Programming Parallel Computers

Aalto 2023

Pre: prerequisite test

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
Pre0: prerequisite test In this task you do not need to worry much about the performance. The most straightforward implementation that you can imagine should be fast enough to meet the time limits. No specific techniques are required and you are not expected to use any form of parallelism yet; any solution that works correctly is fine.	0	–	–	1	★	R	2023-04-28 at 23:59:59

General instructions for this exercise

You need to write a function that takes as input a bitmap image and the coordinates of a rectangle, and it has to calculate the **average color** of all pixels inside the rectangle.

Interface

We have already defined the following type for storing the result:

```
struct Result {
    float avg[3];
};
```

You need to implement the following function:

```
Result calculate(int ny, int nx, const float *data,
                 int y0, int x0, int y1, int x1)
```

Here `data` is a color image with `ny*nx` pixels, and each pixel consists of three color components, red, green, and blue. In total, there are `ny*nx*3` floating point numbers in the array `data`.

The color components are numbered `0 <= c < 3`, x coordinates are numbered `0 <= x < nx`, y coordinates are numbered `0 <= y < ny`, and the value of this color component is stored in `data[c + 3 * x + 3 * nx * y]`.

The parameters `y0`, `x0`, `y1`, and `x1` indicate the **location** of the rectangle. The upper left corner of the rectangle is at coordinates (`x0`, `y0`), and the lower right corner is at coordinates (`x1-1`, `y1-1`). That is, the width of the rectangle is `x1-x0` pixels and the height is `y1-y0` pixels. The coordinates satisfy `0 <= y0 < y1 <= ny` and `0 <= x0 < x1 <= nx`.

In the result that you return, `avg[c]` has to contain the arithmetic mean of the color component `c` for all pixels inside the rectangle.

Details

Even though the input and output are single-precision floating-point numbers, you must do **all arithmetic with double-precision floating point numbers**, and only round the final result back to single precision.

You can assume that there are at most 10 million pixels in the input image.