
# AutoDEM

## Process Overview and Performance

Authors: Wilkin Chau
Michael Goldberg

Date: August 14, 2010

# Executive Summary

Due to the large amount of computationally intensive operations, the process of digital elevation extraction from a pair of epipolar images is one of the most time consuming processes. Even with some clever optimization coding strategies, the AutoDEM PPF process requires hours, sometimes more than a day, to generate a single digital elevation model (DEM) image. To improve the performance, the DEM extraction process has been redesigned to take advantage of the multi-core processor and graphics processing units (GPUs) for the GXL system.

This report details the performance of the new DEM extraction application for the GXL system. The main logic of the application is based on the code in “genautodem.cpp”, which contains the core functionality of the DEM extraction for the PPF AutoDEM. Besides improving the speed performance, the program logic of the new DEM extraction has been redesigned for ease of maintenance.

Similar to other GXL applications, such as pansharpening and orthorectification, the new DEM extraction obtains substantial speed improvement over the PPF AutoDEM. Five datasets, which were acquired from the sensors of WorldView, Ikonos and GeoEye-1, have been tested. Using the PPF AutoDEM on a Linux machine, the processing times are in the range of 3 – 9 hours, while the GXL AutoDEM completes each individual process between 14 and 76 minutes. The speeds up factors are in the range of 7 and 27 times for a Linux desktop machine with a single GPU card and 4-core processor. It took 9 to 53 minutes to process each dataset, and the speed-up factors are in the range of 9 and 20 times on a Windows machine with one GPU and 8-core processors. As the GXL AutoDEM application is designed to scale with better hardware, higher performance is expected for the systems that have more cores, more GPUs and more processing power.

# Table of Contents

# Tables

# 1 Introduction

Due to the intensive computational requirement to estimate the digital elevations from a pair of epipolar images, the PPF AutoDEM normally requires many hours (or a day) to generate a single DEM image. This is one of the most time consuming processes developed by PCI. To reduce the processing time, we undertook a project of redeveloping automatic DEM extraction. This project is a part of GXL system development using multi-core processor and GPU technology to obtain a substantial performance gain.

There is no major change of logic for the DEM extraction. But some modifications have been made to improve the performance and code quality and robustness. For performance, the changes are mainly for utilizing multi-threads and avoiding branching, which can drastically reduce performance. These changes may introduce some overheads, which can be minimized with carefully designed code. In addition to the slow performance, there are issues with the code quality in the AutoDEM PPF. Since this PPF has been developed and modified over a period of 10 years, the code has become segmented and very hard to maintain. The program logic has been modified to avoid code fragmentation and to produce easily maintainable code.

This report provides an overview of the DEM extraction process, followed by a brief description of enhancements used for speeding up the process. Finally, the report presents the speed performance results obtained based on tests run on Linux and Windows desktop machines.

# 2 Process Overview

Elevation is estimated based on a pair of epipolar images, which are derived from the left and right raw images using their RPC math models. One can deduce the elevation of a point on the left epipolar image by finding the matching point in the right epipolar. The DEM estimation procedure can be summarized as following:

1. Estimate the x- and y-direction parallax ranges, which describe the pixel displacement of the points in the left and right image, to determine the search range of the right image.
2. Calculate the elevations in coarse resolution based on the matching results in the parallax ranges.
3. Use the coarse level elevations from Step 2 to compute the projection points in the right epipolar image for the first level elevation estimation.
4. Perform matching for the low-detail DEM using the coarse level elevations to define the

searching region. Matching is performed with full resolution data using 17x17 matching window.

5. If medium- or high-detail DEM is requested, matching results are refined around the previously matched points using window size of 11x11 for the medium-detail and 5x5 for high-detail DEM.
6. Determine the full resolution DEM based on the matching results in Step 4 or 5.

## *2.1 Determine the Parallax Range*

Finding a matched point is an expensive operation since hundreds of comparisons are required to determine the best matching point. To reduce the number of comparisons, search is conducted only over the possible region for the given elevation range. To determine the search region, the DEM extraction process estimates the maximum x- and y-direction parallax ranges. The parallax range is calculated by using the four corners and centre point of the left epipolar image project to the right epipolar image for the specified minimum and maximum elevations. The minimum and maximum displacement between the left and right image is used as the minimum and maximum parallax amount, respectively. Because the mathematical model that describes the mapping between the image coordinates and the ground position is only approximate, the projection transformation is not precise. To make sure all matched points will be covered, the parallax range is enlarged to a preset amount to accommodate the projection error.

## *2.2 Estimate Coarse Level Elevation*

The coarse level elevation is determined from the down-sampled epipolar images. If the image is more than 3000 pixels high or wide, overview epipolar images are first used to estimate the rough elevations, which are later used for coarse level elevations. For small epipolar images the coarse level elevations are directly computed using the parallax range.

The estimated parallax range defines the search range of the right epipolar image for the very first DEM estimation, which can be obtained from the overview or coarse down-sampled epipolar images. For each individual point on the left epipolar image, the local region is compared against the local regions of the search region in the right image. To determine the best matched point, the normalized cross-correlation coefficients (NCC) of the local regions from the left and right epipolar images are computed. The point in the right image with the highest NCC coefficient is classified as the matched point, provided that the coefficient is higher than a predefined threshold. Once the matched points are found, the elevations are computed. Depending on the resolution, different window sizes are used to compute the NCC coefficients. For overviews, the window size is 7 by 7 pixels, and for the coarse level 25 by 25 pixels window is used.

If overviews are used, the coarse level DEM are derived from the rough elevations. Given a coordinate in the left image and its elevation, one can determine the corresponding projection points of right epipolar image using the math model. The projected points of the right image are then computed using the rough elevations. These points are then used as the mid-point of the search region, which is 33 pixels wide and 3 pixels high, for matching (with window size of 25x25); and subsequently estimate the coarse level DEM.

Several post processing steps have been used to refine the estimated DEM. Sometimes matched points cannot be identified, while other points may not be matched correctly due to poor image acquisition conditions or obscured view. To minimize the adverse effects of incomplete results, image filtering and hole-filling are applied to the estimated DEM. Noise filters are applied twice to remove the outliers, which may cause bright and dark spots for hole-filling. Hole-filling is then applied to fill in the missing elevation values. Median and average filters are applied to create a smooth DEM result.

## *2.3 Low-Detail DEM Matching*

The coarse level elevations help to reduce the search range for point matching between the left and right epipolar image. Using the math model and the coarse elevations, the points in the left image are projected to the right image. These projected points are then used as the mid-point of the search range to estimate the low-detail DEM. The size of search region is 21 pixels in x-direction and 3 pixels in y-direction. The NCC coefficients are computed using the window size of 17x17 at this level.

## *2.4 Medium / High-Detail DEM Matching*

If desired, further matching refinement is made by using smaller window size to compute the NCC coefficients. The idea is that large matching window size may not be matched correctly since the matching of pixel with extreme values in the window contributes more for the coefficients. With a smaller window size, the matching result is less likely to be negatively affected by the extreme values. The refinement procedure is based on the previously matched location. The DEM isn't estimated until all refinements are completed. For the refinement, the search is done within the same line as the previously matched location and the matched location in x-direction becomes the centre of the search region of for the next matching. For medium level DEM, the search range is +/- 5 pixels of the centre and the high level is +/- 2 pixels. The matching window size is 11x11 and 5x5 for the medium- and high-detail DEM, respectively.

## *2.5 Determine Full Resolution DEM*

After performing matching up to the target detail, elevations are estimated based on the matching

results. The matching results provide information of the corresponding location in the right epipolar image of a given point in the left image. Along with the math model, these image positions are used to determine the elevation of the point. An image of elevations is constructed in a pixel-by-pixel fashion. The final DEM from the full resolution data is produced by applying the same filtering and hole-filling procedure as for the coarse level DEM to the elevation image.

The above description provides an overview of the AutoDEM extraction process. This overview outlines both the PPF and GXL versions. The PPF version has been developed and enhanced over a period of 10 years. There are many fine tuning steps in the process. Efforts have been made to try to minimize the difference between the PPF and GXL versions. However, changes are still being made to improve the efficiency and code quality. One of the major differences is the definition of matching score. For PPF the matching score is defined as the total difference of the mean deviant of matching window in the left and right images, while GXL AutoDEM uses the NCC coefficient as score.

# 3 Performance Enhancements

The main objective of this project is to reduce the processing time for the DEM extraction procedure by taking advantage of multi-threaded and GPU technologies. To determine an elevation, a matching procedure is invoked to determine the corresponding point in the right epipolar image for each individual point in the left epipolar image. The majority of the computation burden comes from this matching operation, which requires comparing hundreds of local areas for every location in the left image. Therefore, the matching operation is the main focus for speed improvement.

## *3.1 Matching Operation*

Normalized cross-correlation (NCC) is used to determine the corresponding point in the right epipolar image for each location in the left image. The computation of NCC coefficient requires the window sum of pixel values and window sum of the squares in the template and search areas as well as the window sum of products between the template and search areas. To obtain a single NCC coefficient for a matching window size of K x K, it requires 2 x $K^2$ additions for a window sum, 2 x $K^2$ additions and 2 x $K^2$ multiplications for window sum of squares (the factor of 2 is due to the computation requirement for both the template and search windows), and $K^2$ additions and $K^2$ multiplications for window sum of products. The required number of operations is 5 x $K^2$ additions and 3 x $K^2$ multiplications. For search region with M pixels, the total number of

operations is M times of the above amounts. This brute force approach only work for small windows and search area. For general DEM estimation, this becomes impractical. For example, with 10x10 matching window and search area of 100 pixels, it requires 50,000 additions and 30,000 multiplications to identify the match point of a single location in the left epipolar images.

### 3.1.1 Overlap Speedup Strategy

When perform matching for an area, there is a way to reduce the number of required computation by taking advantage of overlap matching window of the neighbouring pixels. To compute the window sums of an area, one can break down the computations into two groups: one for the pixels in the overlap area, and one for the non-overlap area. The computational operations can be reduced by computing the window sums in overlap area only once for both pixels. To compute all of the window sums for two adjacent pixels, for example, the above strategy requires only total of 5 x ($K^2$ + K) additions and 3 x ($K^2$ + K) multiplication, not 10 x $K^2$ additions and 6 x $K^2$ multiplications as before. Effectively, it only needs 5 x K additions and 3 x K multiplications to get the window sums for the second pixel. As for the previous example of 10x10 matching window and 100-pixel search area, instead of 100,000 additions and 60,000 multiplications, it only requires 55,000 and 33,000 additions and multiplications, respectively.

The above overlap strategy works effectively when all the window sums are used by the process. To compute the NCC coefficients, the window sums of pixel values and the sum of squares for the template and search areas are required for every pixel. Using this approach to compute these window sums works extremely well; and therefore, it is being used in both PPF AutoDEM and GXL AutoDEM processes.

### 3.1.2 Computing the Window Sums of Products

The overlap speed up approach is much less effective for computing the window sum of the products between the template and search windows. When the displacements are the same for all pixels in the template area and their corresponding matched pixels in the search area, the speed up approach will work perfectly. In fact, an initial test results show the speed performance gain for computing NCC coefficient is about 200x in such ideal situation. Unfortunately, the constant displacement never occurs in practice. Depending on the range of elevations in an area and image resolution, the displacement difference may be in the ranges of tens or even hundred pixels. To handle the area using the above speed up approach, it requires pre-computing all window sums of products to cover all possible displacements, even though lots of the pre-computed window sums may not be needed. In some situation, such overhead is so big that the approach is slower than straight computation of the window sums. This may be the reason that

the PPF AutoDEM process always uses brute force approach to compute the window sums of products.

For the GXL AutoDEM process, a hybrid approach is used to computing the window sums of products. To reduce the overhead for the speed up strategy, the process uses small template block to avoid large displacement difference among the pixels within the block. Inside the block, pixels that have the same displacement values are group together. Each group is handled separately. If there are less than 100 pixels in a group (the value of 100 is chosen based on some empirical tests), brute force approach is used; otherwise, the speedup approach is used. To further reduce the overhead, the process computes the window sums of products for the areas that cover only the interest pixels. Due to the small search area and matching window size in the refinement step, which is for the medium- or high-detail DEM output, the window sums of products are always computed using the brute force approach.

## *3.2 DEM Filtering and Hole-Filling*

After getting the estimated DEM, there are some void areas related to the locations that have no matched points found in the right epipolar image. Noise filtering that removes the extreme DEM value is first applied, followed by a hole-filling procedure to fill in the void areas to create a complete DEM raster image. Median and average filters are then applied to smooth the final DEM image. These steps apply the same operation for each individual pixel in the image and can be processed efficiently by the GPU. Therefore, the filtering and hole-filling procedures are assigned to the GPU if one available.

## *3.3 DEM Matcher Threads*

To take advantage of the multi-core processor, the GXL AutoDEM process breaks down the epipolar images into small blocks. Each block is handled independently by a main thread, which has its own memory pool. The main threads are responsible for estimating the DEM for the blocks. If GPUs are available, each GPU will be assigned to a different main thread to perform the matching operation. The maximum number of main threads is restricted by the number of cores and the amount of host memory available. Given enough of memory, all cores will busy on processing the block data simultaneously. This approach maximizes the usage of processing power. For example, with one gigabyte of memory available for a four-core processor, the application constantly uses 98% of CPU processing resource to estimate the DEM.

# 4 Performance Results

## 4.1 System Hardware

A Linux desktop machine with Intel® Core2™ Extreme Quad-Core 2.66GHz Q6700 CPU, 8 GB RAM and a Windows desktop machine with Intel® Core i7 920 8 cores @ 2.67Hz CPU, 2 GB RAM were used to evaluate the performance. Both machines were equipped with NVIDIA GeForce GTX 280 GPU with 1GB of GDDR3 SDRAM.

## 4.2 Development Environment and Tools

The development environment is openSuSE Linux 11.1 (64-bit). Major development was done using the GCC (version 4.2.1), using the OpenMP 2.5 libraries for multithreading and NVIDIA's CUDA SDK release 2.3 for programming the GPUs.

## 4.3 Test Datasets

Five datasets were used to test the speed performance of the GXL AutoDEM application. They are all panchromatic images and are the products of WorldView, WorldView2, Ikonos and GeoEye-1 (see Table 1). Epipolar images, generated by the PCI Orthoengine application, were in PCI PIX format. The images from the WorldView products have data type of 16U, all others are 8U.

**Table 1. Test Datasets**

| Scene | Morrison | India | Australia | Rwanda | Yemen | Hobart |
|---|---|---|---|---|---|---|
| **Product** | WorldView | WorldView 2 | Ikonos | GeoEye-1 | GeoEye-1 | GeoEye-1 |
| **File Size (MB)** | 2,440 | 1,740 | 290 | 202 | 310 | 445 |
| **Data Type** | 16U | 16U | 8U | 8U | 8U | 8U |
| **Epipolar Image Dimension** | 16401P x 21267L | 29269P x 46221L | 13690P x 16558L | 11357P x 13952L | 14776P x 16285L | 16400P x 20244L |

## *4.4 Testing Parameters*

All datasets were tested with parameter of “extinter” (Extract Interval) set to 2, “demdet” (DEM Details) set to “medium”, and “datatye” set to 32R.

## *4.5 Test Performance*

The processing times of AutoDEM PPF running on the Linux desktop are used as reference. Table 2 shows the processing times and throughputs of the test datasets. The AutoDEM PPF took nearly four hours to complete the smallest dataset and over nine hours to complete the largest dataset. The throughput range is between 2,034 and 10,174 pixels per second for the Linux workstation and 1,517 and 12,253 for the Windows workstation. With a more powerful processor, the Windows workstation had higher throughput. The increase factors are in the range of 1.75x and 4.33x. It is not known what causes the difference in increase factors among the datasets. The high throughout of the India dataset is partly due to the larger amount of background pixels in the raster.

**Table 2. PPF AutoDEM Performance**

| Scene | Morrison | India | Australia | Rwanda | Yemen | Hobart |
|---|---|---|---|---|---|---|
| **Output DEM Dimension** | 8200P x 10633L | 14634P x 23110L | 6845P x 8279L | 5678P x 6976L | 7388P x 8142L | 8200P x 10122L |
| **Output Raster Pixels (x1000)** | 87,190 | 338,192 | 56,670 | 39,610 | 60,153 | 83,000 |
| **Linux Workstation** | | | | | | |
| **Processing Time** | 5 hrs 09 mins | 9 hrs 14 mins | 7 hrs 27 mins | 3 hrs 53 mins | 8 hrs 13 mins | 8 hrs 35 mins |
| **Throughput (pixels / sec.)** | 4,703 | 10,174 | 2,113 | 2,833 | 2,034 | 2,989 |
| **Windows Workstation** | | | | | | |
| **Processing Time** | 2 hrs 41 mins | 7 hrs 40 mins | 2 hrs 46 mins | 2 hrs 52 mins | 4 hrs 29 mins | 15 hrs 12 mins |
| **Throughput (pixels / sec.)** | 9,026 | 12,253 | 5,690 | 3,838 | 3,727 | 1,517 |

By using the multi-thread, GPU technology and software enhancement, the GXL AutoDEM consistently provides much better performance compared to the PPF version. Both the Linux and

Windows machines used four main threads during the tests. On the Linux machine, it took about 76 minutes to finish the largest dataset. The throughput range is between 47,722 and 74,035 pixels per second. Compared to the throughput of PPF AutoDEM on the same workstation, the speeds up factors are between 7x and 27x for the Linux machine (Table 3). On the Windows machine, it took about 52 minutes to finish the India dataset. The throughput range is between 71,757 and 106,989. The speeds up factors are in the range between 8x and 20x (Table 5).

**Table 3. GXL AutoDEM Performance with 4-core CPU & 1 GPU ( Linux )**

| Scene | Morrison | India | Australia | Rwanda | Yemen | Hobart |
|---|---|---|---|---|---|---|
| **Processing Time** | 27 mins 45 secs | 76 mins 08 secs | 18 mins 09 secs | 13 mins 50 secs | 18 mins 26 secs | 24 mins 09 secs |
| **Throughput (pixels / sec.)** | 52,367 | 74,035 | 52,038 | 47,722 | 54,387 | 57,281 |
| **Speedup Factor** | 11x | 7x | 25x | 17x | 27x | 21x |

**Table 4. GXL AutoDEM Performance with 4-core CPU & 1 GPU ( Windows )**

| Scene | Morrison | India | Australia | Rwanda | Yemen | Hobart |
|---|---|---|---|---|---|---|
| **Processing Time** | 19 mins 44 secs | 52 mins 44 secs | 11 mins 23 secs | 9 mins 12 secs | 13 mins 15 secs | 15 mins 48 secs |
| **Throughput (pixels / sec.)** | 73,641 | 106,989 | 82,972 | 71,757 | 75,664 | 87,553 |
| **Speedup Factor** | 8x | 9x | 15x | 19x | 20x | 57x |

Performance was also measured without using a GPU. The results show how much performance gain can be obtained by the multi-thread technology alone. On the Linux machine, it took about 261 minutes to finish the largest dataset. The throughput range is between 9,353 and 21,623 pixels per second. Compared to the throughput of PPF AutoDEM, the speeds up factors are between 2x and 8x for the Linux machine (Table 5). On the Windows machine, it took about 260 minutes to finish the India dataset. The throughput range is between 27,001 and 43,206. The speeds up factors are in the range between 3x and 28x (Table 6). With an *N*-core processor, one

might expect that a program can run less than *N* times of the speed of the identical program running without using multi-threading. As shown the Tables below, the GXL AutoDEM with CPU only sometimes has speed up factor over the number of cores in the processor. However, the gain is not consistent across all datasets. The application only has a speed up factor of 2 for the India dataset on the Linux workstation, while the speed factor is 8x for the Rwanda dataset. The poor performance of the India dataset is mainly because only two main threads were being used for most of the time. While the application specifies the number of main threads at most equal to the number of cores, constraint by the availability of the memory, it is possible that some background jobs take up some resources and reducing the processing time available for the main threads. In such case, additional performance penalty exists, which is caused by performing the context switching to prepare for the main threads swap in/out for the kernel to process.

**Table 5. GXL AutoDEM Performance with 4-core CPU Only ( Linux )**

| Scene | Morrison | India | Australia | Rwanda | Yemen | Hobart |
|---|---|---|---|---|---|---|
| **Processing Time** | 97 mins 02 secs | 260 mins 40 secs | 58 mins 37 secs | 70 mins 35 secs | 94 mins 07 secs | 96 mins 38 secs |
| **Throughput (pixels / sec.)** | 14,976 | 21,623 | 16,113 | 9,353 | 10,652 | 14,315 |
| **Speedup Factor** | 3x | 2x | 5x | 8x | 3x | 5x |

**Table 6. GXL AutoDEM Performance with 8-core CPU Only ( Windows )**

| Scene | Morrison | India | Australia | Rwanda | Yemen | Hobart |
|---|---|---|---|---|---|---|
| **Processing Time** | 38 min 48 secs | 155 min 27 secs | 24 min 4 secs | 24 min 31 secs | 37 min 4 secs | 32 min 1 sec |
| **Throughput (pixels / sec.)** | 37,452 | 36,259 | 39,245 | 27,001 | 27,047 | 43,206 |
| **Speedup Factor** | 4x | 3x | 7x | 7x | 7x | 28x |

# 5 Conclusion

The GXL AutoDEM project demonstrates again the benefit of using multi-thread and GPU technologies. While software enhancement provides some performance gain, the majority of performance gain is obtained by taking advantage of the hardware technologies. Unlike most of the GXL applications, which are I/O bound, the DEM extraction is processing bound and leads to the impressive speed performance.

The performance on the Windows machine is better than the performance on the Linux machine in the tests. It is partly due to the speed improvement of the i7 920 CPU over the Q6700. More importantly there are 8 cores in the Windows machine while only 4 cores in the Linux. The speed gain over Windows shows the capability of the new GXL AutoDEM application making use of the available hardware effectively. With enough memory to support more main threads, it is expected that performance on CPUs that have higher speed and more cores can show even more improvement.

GeoImaging Accelerator Ortho Performance Test Results
A PCI Geomatics Whitepaper.

Authors: Wilkin Chau, Michael Goldberg

**PCI Geomatics Enterprises Inc.**
**50 West Wilmot Street**
**Richmond Hill, Ontario**
**Canada, L4B 1M5**
**Phone: (905) 764-0614**
**Fax: (905) 764-9604**
**Email: info@pcigeomatics.com**
**Web: www.pcigeomatics.com**