Parallel Visual Computing 2012/13
Assignment 2

Every 2 weeks you can achieve a maximum (saturated) number of 20 points. The minimum to pass is 10 points in each assignment.

Mandatory Task

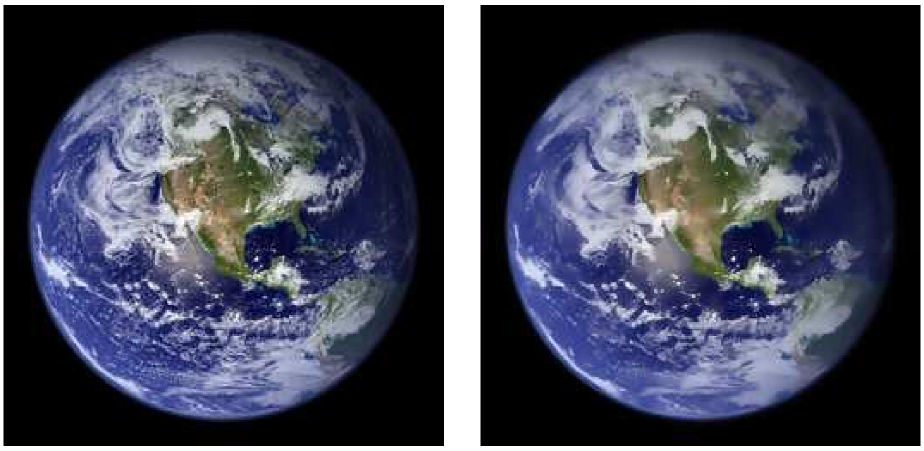

Figure 1: Example of an image (earth_small.ppm) filtered with the bilateral filter. Left: original, Right: Filtered Image, $\sigma_d = 2$, $\sigma_r = 0.1$.

Bilateral Filter Implement and improve the performance of the Bilateral Filter [TM98]. The effect of this filter is that of a Gaussian filter in image regions of similar color. However, when there are strong edges, these are not blurred. An example is shown in Fig. 1. The definition of the filter is given by

$$\tilde{I}(x, y; \sigma_d, \sigma_r) = \frac{1}{w(x, y; \sigma_d, \sigma_r)} \cdot \iint I(x + \hat{x}, y + \hat{y}) \cdot k_d(\hat{x}, \hat{y}; \sigma_d) \cdot k_r(||I(x, y) - I(x + \hat{x}, y + \hat{y})||; \sigma_r) d\hat{x} d\hat{y}.$$

where I is the original image, and $\tilde{I}$ the filtered image. The semicolon ";" denotes a split between the parameters of the function (e.g. filtered image) and the parameters of the filter. The filter consists of a kernel that depends on a standard spatial Gaussian kernel

$$k_d(x, y; \sigma_d) = 1/2\pi\sigma_d^2 \cdot \exp^{-(x^2+y^2)/(2\sigma_d^2)}$$

as in the last assignment (σ_d controls the amount of spatial blur), and a modulation term that depends on the intensity difference between the center point of the kernel and the current integration position

$$k_r(d; \sigma_d) = 1/\sqrt{2\pi}\sigma_d \cdot \exp^{-(d^2)/(2\sigma_d^2)}\,.$$

Here d is the intensity difference between the two points in the image and σ_r controls the edge sensitivity of the filter. The weight w normalizes the filter and preserves the intensity scale of the image

$$w(x, y; \sigma_d, \sigma_r) = \iint k_d(\hat{x}, \hat{y}; \sigma_d) k_r(||I(x, y) - I(x + \hat{x}, y + \hat{y})||; \sigma_r) d\hat{x} d\hat{y}.$$

15 Points

Hint: You can might be able to re-use the Gaussian filter implementation.

Documentation

- Document the performance of your initial implementation ($\sigma_d \in \{2.0, 3.0, 4.0\}, \sigma_r \in \{0.1, 0.2, 0.3\}$) (choose the kernel half size such that the full kernel is covered).
- Document reasons for and the performance improvements of your individual optimization strategies (one choice of parameters is sufficient).
- Document your final solution in the same way as the initial implementation.

5 Points

Optional Tasks

Each item yields **5 Points** if properly documented (see task documentation).

- For the separable convolution of assignment 1, change the memory arrangement of the vertical convolution such that continuous data access is possible.

- Implement a Fourier Transform-based convolution (you may use a library like fftw), compare its performance with the pre-computed 2D kernel version of the algorithm and with separable convolution for different kernel sizes. Should a smart convolution algorithm decide dynamically which version to use ? Is this point of trade-off the same for different machines ?
- Implement a complete convolution loop of your choice completely in assembler - can you beat the compiler ?
- Perform a comparison to "Intel Integrated Performance Primitives" and "OpenCV". How close are we to their performance ?

References

[TM98] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839 –846, 1998.