AMD **Accelerated** Parallel Processing
TECHNOLOGY

SAMPLE

Unsharp Mask Filter

1 Overview

1.1 Location

`$<AMDAPPSDKSamplesInstallPath>\samples\opencl\cpp_cl\1.x`

1.2 How to Run

See the *Getting Started* guide for how to build samples. You first must compile the sample.

Use the command line to change to the directory where the executable is located. The pre-compiled sample executable is at `$<AMDAPPSDKSamplesInstallPath>\samples\opencl\bin\x86\` for 32-bit builds, and `$<AMDAPPSDKSamplesInstallPath>\samples\opencl\bin\x86_64\` for 64-bit builds.

Type the following command(s).

1. `UnsharpMask`
   This command generates the sharpened image of the input image file.
2. `UnsharpMask -h`
   This command prints the help file.

1.3 Command Line Options

Table 1 lists, and briefly describes, the command line options.

Table 1 Command Line Options

Short Form	Long Form	Description
`-h`	`--help`	Shows all command options and their respective meanings.
`-q`	`--quiet`	Quiet mode. Suppresses all text output.
`-e`	`--verify`	Verify results against reference implementation.
`-r`	`--radius`	Specify, in pixels, the area around the pixel in terms of the radius.
	`--threshold`	Specify the minimum difference, in levels, between the pixels
`-a`	`--amount`	Specify, as a percentage, the amount of brightness or darkness
`-g`	`--gui`	Display the sharpened image.
`-i`	`--iterations`	Number of iterations for kernel execution.
`-v`	`--version`	AMD APP SDK version string.
	`--dImage`	Disable OpenCL image buffers. Use normal OpenCL buffers.

2 Introduction

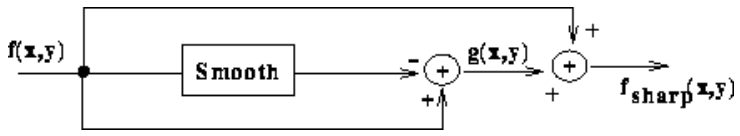
Unsharp Masking (USM) is an image manipulation technique, often available in digital image processing software.

The "unsharp" in the name derives from the blurred, or "unsharp," positive image that the technique uses to create a "mask" of the original image. The un-sharped mask is then combined with the negative image, creating an image that is less blurry than the original. The resulting image, although clearer, probably loses accuracy with respect to the image's subject. In the context of signal-processing, an unsharp mask is generally a linear or nonlinear filter that amplifies high-frequency components.

Typically, three settings control digital unsharp masking:

- `Amount (-a)` is listed as a percentage, and controls the magnitude of each overshoot (how much darker and how much lighter the edge borders become). This setting can also be thought of in terms of how much contrast is added at the edges. It does not affect the width of the edge rims.
- `Radius (-r)` affects the size of the edges to be enhanced or how wide the edge rims become, so a smaller radius enhances smaller-scale detail. Higher values for Radius can cause halos at the edges and a detectable faint light rim around objects. Fine detail needs a smaller Radius. Radius and Amount interact; reducing one allows more of the other.
- `Threshold (-t)` controls the minimum brightness change that will be sharpened or how far apart adjacent tonal values have to be before the filter does anything. This lack of action is important to prevent smooth areas from becoming speckled. The threshold setting can be used to sharpen more-pronounced edges, while leaving subtler edges untouched. Low values should sharpen more because fewer areas are excluded. Higher threshold values exclude areas of lower contrast.

  The following figure shows the unsharp filtering operator.



$$g(x,y) = f(x,y) - f_{smooth}(x,y)$$

$$f_{sharp}(x,y) = f(x,y) + k * g(x,y)$$

In the above figure and equations,

`f(x,y)` is the input,

`k` is a scaling constant, reasonable values for which range from 0.2 to 0.7, and

`fsharp(x,y)` is the final output.

3 Pseudocode

```
color[][] usm(color[][] original, int radius, int amountPercent, int
threshold) {

  // copy original for our return value
  color[][] retval = copy(original);

  // create the blurred copy
  color[][] blurred = gaussianBlur(original, radius);

  // subtract blurred from original, pixel-by-pixel to make unsharp
mask
  color[][] unsharpMask = difference(original, blurred);

  color[][] highContrast = increaseContrast(original, amountPercent);

  // assuming row-major ordering
  for(int row = 0; row < original.length; row++) {
    for(int col = 0; col < original[row].length; col++) {
        color origColor = original[row][col];
        color contrastColor = highContrast[row][col];

        color difference = contrastColor - origColor;
        float percent = luminanceAsPercent(unsharpMask[row][col]);

        color delta = difference * percent;

        if(abs(delta) > threshold)
          retval[row][col] += delta;
    }
  }

  return retval;
}
```

4 Implementation details

The Unsharp Mask filter is implemented using two different approaches.

The first approach is by using the Image buffers which gives better results when compared to the usage of normal OpenCL buffers (the second approach). Two-pass kernels using the image buffers are implemented. The kernels process the input image taking 1D inputs at a time. All the kernels have the same algorithm as explained in the previous section except the change in the data structures (normal buffers versus image buffers) and number of passes. The results from the kernels are verified against the CPU result, if the `'-e'` option is provided. Gaussian kernel is calculated in the CPU itself and the value is passed to the OpenCL device through kernel arguments. The image output is displayed in a window which is implemented using the OpenGL API calls.

The second approach is very similar to the first approach. The only difference is that the second approach uses normal OpenCL buffers instead of Image buffers.

5 References

1. http://en.wikipedia.org/wiki/Unsharp_masking
2. http://gimp.open-source-solution.org/manual/plug-in-unsharp-mask.html

3. http://www.luminous-landscape.com/tutorials/understanding-series/understanding-usm.shtml
4. http://homepages.inf.ed.ac.uk/rbf/HIPR2/unsharp.htm

Contact

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

For AMD Accelerated Parallel Processing:
URL: developer.amd.com/appsdk
Developing: developer.amd.com/
Support: developer.amd.com/appsdksupport