ENSICAEN – Majeure Image – Génération – TP1

You have to use openCV with python or C++ (python preferred).
To install locally python, use the command line:
pip3 install --user opencv-python

Part A – Feature-preserving smoothing

In this part, the 5 images in the folder 'part-a' will be used for testing the methods and illustrating the report. They have different characteristics (gray-scale, color, with and without edges and textures). For each method, you should check if there is a difference for gray-scale images and color onres (usually openCV methods manage both gray-scale and color images).

1) Gaussian blur

Gaussian blur convolves the image by a spatial Gaussian kernel of standard deviation h. This is equivalent to apply a heat diffusion process, with the image as a starting heat map, for a time proportional to h.

https://docs.opencv.org/4.0.1/d4/d86/group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1

TODO. Apply this filter to each image for several values of h.

Obviously all the structures (edges and textures) in the image are blurred indifferently. The removed structures can be observed by computing the difference between the filtered image and the source image. The resulting image is known as the residual image.

TODO. Modify your code to also plot the residual image.

The blurred structures (high values in the residual) are usually those that feature-preserving smoothing methods try to preserve.

2) Bilateral filter

TODO. Test the behavior of the bilateral filter for several values of the parameters (increase only one parameter at a time). The residual images must be computed and plotted.

https://docs.opencv.org/4.0.1/d4/d86/group__imgproc__filter.html#ga9d7064d478c95d60003cf839430737ed

TODO. Filters can be iterated for more smoothing effect. Modify your code to allow to iterate the bilateral filter, and test it for 2 to 5 iterations.

3) NL-means filters

TODO. Test the behavior of the NL-means filter for serveral values of the parameters (increase only one parameter at a time). The residual images must be computed and plotted. Compare the results with the ones obtained by the bilateral filter.

https://docs.opencv.org/4.0.1/d5/d69/tutorial_py_non_local_means.html

for gray-scale images:
https://docs.opencv.org/4.0.1/d1/d79/group__photo__denoise.html#ga4c6b0031f56ea3f98f768881279ffe93
and for color images:
https://docs.opencv.org/4.0.1/d1/d79/group__photo__denoise.html#ga03aa4189fc3e31dafd638d90de335617

4) Guided filtering – Joint Bilateral filter

Feature-preserving smoothing of an image can be guided by another image. This can be done with the joint bilateral filter :

https://docs.opencv.org/4.0.1/da/d17/group__ximgproc__filters.html#ga80b9b58fb85dd069691b709285ab985c

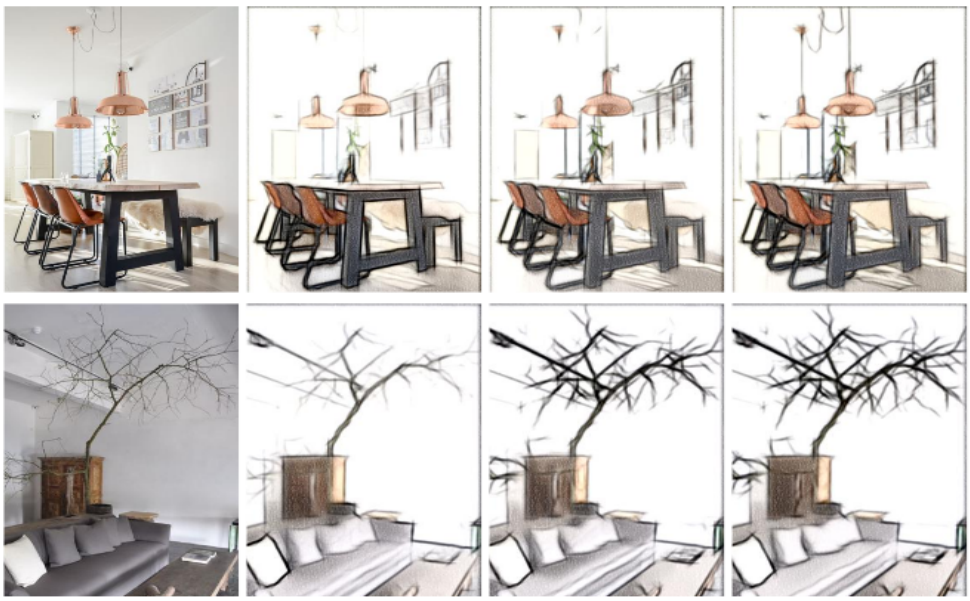
where the parameter joint represents the guide image. We are going to use the edge detector given by the Sobel operator as a guide:

https://docs.opencv.org/4.0.1/d4/d86/group__imgproc__filter.html#gacea54f142e81b6758cb6f375ce782c8d

Usually the source image is smoothed a little (for instance with a Gaussian blur) before an edge detector is applied.

TODO. Test the behavior of joint bilateral filtering as in Section 2 (including the possibiliy to iterate the filter). The residual images must be computed and plotted.

5) Image abstraction



Abstraction effects can be applied to a source image by:
– applying bilateral filtering (or joint bilateral filtering) with large parameters for obtaining an image close to a cartoon image
– applying an edge detector to the source image, or a filtered version (for instance the one obtained in the first step)
– combining both the filtered image and the image of edges. The most simple combination consists to multiply the filtered image by the image representing edges, assuming that its values vary from 1 (filtered image unchanged) and a parameter c (filtered image multiplied by c).

TODO. Code the abstraction effect as a function and test it.

You have finished, you can go to part-b