Interpolation of an image (multi-threaded)

1. Source image enlargement:

* When interpolate pixels around the border, the area outside the image may need to be sampled. These could be solved largely in two ways. But I would like to place some of my analysis for this.

1st Method : When sampling the area outside the image, copy the pixel from the border.

2nd Method : First, pad some colors around the border first. (plus, the source image could be enlarged so the indexing may be more straightforward.) Then just interpolate over the padded area.

I chose the 2nd method for the reason stated below.

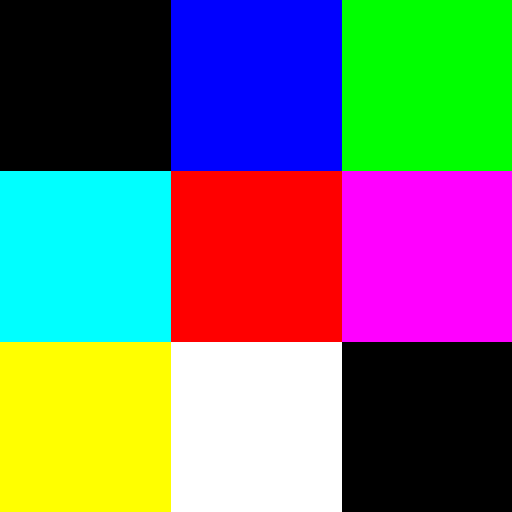
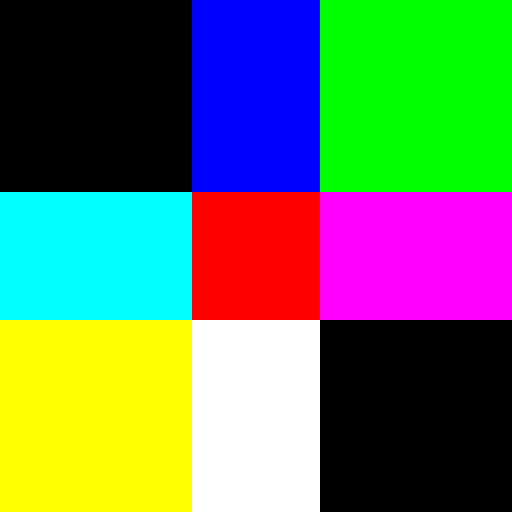
* Although the 2nd method adds extra computation, it removes a few if statements that is used over the destination image, which is usually larger.
* **(Theoretical) performance comparison** between different design choices.

1st method adds the computation :

2nd method adds the computation :

Typically, a destination image is larger than the source image. Therefore,

Thus, I chose the 2nd method.

**Figure 1 – Source image (left). Expanded source image (middle). Filtered image (right).**

1. Thread management:

* Std::thread – I used STL library thread.
* I implemented asynchronous join of threads myself.
  + Because STL thread lacks asynchronous join of threads, I coded it in ThreadManager class myself.
    - You may notice one global variable for that. I almost never use it especially in a threaded environment. However, implementing asynchronous joining may involve communication between threads. One global might not matter for this scope of projects. So I went with that.

1. Performance Data

All the data is generated using a notebook computer with i7-4700HQ 2.4Ghz and Geforce GTX850M.

* Below is elapsed time to generate an image of size (16384 X 16384) with a different number of threads.

(Unit : seconds)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No of Thread | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  | 43.9 | 22.348 | 15.396 | 12.468 | 10.844 | 9.593 | 9.232 | 8.798 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No of Thread | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|  | 8.761 | 8.763 | 8.674 | 8.722 | 9.284 | 8.878 | 8.754 | 8.671 |

* Below is elapsed time to generate a different size of images with 8 threads. (Image size bigger than (16384 X 16384) will exceed the allowable array size by my C compiler.)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Image Size | (16,16) | (32,32) | (64,64) | (128,128) | (256,256) | (512,512) |
|  | 0.015000 | 0.016000 | 0.016000 | 0.047000 | 0.093000 | 0.187000 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Image Size | (1024,1024) | (2048,2048) | (4096,4096) | (8192,8192) | (16384,16384) |  |
|  | 0.359000 | 0.750000 | 1.609000 | 2.943000 | 8.470000 |  |