Uros Velimirovic Image Processing for Darwin Digital

The software was developed in visual studio 22 using c++.

Structure:

Program consists of a main abstract class called Image. Four classes are derived from it: OriginalImage, GreyscaleImage, SobelImage and MergedImage. They all use polymorphism by overriding the method startProcessing(). Each class therefore has its own implementation of that method. Main function creates an OriginalImage object. That object creates a grayscale image using luminosity method and then keeps the GreyscaleImage object after saving the image. GreyscaleImage creates SobelImage and so on. In conclusion, these classes use the pipeline mechanism by performing operations one after another, as one modular input-output system. There is also a custom object-oriented shell for STB functions which are provided.

Each class creates and saves images in the pipeline system. That way, the work is distributed evenly.

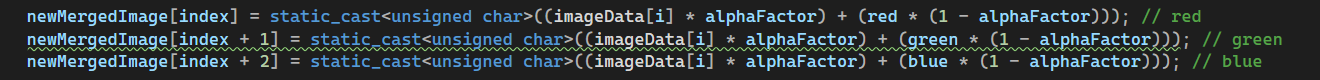
Sobel operator optimization:

Cache line size on most x86\_64 cpus is 64 bytes. Therefore for optimization of sobel algorithm we will use this parameter. Of course, code can be adjusted so that it fits some other machines. We will use x86\_64 architecture info since this architecture is the most common The algorithm is a simple sobel convolution algorithm just adjusted to run faster. The main problem with the basic algorithm is that if the matrix is big enough it will store certain rows in cache and later remove them from cache to insert some others. As processing is going, removed rows will have to be inserted into cache again because they will be processed again for new pixels beneath them. So the solution is instead of going to the end of the matrix row, we will divide the matrix into blocks so that each block contains three rows of certain size + one row from output image matrix. The size of these rows will be calculated as to fit cache the best so that we minimize the miss penalties in cache. This ultimately makes the program run faster. For more details consult the appropriate literature.

Alpha blending:

For final image mergin, alpha blending was used. Alpha factor is a factor which can go from 0 to 1 depending how strong the effect should be.

These formulas were applied while setting alpha factor to 0.5:



They can be found in the code.

Thoughts:

I had great fun while coding this project. I learned something new from digital signal processing and overall i’m very satisfied with it. I hope you enjoy it too.