New Mexico Tech

Real-time noise filtering of webcam video

By David Park – EE552: Digital Image Processing

11

Purpose

The goals of this project are as follows: acquire live video from some type of webcam (for extra credit), and filter it in real time using a massively multithreaded, GPU accelerated filter. Another goal is to experiment with several types of filters to achieve the best performance.

Procedure

All software was written in Microsoft visual studio, and utilized libraries from the CUDA and openCV projects. This project was implemented in several steps; the first being to develop single threaded version of the filtering algorithm to verify the procedure. This single threaded geometric mean filtering algorithm was tested with a sample image that was corrupted with Gaussian noise. The code for the geometric mean filter was previously written in MATLAB for a previous assignment; this allowed the code to be translated by hand into C. The GPU version was then implemented in a similar fashion to the CPU version; it also used a single image for testing purposes. After verification of the GPU algorithm was complete, it was merged with the webcam code, which is a modified from a Code Laboratories eye SDK example project. The harmonic mean code was then implemented to provide additional filtering functionality, as well as the ability to switch between filters while the video stream is running.

Results

Near real time filtering was obtained, with a frame rate lower than the stated goal of 60 frames per second. This may have several factors, including:

- The requirement of a single threaded host program
- Processor-block interaction on the GPU
- Memory bottlenecks

Another problem encountered was the overflow of the individual pixel variables in the GPU implementation of the Geometric mean filter. This will be further illustrated with figures below. To further clarify the results of the project, filter output using the single image during testing is also discussed. The single input image used in the testing process is shown below in figure 1.

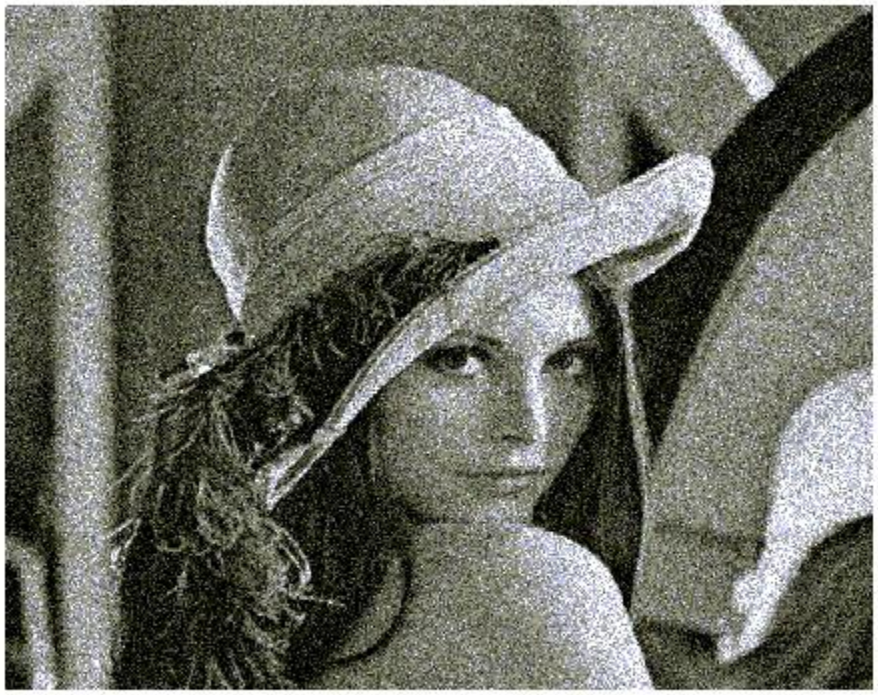

Figure 1 Single input image for algorithm testing

This image was then filtered on the CPU and GPU using both the Geometric mean and harmonic mean filters. Results from each are shown in figures 2 and 3.

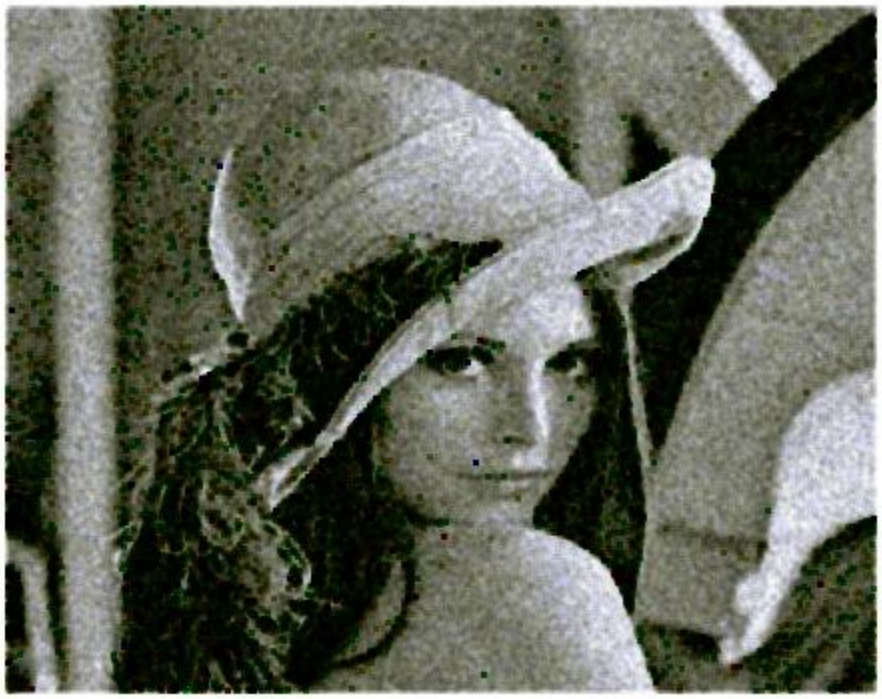

Figure 2 Geometric mean result with GPU implementation using 3 x 3 neighborhood

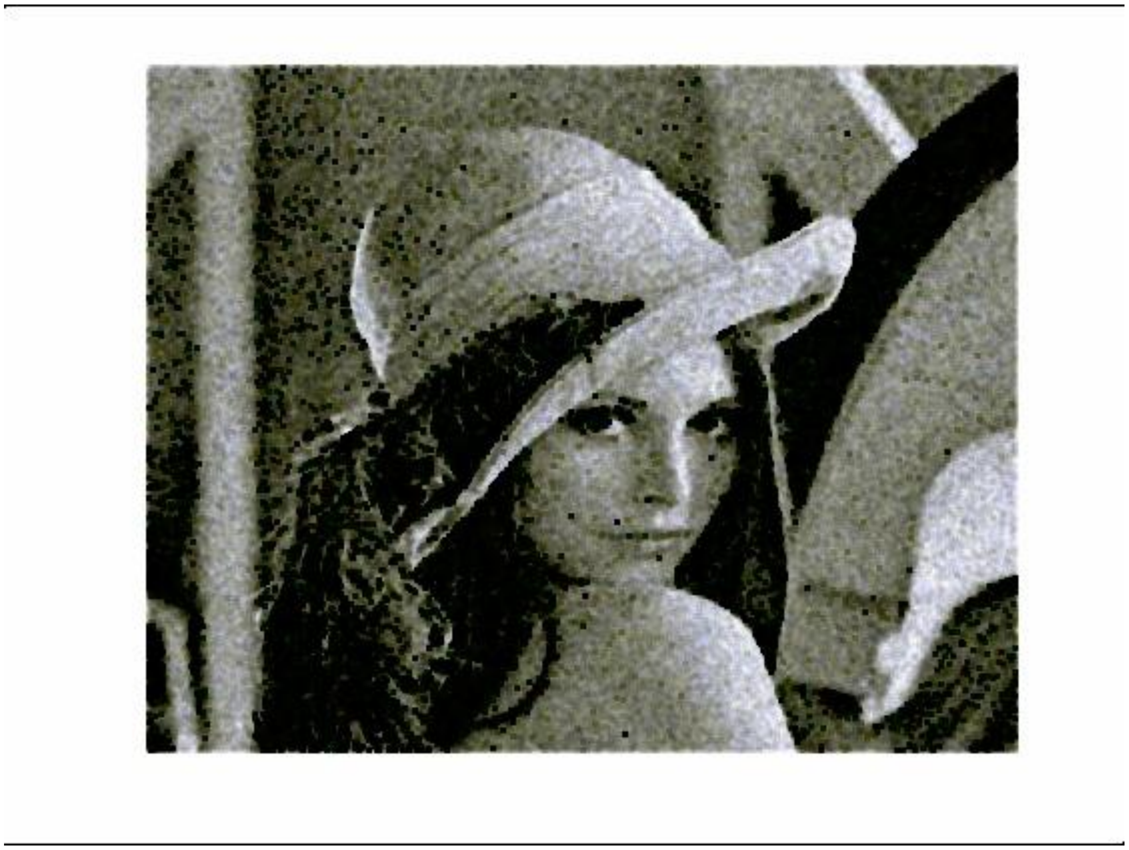

Figure 3 Harmonic mean result using GPU implementation using 3 x 3 neighborhood

The possibility of overflow in the GPU implementation of the geometric filter was discussed earlier. This overflow manifests itself when using a 5 x 5 neighborhood, and is illustrated in figure 4.

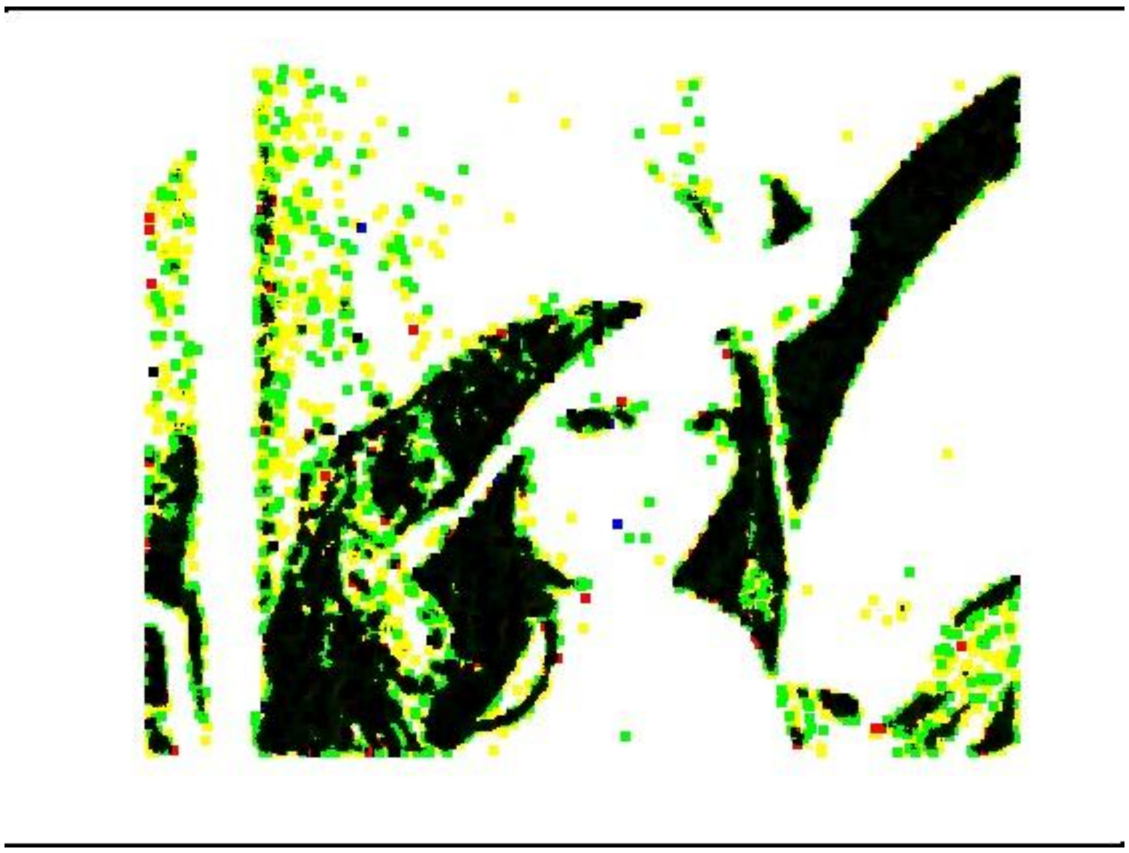

Figure 4 Overflow in GPU Geometric filter, possibly caused by numerical error, with a 5 x 5 neighborhood

The overflow is prevented by changing the neighborhood from a 5 x 5 to a 3 x 3, although some error is possibly still present as illustrated below. The following figure shows the results from subtracting the results from the CPU and GPU implementations of the geometric mean filter with a 3 x 3 neighborhood.

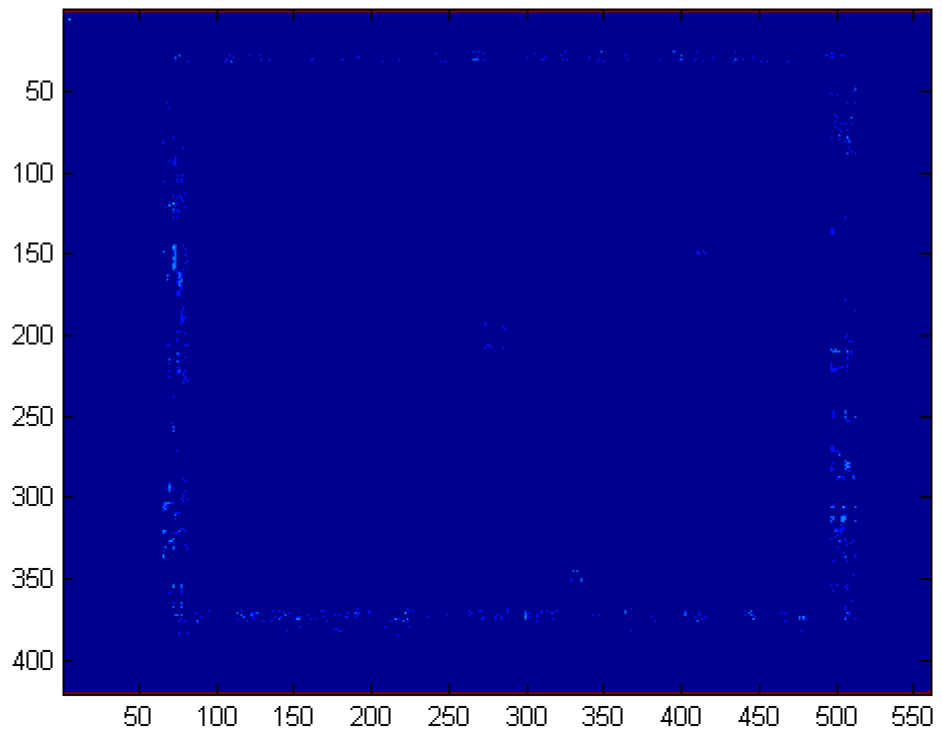

Figure 5 differences between the GPU and CPU implementations of the geometric mean filter

Although there is also differences in the harmonic mean results, this error could indicate the source of the overflow in the GPU implementation, since pow is used, and it's error is possibly proportional to the magnitude of numbers operated on.

Conclusion

This project was successful, given the time allotted, and has potential to be improved. There are still some factors limiting the advance of GPU development, but NVidia is defiantly making it easier as time goes on.

Sources

Digital image processing, third edition by Gonzalez and Woods

Learning OpenCV by Bradski and Kaebler

Cuda By Example by Sanders and Kandrot

Acknowledgments

Dr Hector Erives

Dr. Scott Teare

David Breen

Various Fourm Posts