“Implementation of Bilateral Filtering on CUDA”

1Ashish A.Deshmukh, 2Sanjay L. Badjate
1-2Electronics Engg, S.B.J.I.T.M & R, Nagpur, India

Abstract—The bilateral filter is a non-linear technique that can blur an image while respecting strong edges. Image filtering is a term which is very common and known to all. The world today is of high definition imaging, so it is very necessary that we have an accurate and fast method of Image filtering. From long time MATLAB has been used for Image Processing. But MATLAB takes a long time to filter an image of larger resolution as it uses CPU processor for processing. So to solve this we will be implementing chosen Image Filtering technique on NVIDIA’s Graphics Card using NVIDIA’s tool for parallel processing known as CUDA. The execution will be performed on a Graphical Processing Unit (GPU) with many processing cores, unlike the conventional method of executing on a Central Processing Unit (CPU) with few processing cores. This GPU implementation will be validated with the results obtained from a CPU implementation, followed by comparing the execution times of the CPU and the GPU. For the above comparisons, the CPU used will be Intel ® E5300, a low cost dual core CPU and NVIDIA ® GT610, which is again a low cost GPU.

Keywords—Bilateral Filtering, GPGPU (General Purpose Computing for Graphics Processing Unit),CUDA (Compute Unified Device Architecture),

I. INTRODUCTION

The development in the VLSI technology has made the availability of low cost computing hardware. However, the area of scientific computing is still facing a deficiency of low cost computing platform. One of the recent solutions to this problem is GPGPU computing, that is General Purpose (computing on the) Graphics Processing unit. Consider the following example, a NVIDIA GeForce 520M consists of 48 processing cores, with a clock speed of 1480 MHz. This GPU is available at a reasonable cost, yet is immensely powerful in terms of computational capability. However, this multicore parallel computing architecture is useless unless the software algorithm also supports parallelism. NVIDIA provides a free SDK CUDA for such software development. CUDA is the name of NVIDIA’s parallel computing architecture in our GPUs. NVIDIA provides a complete toolkit for programming the CUDA architecture that includes the compiler, debugger, profiler, libraries and other information. Developers need to deliver Quality products that use the CUDA architecture. The CUDA architecture also supports standard languages such as C and Fortran and APIs for GPU Computing, such as Open CL and Direct Compute. Image Processing is one such field, where the time required for execution on Images with lots of data or larger size images is very high. It typically involves large computations. Community of users is wide and includes not only common customers with digital cameras, but also astronomers analyzing data from telescopes and space crafts, specialists dealing with magnetic resonance imaging (MRI) or computed tomography, and biologists working with optical microscopes. The users demands for image quality, in terms of resolution, signal-to-noise ratio or dynamic range, are constantly growing. As a result, both the image acquisition devices and image processing algorithms are developing in order to satisfy the users’ needs. On the other hand, the complex image processing methods can be very time consuming and their computation can be challenging even for recent computer hardware. Therefore, an integral part of the field is optimization, so that the complex algorithms could be performed in a reasonable time. For example, if a patient undergoes an magnetic resonance or ultrasound investigation, it is essential to have the data processed in several minutes so that a specialist can consult the results with the patient and propose next steps within a single visit. In other applications like user-assisted image segmentation, the data need to be processed real-time. The most important computational hardware for image processing is Central Processing Units (CPU), for their versatility and tremendous speed. According to the oft-quoted Moore’s Law, the CPU’s computing capability is doubled every two years. However, this trend is now inhibited as the semiconductor industry has reached physical limits in production. Soon, it will be no longer possible to decrease the size of transistors, and therefore to increase the clock frequency. Thus, manufacturers seek for a different ways to enhance the CPU speed. The most significant trend is to increase CPU’s ability to process more and more tasks in parallel, by extending the instruction pipeline, involving vector processing units, and increasing the number of CPU cores.

In contrast to CPU, Graphics Processing Units (GPU) are recently considered to be very efficient parallel processing units. While their original purpose was to provide a graphic output to users, their computing capabilities have been getting more and more complex
and their performance has overcome common CPUs, in terms of both computing speed (in floating point operations per second) and memory bandwidth. Besides, their performance growth is not limited by current manufacturing technology, so the gulf between CPU and GPU computational power is still getting wider. The GPU architecture benefit from a massive fine-grained parallelization, as they are able to execute as many as thousands of threads concurrently. Attempts to utilize GPUs not only for manipulating computer graphics, but also for different purposes, lead to a new research field, called “General-purpose computation on GPU (GPGPU”).

II. OVERVIEW ON CUDA

NVIDIA developed a parallel computing platform and programming model called as CUDA. By harnessing the power of the graphics processing unit (GPU) it enables dramatic increases in computing performance.

With millions of CUDA-enabled GPUs sold to date, researchers, scientists and software developers are finding broad-ranging uses for GPU computing with CUDA. Some examples are

Analyze air traffic flow: The National Airspace System manages the nationwide coordination of air traffic flow. Computer models help keep airplane traffic moving efficiently congestion and identify new ways to alleviate. By using the computational power of GPUs, a large performance gain, reducing analysis time from ten minutes to three seconds is obtained by a team at NASA.

Identify hidden plaque in arteries: Heart attacks are the leading cause of death worldwide. Harvard Medical School, Brigham & Harvard Engineering, & have teamed up to use GPUs to identify hidden arterial plaque and simulate blood flow without exploratory surgery or invasive imaging techniques.

Visualize molecules: A molecular simulation (nanoscale molecular dynamics) called NAMD obtained a large performance boost with GPUs. The parallel architecture of GPUs results in a the speed-up, which enables NAMD developers to port compute-intensive portions of the application to the GPU using the CUDA Toolkit.

GPU COMPUTING: THE REVOLUTION

Using CUDA no assembly language required, you can send Fortran code and C, C++ codes straight to GPU.

Developers at companies such as, Wolfram Research, ANSYS, Autodesk, Math Works and Adobe are waking that sleeping giant the GPU to do engineering computing and general-purpose scientific across a range of platforms.

By using high-level languages, for single-threaded performance GPU-accelerated applications run the sequential part of their workload on the CPU. Which is optimized while accelerating parallel processing on the GPU called as “GPU computing????”

GPU does much more than render graphics therefore GPU computing is possible today it sizzles with a teraflop of crunches application tasks and floating point performance. it is designed for anything from medicine to finance.

CUDA is widely deployed through thousands of applications and published research papers and supported by an installed base of over 375 million CUDA-enabled GPUs in notebooks, compute clusters workstations, and supercomputers.

III. BILATERAL FILTERING

The Bilateral using CUDA enhanced parallel computations. The Bilateral filter allows smoothing images, while preserving edges, in contrast to e.g. the Gaussian filter, which smoothes across edges. While delivering visually stunning results, Bilateral filtering is a costly operation. Using NVidia's CUDA technology the filter can be parallelized to run on the GPU, which allows for fast execution, even for high definition images. The Bilateral Filter smoothes surfaces, just as the Gaussian Filter, while maintaining sharp edges in the image. The idea of implementing bilateral filtering is to work in the range of an image what traditional filters do in range of domain. Two pixels are close to one another, that is, gaining nearby spatial location, or they can be same to one another, that is, have nearby values, possibly same in a perceptually meaningful way. Closeness refers to vicinity in the domain, similarity to vicinity in the range. In old days filtering is domain filtering, and enforces closeness by weighing of the pixel values with coefficients that fall in with distance.

The bilateral filter is defined as:-

$$f_{\text{filtered}}(x) = \sum_{\Omega(x)} I(x_i)f_i(||I(x_i) - I(x)||)g_k(||x_i - x||)$$

f_{\text{filtered}} is the filtered image.

I is original input image to be filtered.

X are coordinates of current pixel to be filtered.

Ω is window centered in X.

f_i is range kernel for smoothing differences in intensities.

g_k is spatial kernel for smoothing differences in coordinates.
IV. EXPERIMENTS WITH BLACK AND WHITE IMAGES

In this section on black-and-white images we analyze the performance of bilateral filters. Figure 1 (a) and 1(b) for the removal of texture. Some amount of gray-level quantization can be seen in figure 1 (b), but this is due to the printing process, not by the filter.

![Figure 1 (a) and 1(b)]

Notice that the kitten’s whiskers, much thinner than the filter’s

The picture “simplification” 5 (b) can be useful for data reduction without loss of overall shape features in applications such as image picture editing, manipulation and transmission, image description for retrieval. Figure 2 shows the effect of different values of the parameter $\sigma_d$ and $\sigma_r$ on the resulting image. (10 or 30), range filtering dominates perceptually because it preserves edges.

![Figure 2]

Bilateral filtering with parameters $\sigma_d=3$ pixels and $\sigma_r=50$=intensity values is applied to the image in figure 3 (a) to yield the image in figure 3 (b). We must note that most of the fine texture has been removed, and still all contours are as crisp as in the original one.

![Figure 3 (a) and 3 (b)]

In terms of computational cost, the bilateral filter is twice as expensive as a no separable domain filter of the same size. A simple trick that decreases computation cost considerably is to precompute all values for the similarity functions.

V. EXPERIMENTS WITH COLOR IMAGES

For black-and-white images, levels of gray are generated across edges, leading to production of blurred images. With color images, an additional problem arises from the fact that between any two colors there are other, often rather different

The filtered image does not just look blurred, it also consists of odd-looking, colored auras around objects. Figure 4 (a) in the color plates shows a detail from a picture with a red jacket in front of a blue sky. For example, between blue and red there are various shades of pink and purple. In fact, pixels along the boundary, when projected back into the scene, intersect both blue sky and red jacket, and the resulting color is the average of red and blue i.e. pink. When smoothing, this effect is emphasized, as the broad, blurred purple-pink area in figure 4 (b) shows. Thus, disturbing color bands may be produced when smoothing across color edges. Even in this clear picture, a thin pink-purple line is visible, and is due to a combination of pixel averaging and lens blurring.
To address this difficulty, edge-preserving smoothing could be applied to the RGB components of the image separately. However, the intensity profiles across the edge in the three color bands are in general different. Separate smoothing results in an even more pronounced pink-purple band than in the original image, as shown in figure 4 (c). The pink-purple band, however, is not widened as it is in the standard-blurred version of figure 4 (b).

A much better output can be obtained with bilateral filtering. In fact, a bilateral filter allows mixing the three color bands appropriately, and measuring photometric distances between pixels in the combined space. Also this combined distance can be made to correspond closely to perceived dissimilarity by using Euclidean distance in the CIE-Lab color space. Figure 4 (d) shows the output image from bilateral smoothing of the image in figure 4 (a). The pink band has shrunk considerably, and no extraneous colors appear. Figure 5 (c) in the color plates shows the result of five iterations of bilateral filtering of the image in figure 5 (a).

While a single iteration produces a much cleaner image (figure 5 (b)) than the original image, and is likely to be sufficient for most image filtering needs, multiple iterations have the effect of flattening the colors in an image considerably, but without blurring edges. The output image has a much reduced color map, and the effects of bilateral filtering are simpler to see when printout is taken out. Note that the cartoon-like appearance of figure 5 (c). All edges and shadows are preserved, but most of the shading is gone, and no “new” colors are introduced by filtering process.

VI. CONCLUSION

In this paper we have introduced the concept of bilateral filtering for edge-preserving smoothing. The liability of bilateral filtering is analogous to that of old filtering techniques, which we called domain filtering in this paper. The explicit enforcement of a photometric distance in the range component of a bilateral filter makes it possible to filter color images in a perceptually appropriate manner.

Just as the parameters of domain filters depend on image properties and on the intended result, so do those of bilateral filters. This is however a consequence of the
generality of this technique. Given a specific application, techniques for the automatic design of filter profiles and parameter values may be possible.

VII. FUTURE SCOPE

The bilateral filtering for image processing takes a considerable amount of time to process an image with lots of information, to reduce this computation time this technique will be implemented on GPU's. GPGPU leads to a huge amount of reduction in computation/processing time. The Bilateral Filter has been implemented using CUDA, with a large speed gain. For a naïve transfer from sequential to parallel, this yields an improvement of just above 100 times in running time. Optimizing the code to run more efficiently on the GPU pays off, with the final result running at more than a thousand times faster than the sequential implementation. This allows 20 images per second at full HD for small image manipulations.

So here we are performing Bilateral filtering using MATLAB Tool And then moving to do filtering on NVIDIA's Graphic Card using NVIDIA's tool for parallel processing known CUDA to reduce computational time.

REFERENCES


