Fakultät für Informatik und Mathematik
Dr. Armin Größlinger
Summer semester 2025

Exercises for Lecture: Practical Parallel Programming
Project 2 (Image Processing: VCD, Sobel)

Problem 1 (Project Image Processing) [25+5 Points]

In this project, we want to implement two image processing operators: VCD and Sobel. The VCD operator reduces patchy noise without affecting the sharpness of the image too much, Sobel is a classical edge detection operation.

In `/scratch/ppp2025/2` you can find the project skeleton and some example images. The tool `pnmdiff` (which is included in the project skeleton) can be used to compare two PGM images pixel by pixel.

The skeleton contains sequential implementations of the operators in `single.c`. The sequential implementation is activated by the switch `-n` (without `-n`, the parallel implementation in `parallel.c` is used).

(a) Implement the Sobel operator using OpenMP parallelism. [2 Points]

(b) Implement the VCD operator using OpenMP parallelism. [4 Points]

(c) Add MPI parallelism to the implementation of both operators. A simple 1-dimensional row-wise distribution of the image onto the MPI processes suffices. Note that there is some overlap between the data used by neighboring processes. [6 Points]

(d) Some computations can be made locally in each MPI process without the data communicated between the processes after each iteration of the VCD operator. Structure the MPI parallelism in such a way that the local computations are performed while communications are running. [4 Points]

(e) Due to the dependence on neighboring pixel values, one has to exchange data between neighboring MPI processes (so-called overlap or "ghost" rows). There is a trade-off between the frequency of the communication and performing redundant computations. In general, one can exchange g rows after every g iterations of the VCD operator ($g \geq 1$). The g exchanged rows allow an MPI process to compute the values of the ghost rows needed for the next $g-1$ iterations locally (without communication). $g = 1$ is the case with row exchanges after each iteration (and no redundant computations). Extend your implementation such that $g > 1$ is supported. [4 Points]

(f) The VCD operator performs some computations twice at a pixel and at one of its neighbors: two ϕ and two ξ function values are the same as the values computed at certain neighboring pixels. Avoid (at least) 3 of these 4 redundant computations by caching the values, i.e., storing the values in temporary storage and using these precomputed values in the computation for another pixel. Reuse only happens within an MPI process (i.e., the communication structure is not changed). [5 Points]

(g) (Only for Master:) Eliminate all 4 redundant computations in the VCD operator (by caching). [5 Points]

Note: In subtasks (f) and (g), do *not* use a quadratic amount of additional memory (in the order of the number of rows times the number of columns), use a linear amount (in the order of the number of rows *or* columns) at most.

Test your program with the image `skull.pgm` and the standard parameter values $N = 40, \epsilon = 0.005, \kappa = 30, \delta t = 0.1$ for the VCD operator and $c = 0.9$ for Sobel. For benchmarks, you can also use the very

big image `world.pgm`; you may need to set the parameter `--mem=...` in the `srun`/`salloc` invocation to reserve a sufficient amount of RAM for your job (e.g., `srun --mem=4g ...` to request 4 GiB RAM). When the program is invoked with option `-d` (to monitor the progress of the VCD operator), pass the option `--unbuffered` to `srun` to turn of buffering of output.

Upload your solution (file `parallel.c` only) in ILIAS. Do not miss the deadline! Late submissions cannot be accepted due to legal regulations!

Do not forget to register in the Campus Portal in time if you want to obtain a grade for PPP!

VCD Operator (VCD = Variable Conductance Diffusion):

The VCD operator has the parameters $N, \epsilon, \kappa, \delta t$. The operation (on the whole image) is complete when the following procedure has been applied N times or when on every *inner* pixel (i.e., a pixel not at the border of the image) $|\Delta(x,y)| \leq \epsilon$ holds. In every of the up to N iterations, each pixel $s(x,y)$ is assigned a new value $t(x,y)$ according to the following formula:

$$
\begin{aligned}
t(x,y) &= s(x,y) + \kappa \cdot \delta t \cdot \Delta(x,y) \\
\Delta(x,y) &= \quad \phi\big(s(x+1,y) - s(x,y)\big) - \phi\big(s(x,y) - s(x-1,y)\big) \\
&\quad + \phi\big(s(x,y+1) - s(x,y)\big) - \phi\big(s(x,y) - s(x,y-1)\big) \\
&\quad + \xi\big(s(x+1,y+1) - s(x,y)\big) - \xi\big(s(x,y) - s(x-1,y-1)\big) \\
&\quad + \xi\big(s(x-1,y+1) - s(x,y)\big) - \xi\big(s(x,y) - s(x+1,y-1)\big) \\
\phi(\nu) &= \chi \cdot e^{\frac{-\chi^2}{2}} \qquad \text{mit } \chi = \frac{\nu}{\kappa} \\
\xi(\nu) &= \frac{1}{\sqrt{2}} \cdot \psi \cdot e^{-\frac{\psi^2}{2}} \quad \text{mit } \psi = \frac{\nu}{\sqrt{2} \cdot \kappa}
\end{aligned}
$$

The pixel values are in the range $[0,1]$ (not $[0,255]$), i.e., the greyscale values from the PGM image have to be scaled to the interval $[0,1]$ first. Perform the computations in the whole VCD operation with `double` precision. The exponential function e^x can be computed by `exp(x)` in C, see also `man exp`.

Note that pixels values outside the interval $[0,1]$ can arise. Such values have to be set to 0 or `maxcolor`, respectively, when the image is saved.

Sobel Operator:

The Sobel operator associates to each pixel $s(x,y)$ a new value $t(x,y)$ according to the following formula:

$$
\begin{aligned}
s_x(x,y) = \quad & s(x-1,y-1) + 2 \cdot s(x,y-1) + s(x+1,y-1) \\
& -s(x-1,y+1) - 2 \cdot s(x,y+1) - s(x+1,y+1) \\
s_y(x,y) = \quad & s(x-1,y-1) + 2 \cdot s(x-1,y) + s(x-1,y+1) \\
& -s(x+1,y-1) - 2 \cdot s(x+1,y) - s(x+1,y+1) \\
t(x,y) = \quad & \sqrt{s_x(x,y)^2 + s_y(x,y)^2}
\end{aligned}
$$

For both the operators, one sets $s(x,y) = 0$ for pixels outside the image.

Submission due: Wednesday, June 11, 2025 at 18:00 via ILIAS