An Industry Standard Benchmark Consortium

# DENBench™ Version 1.0

# Benchmark Name: High-Pass Gray-Scale Filter

## Highlights

- **Benchmarks performance for image processing used in digital still camera and other digital imaging products**
- **Explores 2-D data array access and multiply/accumulate capability**
- **Integer implementation**
- **Seven datasets provide a larger workload compared to the single dataset of ConsumerBench Version 1.1**
- **Inputs are black and white Portable Graymap (.pgm) files**
- **Implements Non-Intrusive Cyclical Redundancy Checksum (CRC) to Check Output Quality**

## Application

A high pass gray-scale filter is used in the front end processing of digital still cameras (DSCs). RGB data from either CCD or CMOS sensors is pre-processed by this filter to deliver image enhancement, and then passed to the JPEG image compression processing. This filter takes a blurry image and sharpens it with a 2-dimensional spatial filter. DSCs implement this filter either in software or hardware, with software giving the flexibility to add customization for picture quality. The number of filter taps can vary from 3(H) x 3(V) to more than 5(H) x 5(V). This benchmark is one of the most frequently used algorithms in image processing and represents a good measure of the CPU performance in digital imaging products.

## Benchmark Description

This benchmark explores the target CPU's capability to perform two-dimensional data array access and multiply/accumulate calculations. For each pixel in the image, the filter calculates the output result from the 9 pixels (including the center pixel) multiplied by filter coefficients, accumulated and then shifted left by 8 bits.

The two-dimensional coefficients used here are:

$$\begin{bmatrix} F11 & F21 & F31 \\ F12 & F22 & F32 \\ F13 & F23 & F33 \end{bmatrix} = \begin{bmatrix} -28 & -28 & -28 \\ -28 & 255 & -28 \\ -28 & -28 & -28 \end{bmatrix}$$

Each pixel is computed according to the following equation:

```
PelValue = (Short)(  F11*P(c-w-1) +F21*P(c-w)  +F31*P(c-w+1)
                   +F12*P(c-1)    +F22*P(c)     +F32*P(c+1)
                   +F13*P(c+w-1)+F23*P(c+w) +F33*P(c+w+1) )

Out = (Byte)(PelValue >>8);
```

Here, P(i) is the pixel intensity, c is the center location of the filter window, w is the width of the input image. The data type of P(i) is byte, and the two-dimensional data is arranged in a linear way. Therefore addition or subtraction of the horizontal image width w and offset of -1 or +1 are required to retrieve the two-dimensional window data. The accumulation is performed as 16-bit data and the final output data is converted to byte data after a shift right by 8 bits. The top/left and right/left borders are blacked out by assigning BLACK a value of 0.

The input data sizes vary and use monochrome .pgm files, performing monochrome or gray-scale calculation. It is not an RGB calculation where the same process is performed three times. Usually the enhancement is performed just in the luminance signal Y, which is the gray-scale signal. If the benchmark score is extrapolated for a larger image, the processing time will be almost linearly proportional to the pixel count (e.g. for a 640 x 480 image, it will be multiplied by 4). The iteration/sec score will be the inverse (e.g. for a 640 x 480 image, iterations/seconds will be multiplied by .25).

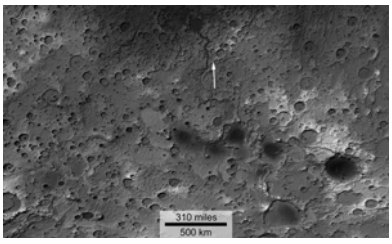
**Description of Datasets**

**Rose Small**

The dimensions are 227x149, 256 colors, 146 colors used in pgm format.

**Goose**

The dimensions are 320x240, 256 colors. The image has 254 unique colors in .pgm format.




**Mars Former Lakes**

Mars Former Lakes is a NASA graphics picture. The dimensions are 800x482, 16 million colors. The image has 255 unique colors in .pgm mode.

**Dragon Fly**

Dragon Fly is an image containing highlights and a wide range of contrast. The dimensions are 606x896, 16 million colors. The image has 162,331 unique colors in .pgm format.

**EEMBC Group Shot**

EEMBC Group Shot is a snapshot of attendees at an EEMBC Board of Directors meeting in 2003. The dimensions are 640x480, 16 million colors. The image has 253 colors in this .pgm file.

**David and Dogs**

David and Dogs is a snapshot of David Weiss and his dogs Sandy, Toga, and Trudy during a rare snowstorm in Austin. It is used as a grayscale image, with good contrast details in the melting snow. The dimensions are 564x230, 256 shades of gray. The image has 215 unique colors.

**Mandrake**

Mandrake is a close up picture of a Mandrill Baboon (sometimes misnamed as "Mandrake"). It has a lot of detail and colors. It has been the default image for the filter benchmarks in both color and gray scale. The dimensions are 320x240, 16 million colors. The image has 213 unique colors in this .pgm version.

**Galileo**

Galileo is a NASA composite image based on actual images of the Jupiter and several of its moons. The dimensions are 290x415, 16 million colors. The image has 36,557 unique colors, and also contains “real black” for over 30% of the picture, which is interesting from an optimization perspective. This .pgm file has 256 colors.

Output quality is measured using Non Intrusive CRC code developed by the EEMBC Certification Laboratory (ECL, LLC). It does not affect the benchmark score.

## Analysis of Computing Resources

**Out of the Box Benchmark:** A “for loop” calculates the filter output one pixel at a time. For one pixel calculation, the center pixel itself and the eight neighbor pixel data should be loaded. This is a time consuming process, considering the offset/width index calculation, and the time spent for the memory or cache access. Higher performance would be expected from a microprocessor with a single-cycle MAC unit.

**Full-Fury Benchmark:** Because of the simple structure of the multiplication and accumulation, a VLIW or SIMD architecture with multiple MAC units is able to offer a simple acceleration. Another possible optimization is loading multiple bytes at a time, although a SIMD architecture may show some overhead for the rearranging the data to feed the SIMD engine. Regarding the memory architecture, the image data is repeatedly used for the consecutive window and can benefit from a data cache. The code size is trivial and will easily fit in to a small L1 instruction cache.