the same position) to give the chance of getting data points assigned in the next iteration where other centroids maybe are moved away. But even with that correction the clustering does not always work. As an example one possible outcome for "bad luck" at the centroids starting points the simple clustering for K=3 is the following:

In this case the 5D-representation help to get this wrong clustering rarer which makes sense in the way that red and blue are centralized and only green is around the picture so the difference between them gets bigger. In practice this enhancement of stability was actually seen when testing many times the same picture again and again.

It is also clear that the clustering although being good is not always exactly the same good. As an example even for no subjective difference in the following future-picture (K=D=3) the clustering took 10 iterations the first and only 4 the second time.
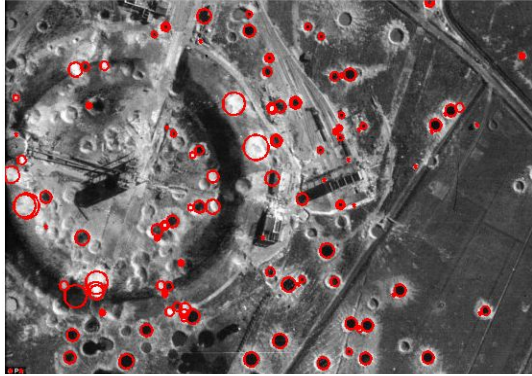

future with 10 iterations on the left in comparison with 4 iterations on the right (K=D=3)

A possible improvement in this situation would be making the starting centroids determinant. For this each centroid could be chosen by using a color of the image that wasn't assigned to another centroid before.

Assignment 3: Scale-Invariant Blob Detection

First it has to be noted, that the threshold θ chosen to eliminate maxima in the filter response which are not very characteristic and only caused by noise, has a major influence on the results as seen below:
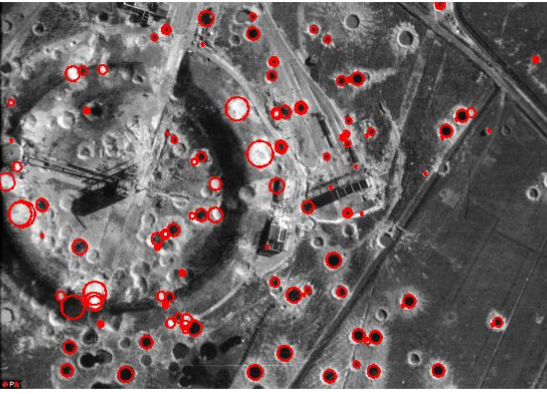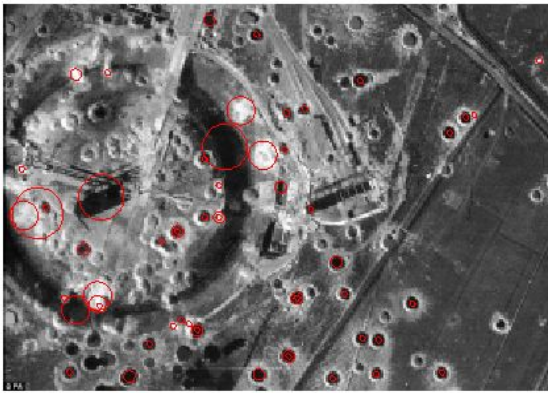
	butterfly.jpg	bombCraters.jpg
θ = 0.4		
θ = 0.25		

In the following work, the butterfly.jpg image is analyzed with θ = 0.4 and the bombCraters.jpg image with θ = 0.25 in order to ensure clear results. All other parameters are chosen based on the the advise given in the task description ("A reasonable
choice is for instance σ_0 = 2, k = 1.25 and 10 levels").
Because the use of loops is famously slow in Matlab, a trick has been used at the non-maximum suppression step. Instead of for-loops, the maxima were detected with image filters. These filters shifted the (response) image in all directions, such that a shifted image can be overlaid over the original one and easily compared with matrix operations in order to find out if a value at a pixel is bigger than all other neighbour pixels. The results of these steps were then also combined with matrix operations. If for loops had not been avoided in

this step it would have been necessary to iterate over 493x356=175508 pixels (resolution of the butterfly image) and as mentioned above, Matlab is not fast at this.

- *Apply the method to both the original images as well as to half-sized versions of them. Draw the detected blobs as circles with appropriate scale. Is the method able to find blobs in a scale-invariant way? If there are errors, what are the reasons for them?*

| Full Size	Half Size

Based on this experiment the blob detector seems to be relatively scale-invariant. Most characteristic points/circles are detected in both images (the white circles of the butterfly for example), but there are some that are not detected in the low-resolution image (the amount of detected points is decreasing from 84 to 56 and from 127 to 57) or are detected additionally in this version of the image (the shadow of the building/big antenna in the left part of the crater image).

The reason for the missing interest points could be the loss of high frequencies during resizing and therefore points which were detected on a small scale, could get lost.

The three very big circles, which can be observed in the half sized crater image (for example the shadow mentioned above), could be there but not in the original sized

image, because the scale of circles was too large to be detected in the original image, but due to the decreased file size the same real world area is represented by less pixels and a smaller filter can detect it.

- *Pick a detected keypoint and plot the response of the LoG for all scales in both image versions. The outcome should be a 2D plot where the x-axis represents the scale of the filter and the y-axis the filter response at the selected keypoint position. Describe and explain the difference between the two curves.*
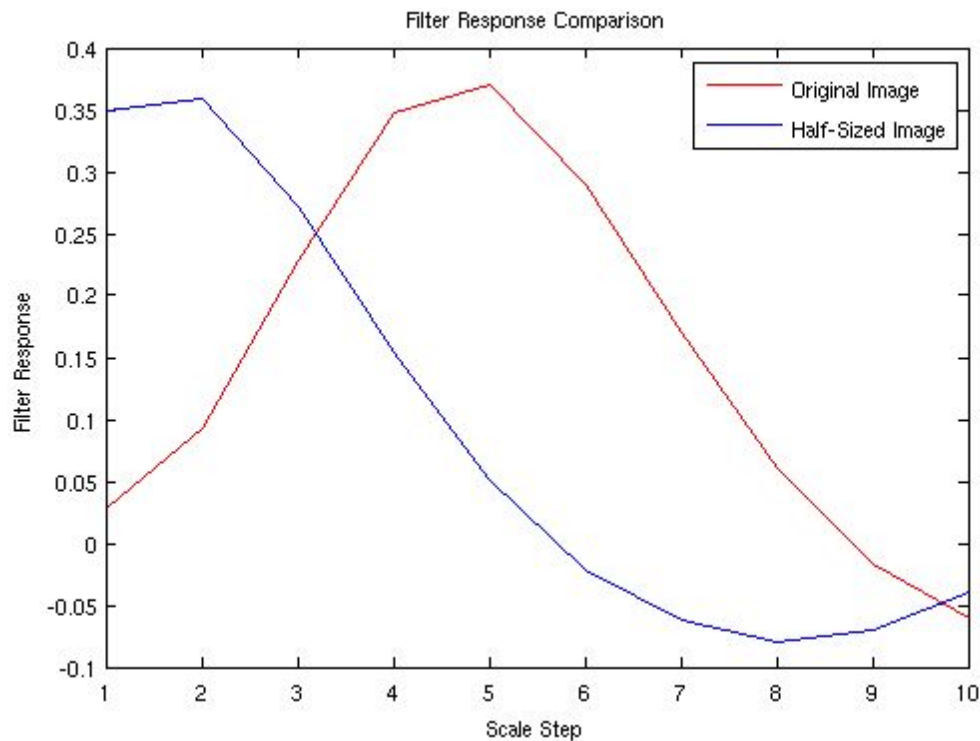
For this experiment the point with the coordinates (257/370) in the crater image and the corresponding point (129/185) in the shrinked image has been used. It represents a recognized circle (crater) in both images. The graph shows that the response of the half sized image is shifted in the negative x direction. That means that a value that is normally observed at scale 5, for example, is at scale 2 in the smaller image. This is caused by the fact that the reduced image size leads to the effect, that a crater/circle that has normally a radius of r pixels, has in the scaled image a radius of r/2 and is therefore detected on another scale. This shifts the whole response.