Zsófia Sára BUDAI
09.03.2023

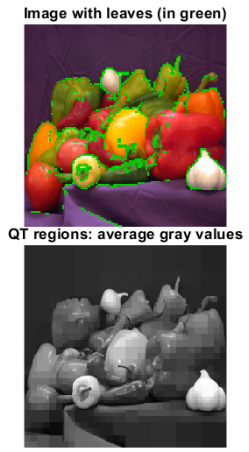
Lab Ass. 3: Pyramids and scale-space

The objective of this practice is to present the student methods for the scale analysis of the scene, so they can be familiarized with them and use them in the following assignments. For the understanding of the behavior of the functions proposed, it is essential to navigate through the MatLab help.

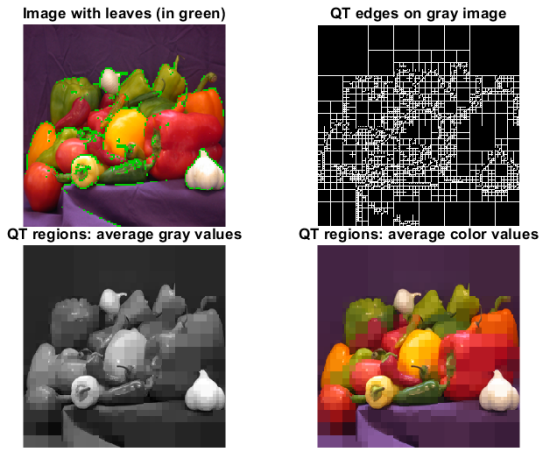
1. Quad-tree decompositions

a. Exercise 1

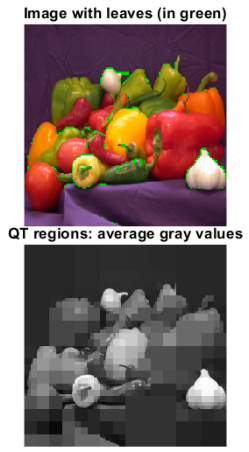
Using quadtree_Example.m I explored the effect of the threshold parameter α in the number and nature of leaf-regions. The parameter K defines the number of levels at which the original image will be divided maximum. Parameter alpha defines the number of leaves in the end for QT decomposition. HIgher alpha value results fewer numbers of leaves and smaller alpha value results in more leaves as it is visible in the following pictures.



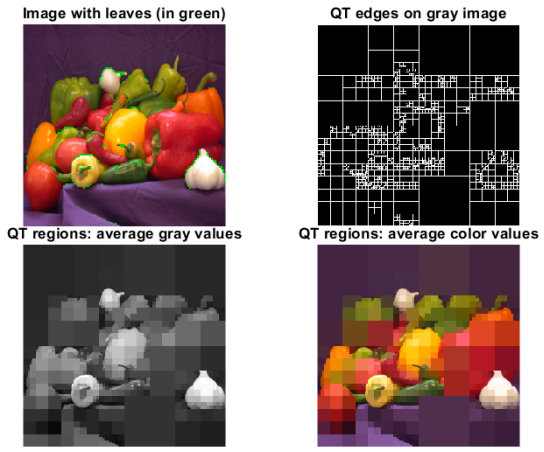
K = 8, alpha = 0.002



K = 8, alpha = 0.003



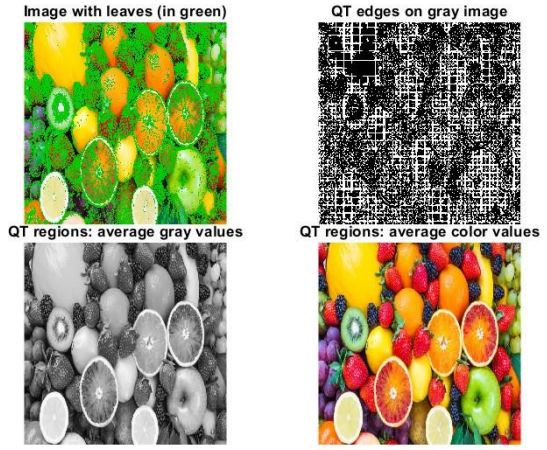
K = 8, alpha = 0.008



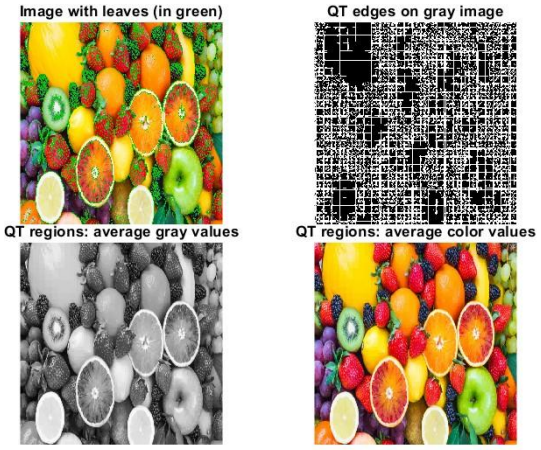
K = 8, alpha = 0.01

b. Exercise 2

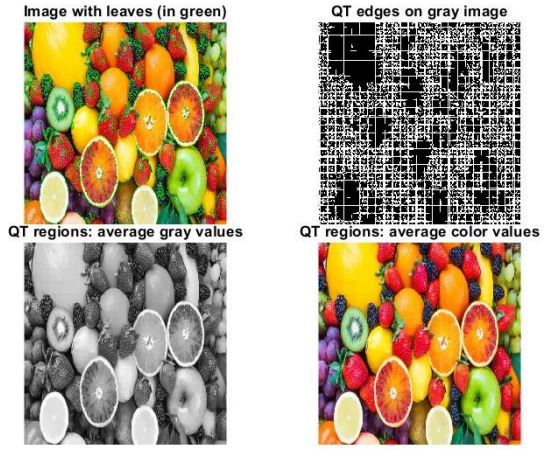
I implemented the quad-tree decomposition guided by a condition function on the regions' gray-level range: fun = @(x) (max(max(x))-min(min(x))) > alpha. With the new function I explore the effect of this function with different images and values for the threshold parameter as it is shown in the following figures. Higher alpha means less leaves as the window size is bigger and the fine details are smoothed out in the image. Using smaller values for alpha uses smaller window size and therefore smaller and finer details are marked as well.



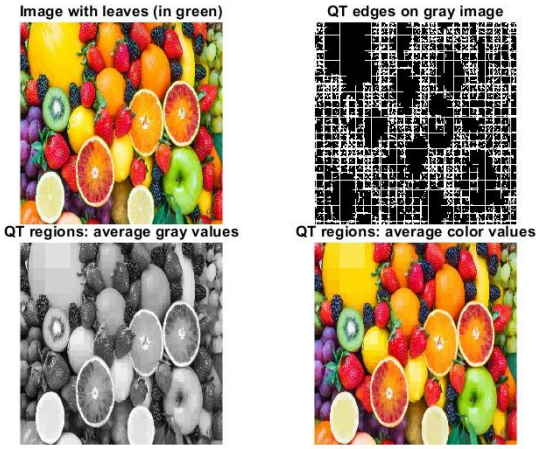
K = 10, alpha = 0.003



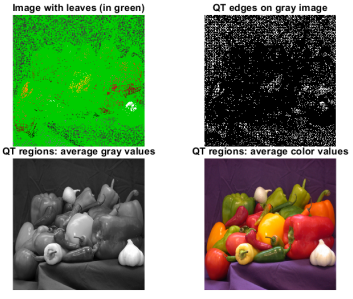
K = 10, alpha = 0.08



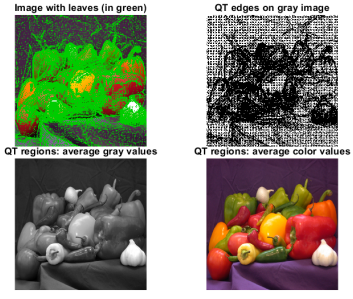
K = 10, alpha = 0.1



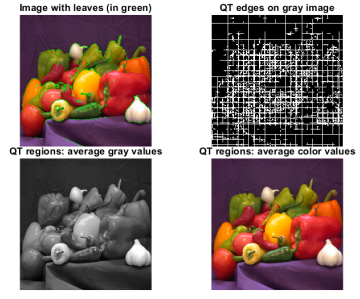
K = 10, alpha = 0.2



K = 10, alpha = 0.002



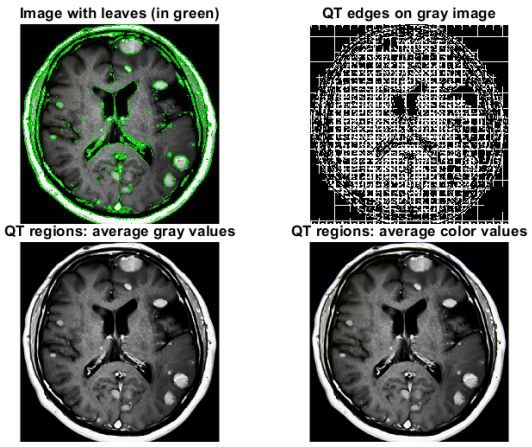
K = 10, alpha = 0.005



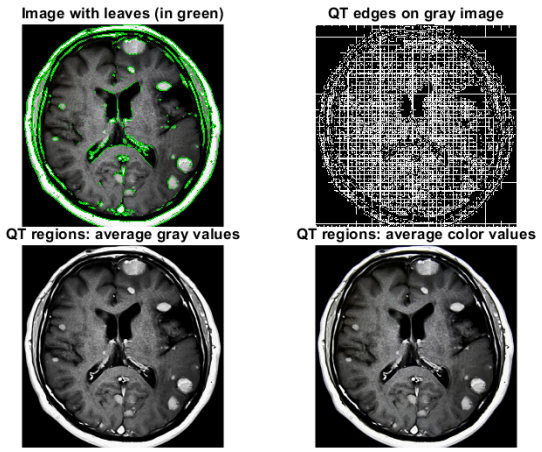
K = 10, alpha = 0.05

c. Exercise 3

I setted the threshold parameter in a way that the leaf-regions are aligned with the nodules' contours and brain boundaries of image ima_T1.jpg. The result and the used parameters are visible in the following table.



K = 10, alpha = 0.05

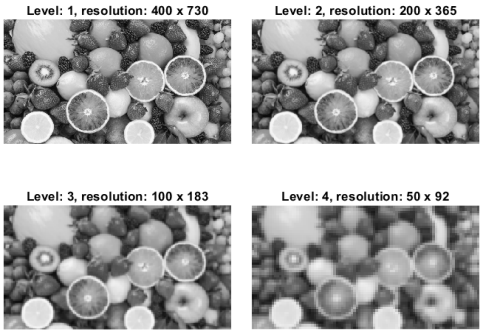


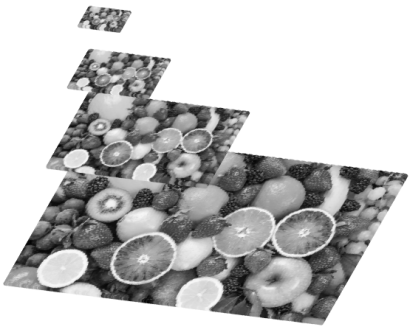
K = 11, alpha = 0.035

2. Gaussian Pyramids

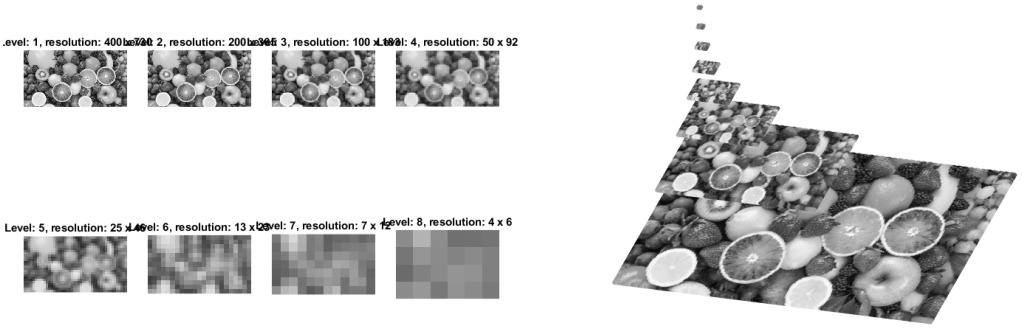
a. Exercise 4

With a bigger downscaling factor the edges and smoothes faster and the resolution of the pictures decrease faster as well. Using a larger downscaling factor will require less levels in the Pyramid to get the same result but the Pyramid contains less details in the levels. My tests are in the following table.

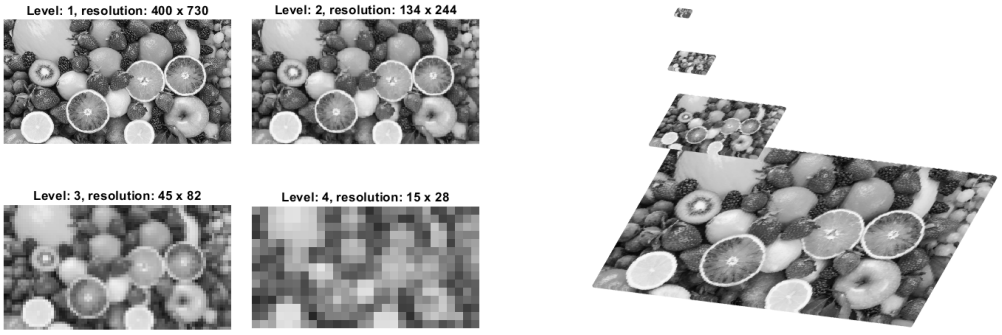




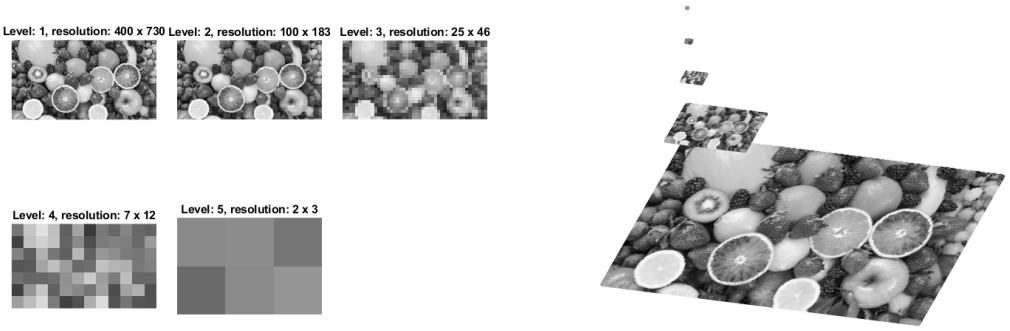
downscaling factor = 2
nlevels = 4



downscaling factor = 2
nlevels = 8



downscaling factor = 3
nlevels = 4

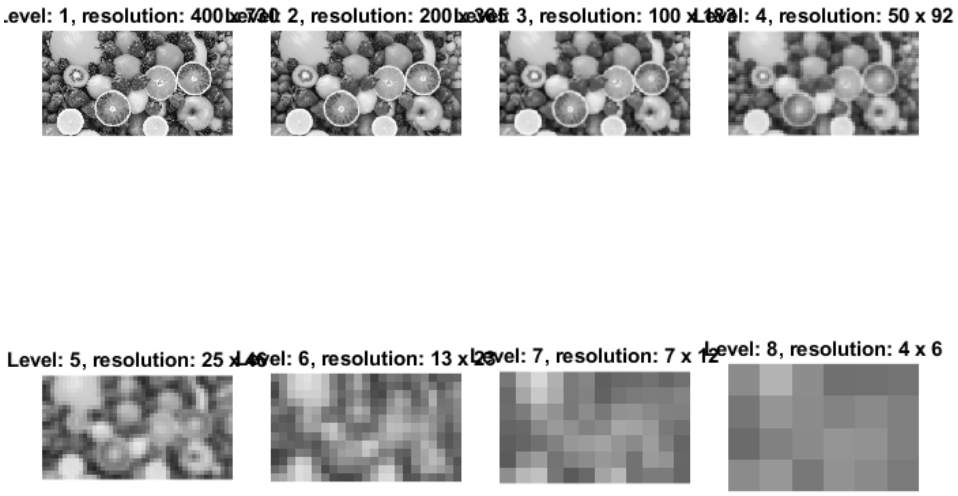


downscaling factor = 4
nlevels = 5

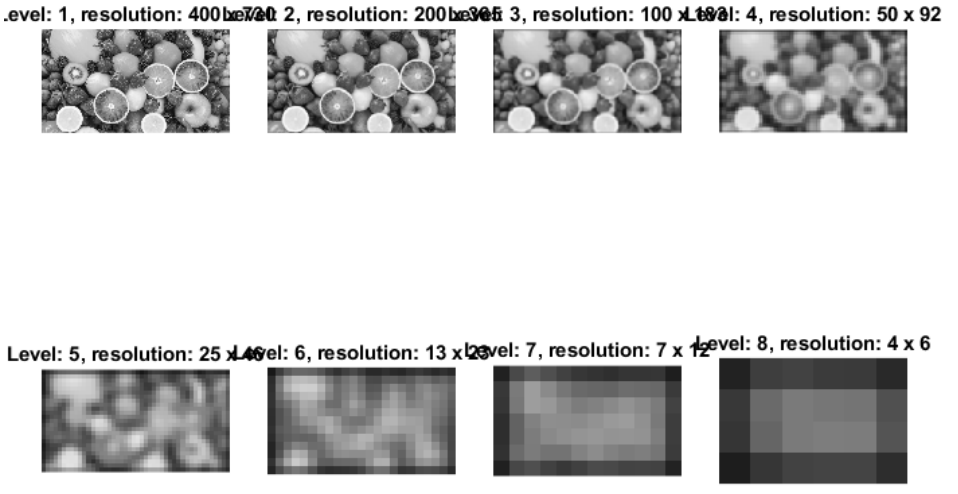
b. Exercise 5

The new level of the pyramid was calculated by replacing the Gaussian filtering by a mean and a maximum filtering with the following equations:

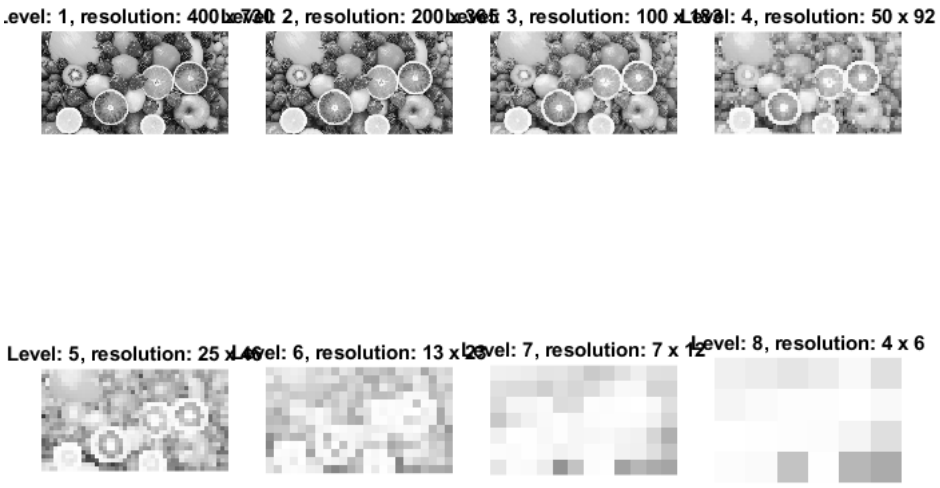
```
PG{lv} = colfilt(PG{lv-1}, [factor factor], 'sliding', @max)
PG{lv} = colfilt(PG{lv-1}, [5 5], 'sliding', @mean)
```



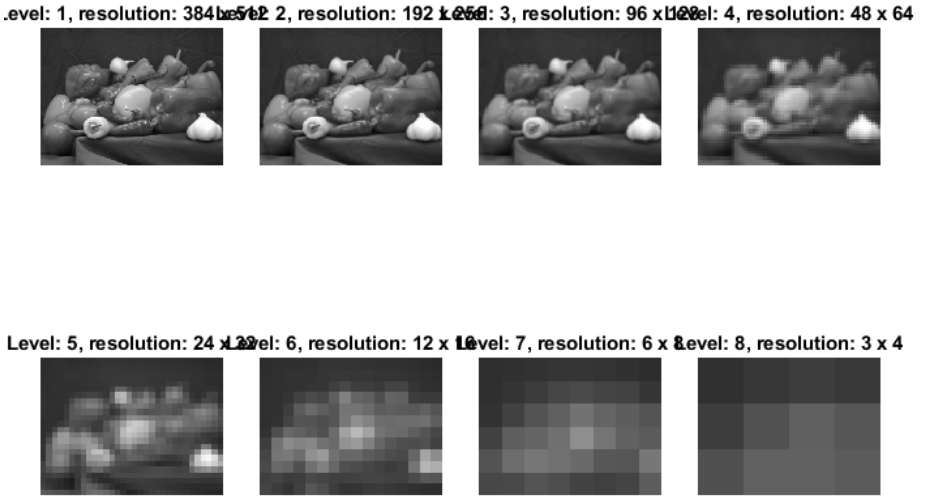
downscaling factor = 2
nlevels = 8
Gaussian



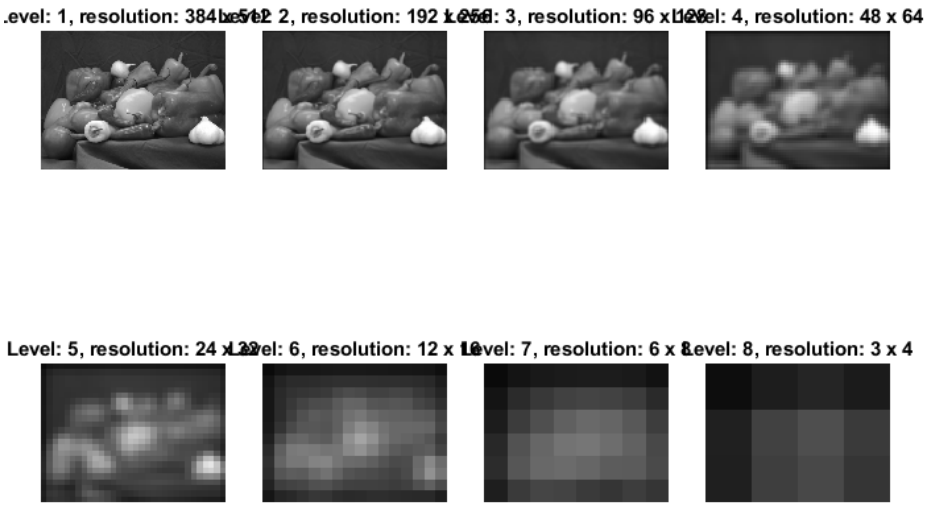
downscaling factor = 2
nlevels = 8
Average



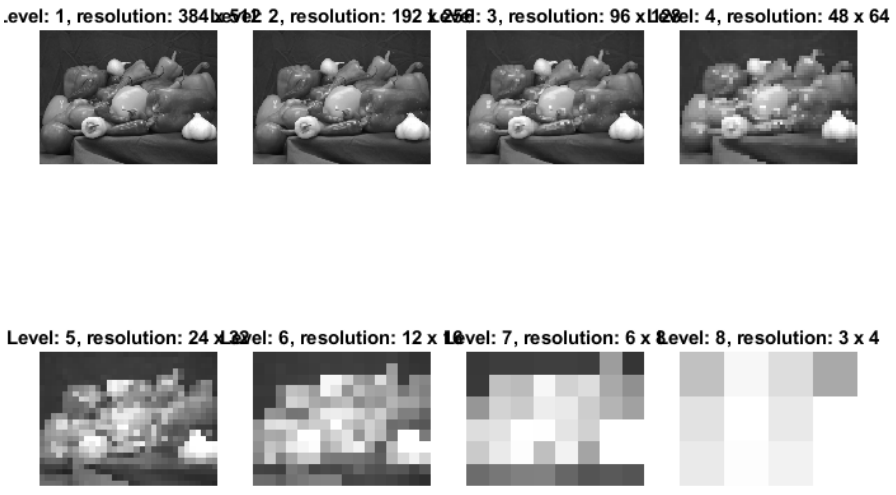
downscaling factor = 2
nlevels = 8
Max



downscaling factor = 2
nlevels = 8
Gaussian



downscaling factor = 2
nlevels = 8
Average



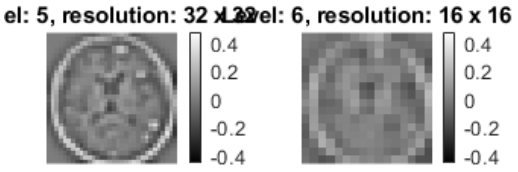
downscaling factor = 2
nlevels = 8
Max

In case of mean filtering the mean is calculated inside the window. Because of the lack of precious pooling the boundering pixels are darkened during the filtering process, modifying the final result. Using max filtering results in very bright images as the maximum value is calculated inside a window and the brightest/whitest values are influencing the resulting lower resolution image. The object boundaries are merged faster compared to the average filtering.

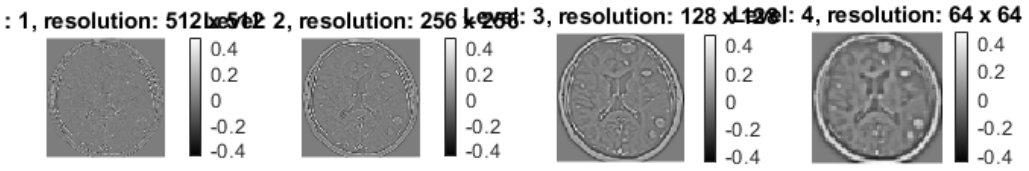
3. Bandpass Pyramids

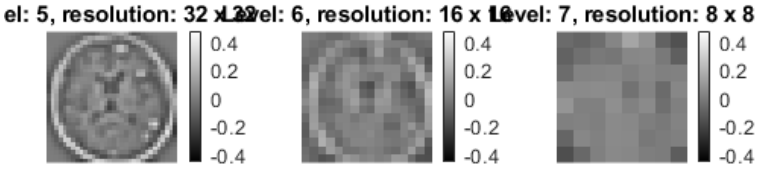
a. Exercise 6

The Bandpass pyramid function requires images to have a standard size therefore in order to use the *ima_T1.jpg* input image it needed to be resized to the standard size. The Bandpass pyramid produces a DoG (Difference of two Gaussian) output at various scales, which is efficient and effective. In the following pictures it is visible that it calculates the difference between two consecutive Gaussian images and because of that for nlevels=7 we got 6 pictures, for nlevels=8 we got 7 pictures and for nlevels=10 we got 8 pictures.

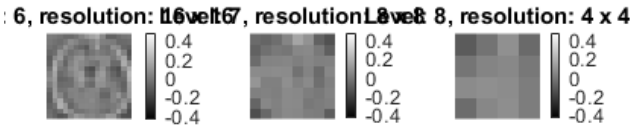


nlevel = 7





nlevel = 8

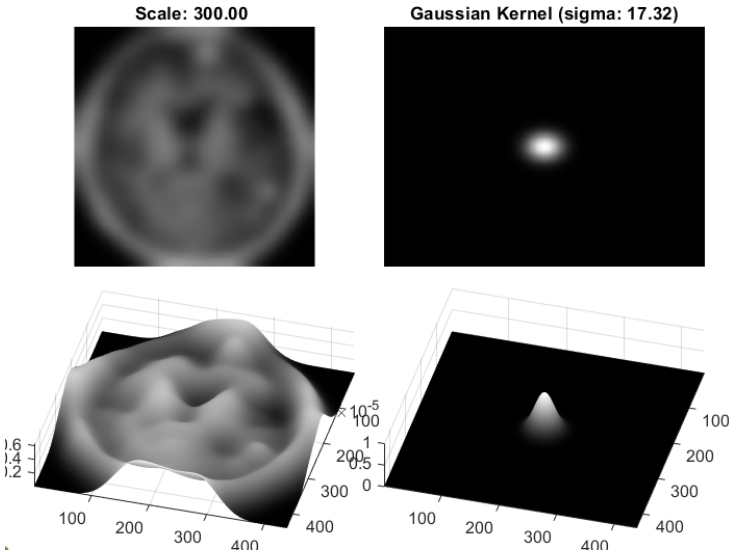


nlevel = 10

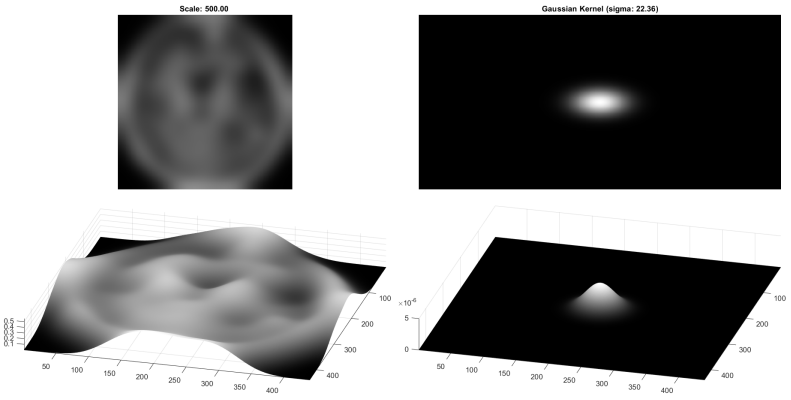
4. Gaussian Scale-space

a. Exercise 8

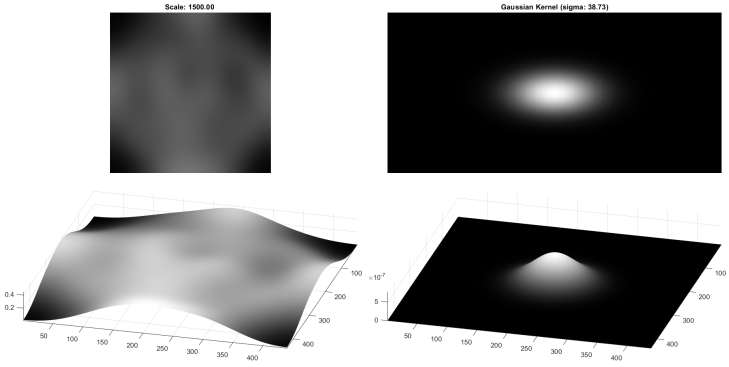
The Gaussian scale-space method is a technique used to identify local maxima and minima in an image. I observed using the Gaussian scale space function with three different scales and t0 values.



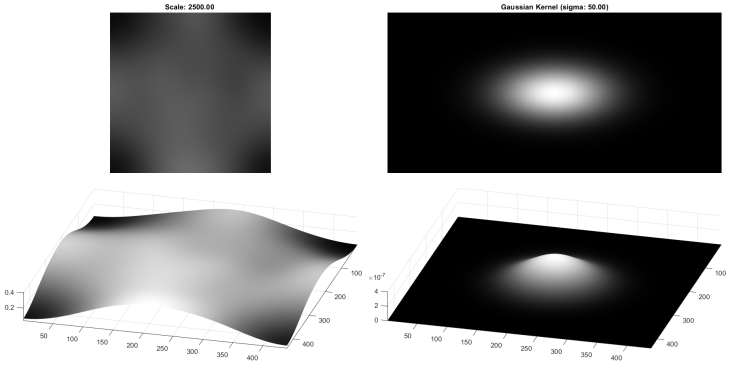
For nscale=300 and t0=1, the size of the Gaussian kernel was 17.32. In this case, I observed numerous local maxima and minima in the output.



For nscale=500 and t0=1, the size of the Gaussian kernel was 22.36. In this case, I observed less local maxima and minima in the output compared to the previous one.



For nscale=300 and t0=5, the size of the Gaussian kernel was 38.73. In this case, the kernel size was larger (38,73), I observed fewer local maxima and minima, as most of the smaller edges in the image were blurred out due to the larger kernel size.



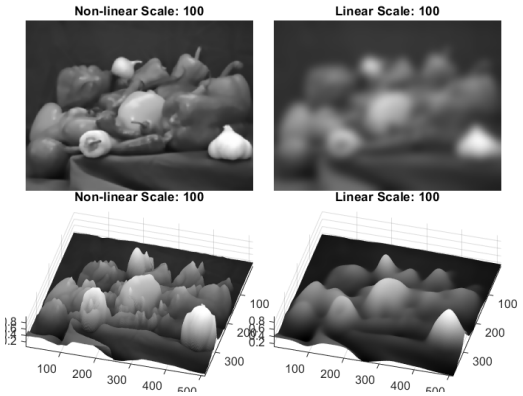
For nscale=500 and t0=5, the size of the Gaussian kernel was the largest with 50. There were the fewest local maxima and minima, although the computation took longer and due to the large kernel size, the whole image is blurred.

5. Non-linear Scale-space

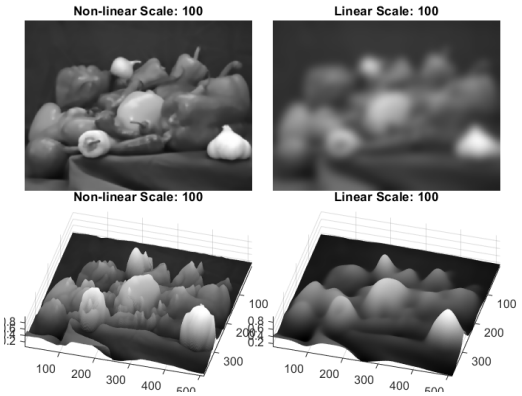
6.

a. Exercise 10

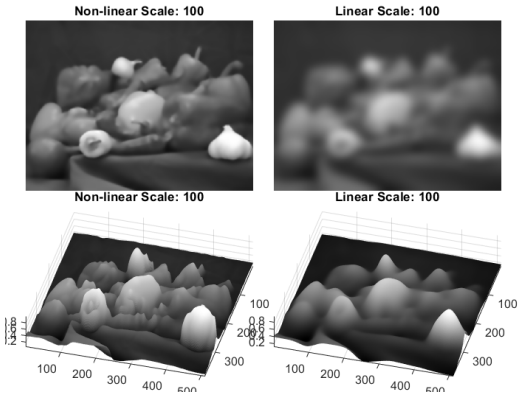
The method under consideration shows that changes in the parameters t0 and nscale have the same effect on the output. However, the homogeneity of the scaled regions is controlled by two other parameters, namely perc and psic, which determine the conductivity. The scaling process stops if these additional criteria are not satisfied, as shown in the figures. Despite this constraint, this method produces more precise and accurate output.



t0 = 1, perc = 0.7
nscales = 100, psi_c = 3



t0 = 1, perc = 0.7
nscales = 100, psi_c = 5



t0 = 1, perc = 0.9
nscales = 100, psi_c = 3

t0 = 1, perc = 0.9
nscales = 100, psi_c = 5

b. Exercise 11

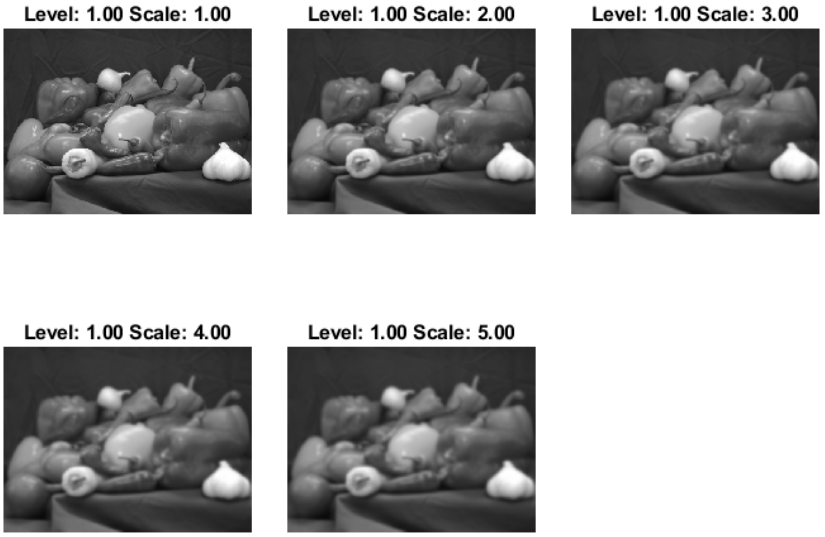
The choice of the conductivity function can have a significant impact on the performance and accuracy of the method.

- Perona-Malik with psi_c(s) = 1 / (1 + (s/ kappa)^2) where s is the scale parameter, and kappa is a constant that determines the strength of the diffusion. It is used in edge detection and segmentation because it reduces noise in the image and sharpens edges.
- Edge-Enhancing Diffusion with psi_c(s) = exp(-(s/ kappa)^2) where s is the scale parameter, and kappa is a constant. It allows more diffusion in the perpendicular direction to the edges, while limiting diffusion along the edges.
- Weickert with psi_c(s) = exp(-c^2 / (s^2 + epsilon)) where s is the scale parameter, and c and epsilon are constants. It has a smoother transition between the diffusion behavior in the homogeneous and inhomogeneous regions.
- Charbonnier with psi_c(s) = (s^2 + epsilon^2)^(-alpha/2) where s is the scale parameter, and alpha and epsilon are constants. It preserves edges while also reducing noise in the image.

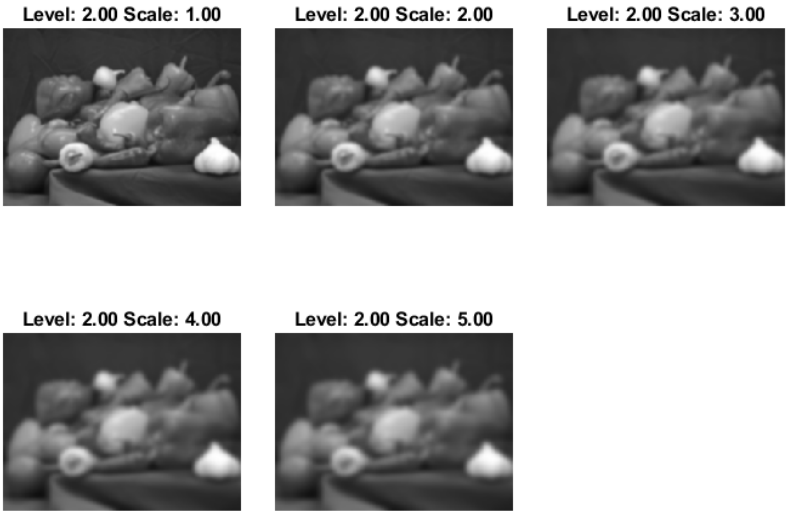
7. Global exercises

a. Exercise 12

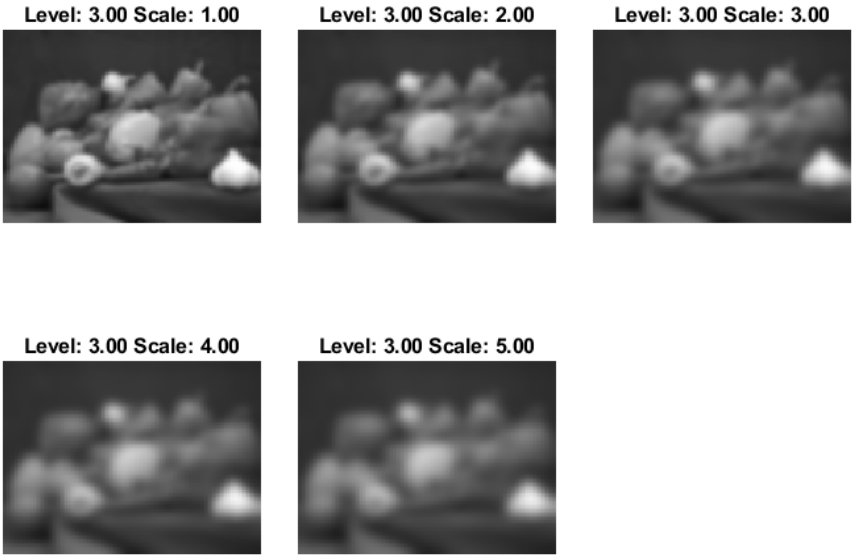
The code can be found in the scale-space folder in exercise12.m file.



Level 1 at 5 scales



Level 2 at 5 scales



Level 3 at 5 scales

I have generated a scale-space Gaussian pyramid, where each level consists of multiple scales (5) of scale-space decomposition. To initialize the image for each new level, I have performed sub-sampling on the image at the middle scale of the previous level. As a result of this approach, I have observed that the images at subsequent levels and for different scales are smoothed using a Gaussian kernel, which causes us to lose some information.

b. Exercise 13

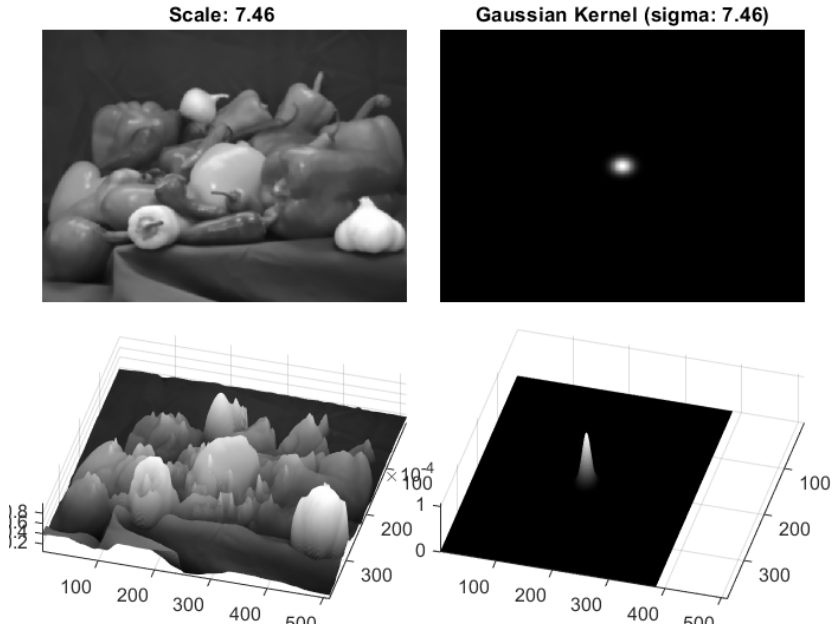
I have generated the exercise13.m in the folder of scale-space with the helper function Linear13.m and in the folder non-linear scale-space with the helper function

NonLinear13.m. I used the equation given for calculating Gaussian standard deviations.

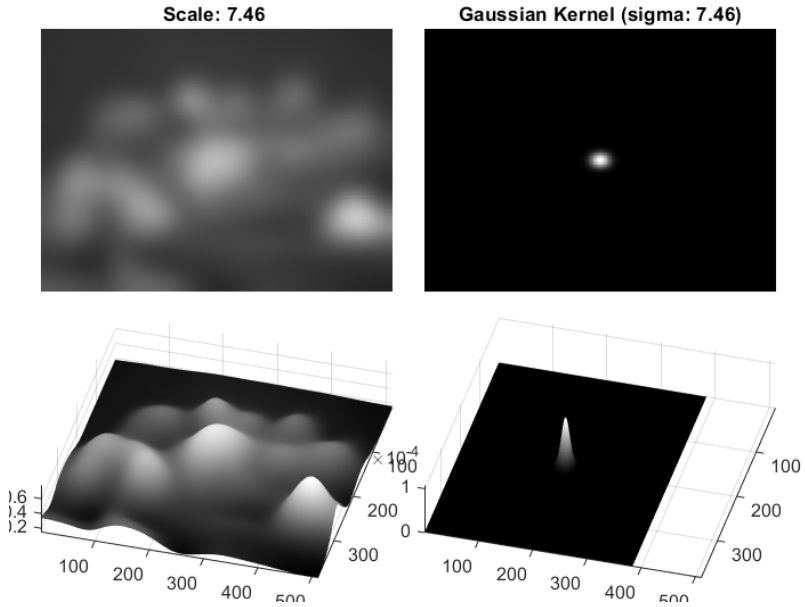
$$\sigma_j(o,s) = \sigma_0 \cdot 2^{o+s/S},$$

$$o \in [0,\dots,O-1], s \in [0,\dots,S-1],\ j \in [0,\dots,N-1]$$

By running it on fixed (10) number of scales I've got the following results for non-linear:



And the result for linear:



From the result non-linear scale-space preserves the edges in the image and avoids blurring out important information, while still providing some degree of smoothing to remove noise and small details.As a result, non-linear scale-space preserves the edges in the image and avoids blurring out important information, while still providing some degree of smoothing to remove noise and small details.