Brook+ SAMPLE

Image Filter

1 Introduction

Filtering is a common task in image processing, typically used to apply an effect such as blurring or extract useful image characteristics like edges. A *filtering mask* is a 2D matrix of “weights” that can be combined with an input image using some operator. A commonly used operator in image processing is *convolution*. The result of convolution is a new image, where each pixel is a weighted sum of the corresponding pixel in the original image and its neighboring pixels, the weights being specified by the filtering mask. To perform convolution, the center of the filtering mask is placed over each pixel in the input image, and a sum of the products of corresponding pixels is computed to obtain the value of the new pixel in the output image. Note that only filtering masks with odd dimensions have a well-defined “center”; thus, only these are considered valid inputs to a convolution operation.

Define an input image **I**(m x n) and a filtering mask **C**(r x r) and an output image **I’**(m x n). For mask **C**, imagine that:

- the center of the mask **C** is at **C**(0,0),
- the upper left corner is at **C**(-⌊ r/2⌋ , -⌊ r/2⌋), and
- the bottom right corner is at **C**(+⌊r/2⌋ ,+⌊ r/2⌋).

Then, the output pixel **I’**(i,j) can be computed using the convolution operator using the following expression:

$$\mathbf{I'}(i,j) = \sum_{P=-\lfloor r/2 \rfloor}^{-\lfloor r/2 \rfloor} \sum_{q=-\lfloor r/2 \rfloor}^{-\lfloor r/2 \rfloor} \mathbf{C}(p, q) * \mathbf{I}(i+p, j+q)$$

Figure 1 shows an example of a 5x5 Gaussian filter and a sample image blurred using this filter.

1/100

0.078633	0.655965	1.330373	0.655965	0.078633
0.655965	5.472157	11.098164	5.472157	0.655965
1.330373	11.098164	22.508352	11.098164	1.330373
0.655965	5.472157	11.098164	5.472157	0.655965
0.078633	0.655965	1.330373	0.655965	0.078633

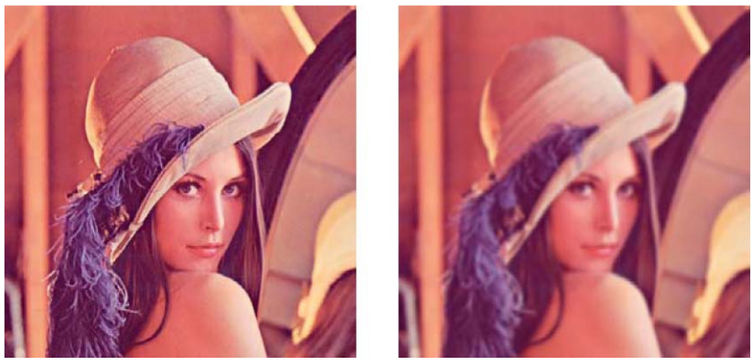

Figure 1 **A 5x5 Gaussian Blur Filter; Left is Input Image, Right is Output Image WithGaussian Blur Filter Applied**

2 Image Filtering on Brook+

In image filtering, each pixel in the output image is a function of the corresponding pixel in the input image and its surrounding pixels. The computation is localized to a neighborhood. Moreover, the same set of arithmetic operations is applied to each pixel in the input image. The data-independent and repetitive nature of the task makes image processing favorable for implementation on an ATI Stream processor. We can achieve a massive computational acceleration by unlocking the high rate of inherent data-parallelism.

We illustrate a simple image processing task in this Brook+ sample. A 3x3 filtering mask is applied to an input image. A simple Brook+ kernel performs the convolution. The inputs to the Brook kernel are: the input image bound as an array, and the 3x3 filtering mask. For each pixel in the output image, the kernel gathers the surrounding pixels in the input image, applies the mask, and accumulates the result. Static offsets are used (instead of a loop) to traverse the 3x3 region around each pixel.

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For Stream Computing:
URL: www.amd.com/stream
Questions: streamcomputing@amd.com
Developing: streamdeveloper@amd.com
Forum: www.amd.com/streamdevforum