Programming Parallel Computers

Aalto 2023

MF: median filter

Please read the general instructions on this page first, and then check the individual tasks for more details. For each task you can download a zip file that contains the code templates you can use for development.

Task	Attempts	Expected	Points	Max	Rating	Rec.	Deadline for full points
MF1: CPU baseline Implement a simple **sequential** baseline solution. Make sure it works correctly. Do not try to use any form of parallelism yet. You are expected to use a naive algorithm that computes the median separately for each pixel, with a **linear-time median-finding algorithm**.	0	–	–	5	★	R	2023-04-30 at 23:59:59
MF2: multicore parallelism Parallelize your solution to MF1 with the help of **OpenMP** so that you are exploiting multiple CPU cores in parallel.	0	–	–	3	★	R	2023-05-07 at 23:59:59
MF9a: better algorithm Design a **better algorithm** that does not recalculate the median separately for each pixel. Make it as efficient as possible, also for very large window sizes. You are encouraged to use all resources that you have in the CPU.	0	–	–	5	★★★		2023-06-04 at 23:59:59

General instructions for this exercise

In this task, we will implement a program for doing 2-dimensional median filtering with a rectangular window.

Interface

You need to implement the following function:

```
void mf(int ny, int nx, int hy, int hx, const float* in, float* out)
```

Here `in` and `out` are pointers to `float` arrays, each of them contains `ny*nx` elements. The arrays are already allocated by whoever calls this function; you do not need to do any memory management.

The original value of pixel (x,y) is given in `in[x + nx*y]` and its new value will be stored in `out[x + nx*y]`.

Correct output

In the output, pixel (x,y) will contain the median of all pixels with coordinates (i,j) where

- `0 <= i < nx`,
- `0 <= j < ny`,
- `x - hx <= i <= x + hx`,
- `y - hy <= j <= y + hy`.

This is the sliding window. Note that the window will contain at most `(2*hx+1) * (2*hy+1)` pixels, but near the boundaries there will be fewer pixels.

Details

For our purposes, the median of the vector $\mathbf{a} = (a_0, a_1, \ldots a_{n})$ is defined as follows:

- Let $v_0, v_1, \ldots v_n$ be the values of $\mathbf{a}$ sorted in a non-decreasing order.
- If $n = 2k + 1$, then the median of $\mathbf{a}$ is v_{k+1}.
- If $n = 2k$, then the median of $\mathbf{a}$ is $(v_k + v_{k+1})/2$.

Examples

These examples show what a correct implementation will do if you apply it to each color component of a bitmap image. In these examples, for each color component, the value of each pixel is the median of all pixel values within a sliding window of dimensions (2k+1) × (2k+1). Hover the mouse on the output images to see the differences between input and output.

k = 1

k = 2

k = 5

k = 10

Noise reduction, k = 1

Noise reduction, k = 2

Hints for MF1–MF2

Hints for MF9a