COV A_4 - Anisotropic Diffusion

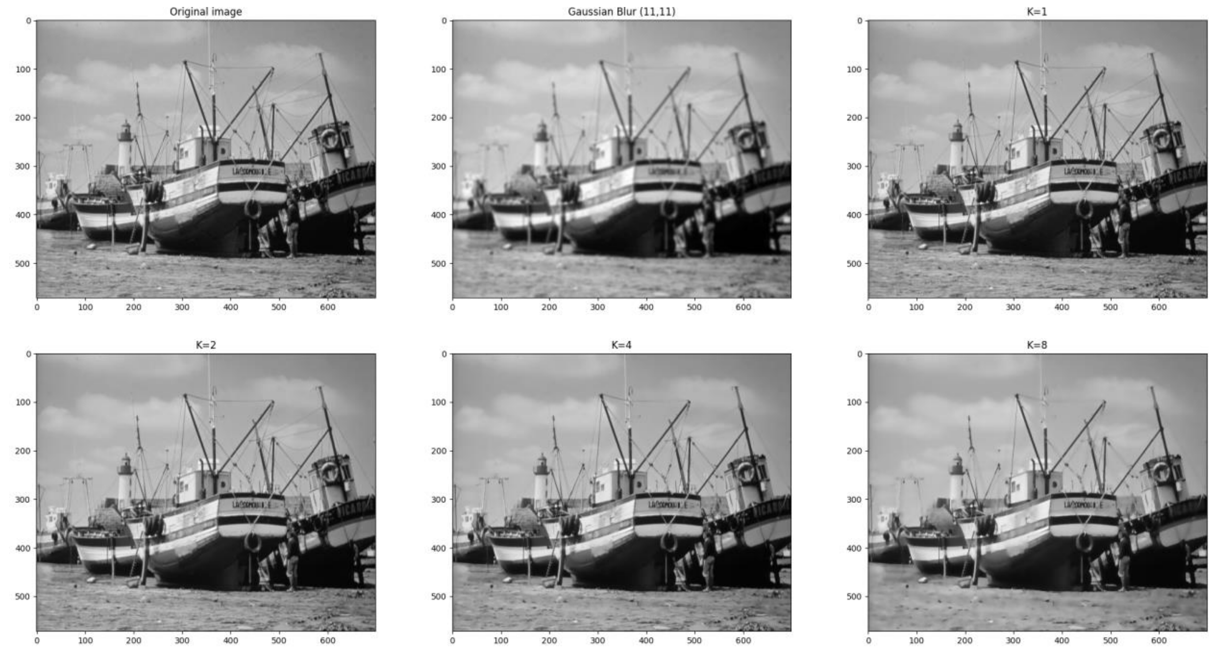
In order to apply anisotropic diffusion, the image gradient as well as the diffusion number is needed. Instead of computing it by hand by specifying the gradient masks, this can be done easily using the np.gradient() method and applying Pythagoras theorem to get the magnitude of the resulting vectors.

After calculating the diffusion number c, the gradient delta (update term) can be calculated and added to the image with a given factor delta t.

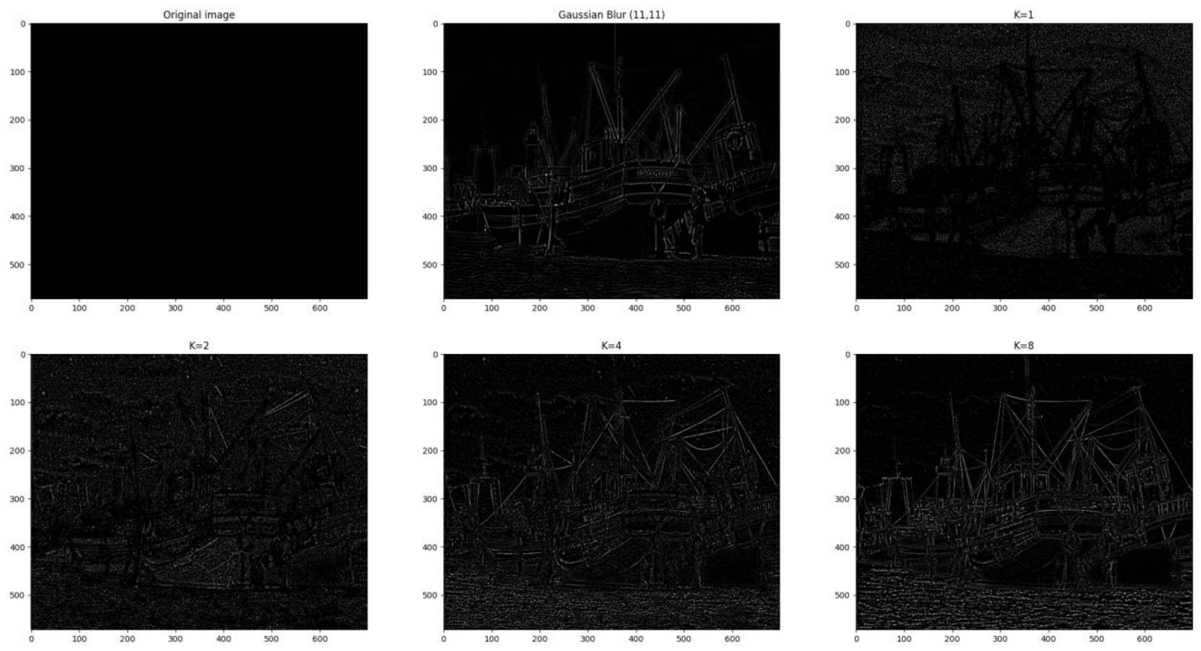
This is done iteratively for an adjustable amount of iterations.

One important parameter of Anisotropic Diffusion is the *Kappa* Parameter (K), which allows us to favour either edge preservation (lower numbers for kappa) or smoothing (higher numbers). Choosing the parameter is dependent on the images it is applied to and the effect that the user wants to achieve.

To show the process and my thoughts, I've calculated several images at once with 50 iterations and a delta t of 0.1 and shown the difference to the source image the image with a gaussian kernel applied to it, similar to what is shown in the slides.

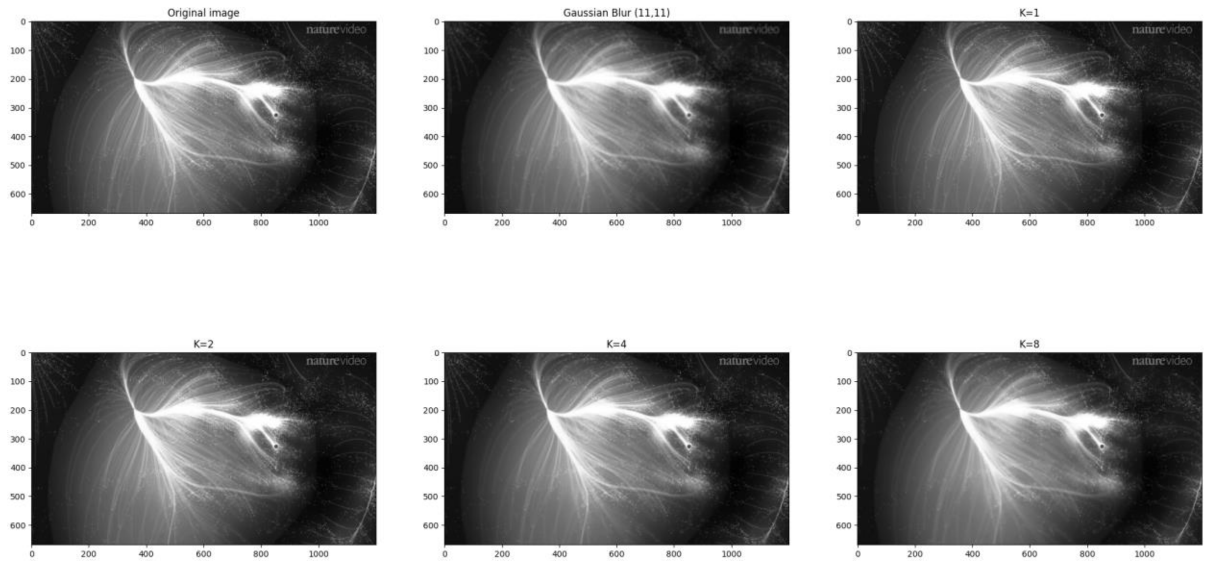


In this example, the standard ImageJ boat is used. Especially in the last image with K=8 it is evident, that the outlines of the boat and the ropes and masts are still sharp, while everything else that was noisy (e.g., the sea, the side of the boat, and the sky) is now smoothed. However, I consider that one step too far and would choose K=4 (or something inbetween), as it keeps much more detail and k=8 looks too muddy.



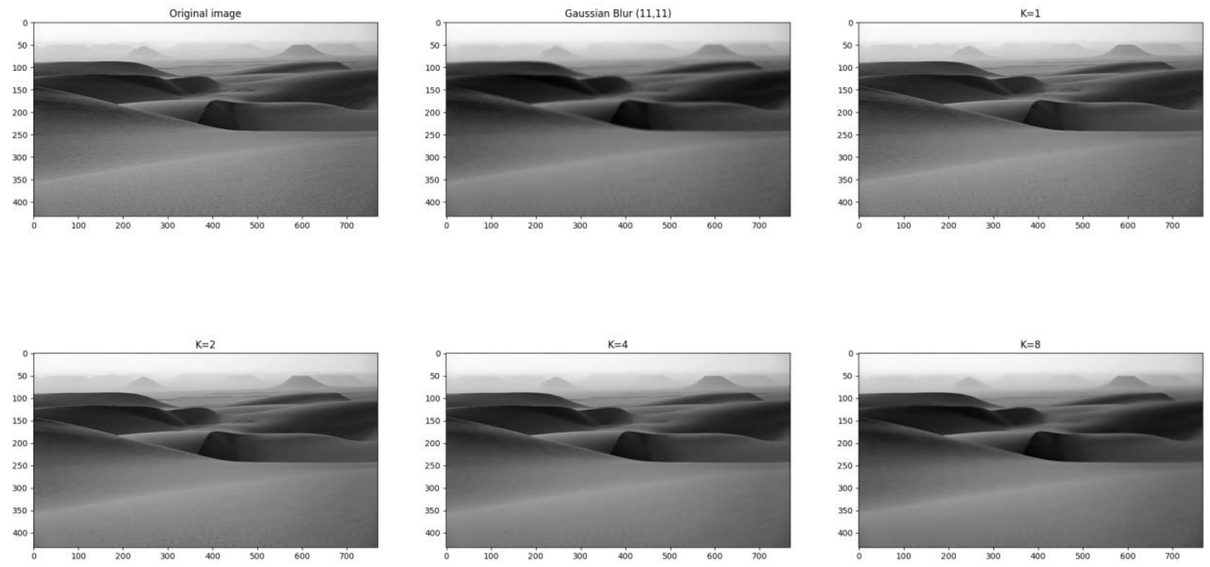
When comparing the difference images, we see that, compared with gauss, anisotropic diffusion does not really operate/blur the edges, as was the task.

Continuing with another example, the laniakea supercluster (a cluster of galaxies), we specifically want to preserve the edges but still smooth noisy areas.

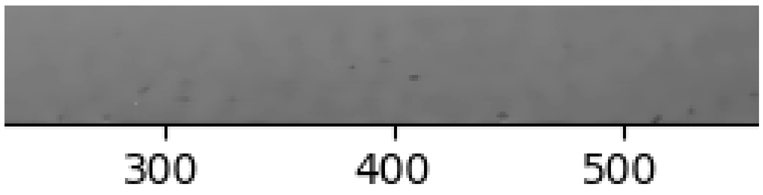


Again, K=4 works great. Here the comparison of edge preservation is rather drastic: K=8 already removes part of the lines.

In another example, one might want to remove the ripples in the sand in a picture of the sahara, while preserving the dune edges:



Here K=4 still produces some artifacts in the lower part of the image, since some ripples are too strong. So a value greater than 4 would be needed here.



All in all, there is no one-fits-all Kappa, as was already mentioned. Therefore it is often necessary to compute several images, or even better, test iteratively. To aid in this process, quantitative metrics, like texture features or signal to noise ratio could be used and adapted based on what effect is to be achieved.