UNIVERSITÄT TÜBINGEN
Prof. Dr.-Ing. Hendrik P.A. Lensch
Lehrstuhl Computergrafik
Dennis Bukenberger (dennis.bukenberger@uni-tuebingen.de)
Zabiha Kehribar (zabiha.kehribar@student.uni-tuebingen.de)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

21. December 2020

Graphische Datenverarbeitung
Assignment 7

Submission deadline for the exercises: 11. January 2021 6.00 am

Source Code Solutions

- Upload **only** the source code files listed in the description in Ilias.
- Upload them one by one and **don't** zip them.

The source code must run on `cgpool120[0-7].informatik.uni-tuebingen.de` by extracting your submitted `.tar.gz` file to a certain folder and running `scons`. You can log onto this machine via ssh by using your WSI account. From outside the WSI network, you may have to use a ssh gateway, e.g. `cgcontact.informatik.uni-tuebingen.de`
Attention: Do not use `cgcontact` for working, but only for ssh-ing to the `cgpool` machines.

The framework you get for completion already compiles and runs as requested above and you only have to modify source and header files - no files have to be created.

Framework Update

This assignment features a new filter framework. Building works as always; run the code by calling `./filter image.ppm`.
Note that most of the filters will need some time to run through. Printing out status messages in the most outer loops of your implementation is advisable. Have a look at the hint at the top of `main.cpp` for more performance.
Feel free to comment out parts of the main function's code for faster debugging if you don't need it for a specific task, but make sure that all your code is enabled before submission.

You may find more information for your implementation in the lecture's slides as well as in the comments around the functions you have to implement in the code.

7.1 Filter (15 + 15 + 30 = 60 Points)

Implement the filters:

a) Gaussian
Calculate the filtered image exactly, i.e. don't limit the size of the gauss kernel by anything else than the image size. Seperate the filter to obtian decent performance.

b) Median
Compute the median for all color channels individually.
Have a look at the example on this Website for an easy way to sort!

c) Bilateral
The bilateral filter cannot be separated, so its computation is very slow. Restrict the computation of the bilateral filter to three times the size of the Gaussian part of the bilateral filter ($3 \cdot \sigma_g$).
Use the vector distance `Vec3::length()` as distance function between colors.

7.2 À-Trous Wavelet Transform (15 + 10 + 10 + 5 = 40 Points)

a) Implement the À-Trous Wavelet transform.
Refer to the image processing slides, pdf page 94 and following for an overview.

b) Reconstruct the original image by $c = c_N + \sum_{i=0}^{N-1} \alpha_i d_i$. Choosing different values for the α_i should boost or dampen certain image features. Experiment with different values for the α_i (i.e. $\alpha = \{0, 0, 0, ..., 1, 1, 1\}$ or $\{1.5, 1.7, ..., 0.7, 0.1\}$).
Setting all α_i to one should give you the original image.

c) Include the Edge-Stopping function into the À-Trous Wavelet transform.

d) Again experiment with different values for the α_i and for the σ_b in the Edge-Stopping function.

Send some exemplary images for the parts b) and d). Use at least the values $\alpha = \{1, 1, 1, ...\}$ (reconstruction of the original image) and $\alpha = \{0, 0, ..., 0, 1\}$ (largest scaled filter level only) while keeping all other values as they were set in the framework initially.
The reconstructed images are exported as `aTrousTransformedImage.ppm`.