Image-Processing-cpp

> Author: *Igor Aradski*
>
> igor.aradski@fer.hr

> This project was created as part of the Parallelism and Concurrency course on University of Zagreb, Faculty of Electrical Engineering and Computing.
>
> Parallelization was applied to an existing project Image-Processing-cpp. For more details about the project itself, go to it's GitHub repository.

Parallelization

Due to the simple implementation of the filters, all optimizations were applied to the `for` loops which were used throughout the program.

Since most `for` loops were working on distinctive parts of the image, the image objects were shared between threads, while other variables were private.

In case all threads needed to acces a variable which was changing, `#pragma omp atomic` or `#pragma omp critical` directives were used.

Testing

All testing was performed on a laptop with a AMD Ryzen 5 4600HS CPU with 6 cores. Tests were performed on the `images/big.pgm` image.

Due to some problems and incorrect reading of compressed version (P5) of `.pgm` files in the original project, it's suggested to use uncompressed files (P2) as input.

For semi-automatic testing a custom writen script `timing.sh` was used. It's supposed to be run with the wanted filter as the first argument followed by one or two `.pgm` files and additional arguments if needed.

> E.g. `./timing.sh not images/big.pgm`
>
> `./timing.sh and images/FER_Zagreb.pgm images/big.pgm`

Results of testing can be found in the folder `results` with a file for each filter. Testing was performed by running the script `timing.sh` which runs the filter 10 times and measures the average time spent on image processing. All running times and the average are writen to the file, first for serial and then for the parallelized version of the program.

Resulting images are saved to the `filtered` folder. Images with suffix `_0` were created using the serial, while those with suffix `_1` were creating running the parallelized version.

Results

While the time of image processing was reduced, further optimizations would make that effect even more visible. Below is the table of some of the results. All results can be found in the `results` folder. All times are in milliseconds (ms).

Filter	Original	Parallelized
not	0.0379	0.0144
otsuBinarize	0.0723	0.0429
and	0.180	0.109
smoothingFilter	0.452	0.137
edgeDetect	0.481	0.151

As visible, time of image processing was reduced by roughly the factor of 2 on "simpler" filters, and around the factor of 3 for more complex filters.

Even though the image processing was optimized, file reading and writing takes a bigger part of total program running time.

Profiling analysis results using gprof are visible in `analysis.txt` and `analysis_serial.txt`.