GeoImaging Accelerator
Ortho Performance Test Results

A PCI Geomatics White Paper

April 2009

Executive Summary

PCI Geomatics has developed new, high-speed, orthorectification functions. The code takes advantage of modern, multi-core processor architecture, as well as NVIDIA's Graphical Processing Units (GPUs), to process standard data products from the WorldView-1, QuickBird and IKONOS satellites.

With two GPUs on board, the code is capable of processing 3.25 TB per day of WorldView-1 Basic products (or 1625 2-Gigabyte scenes), even while using rigorous math model calculations for every pixel in the ortho. This is approximately 65 times faster than the equivalent computation with PCI's ortho PPF, and 9 times faster than using the PPF with a more typical sampling interval of 4 pixels. Even with no GPUs on board, the system can process 1.75 TB per day as a result of multithreaded processing and improved IO.

Using a single GPU, the improvement was still 6 times faster than the PCI ortho PPF, and 1.5 times faster than a non-GPU, multi-threaded implementation.

This demo shows the potential of the GPU in speeding up problems that are computationally intensive in PCI's field of business (DEM creation, image matching, pan-sharpening).

Table of Contents

Figures

Tables

1 Introduction

The Ortho Demonstrator is a high-performance, orthorectification system developed as a demonstration package for use by PCI Geomatics. It takes advantage of modern computer hardware (multi-core processors and Graphical Processing Units) and multithreaded programming techniques to achieve much higher orthorectification throughput than is possible with PCI's existing commercial offering of Pluggable Functions (PPFs) and desktop software (Geomatica).

In order to gain information useful to PCI's future work in multithreaded and hardware-accelerated processing, a multitude of different tests have been carried out. These tests have helped to quantify both the performance and the accuracy of the orthorectification code.

This report describes the background and overall goals of the Ortho Demonstrator project, and provides an extensive description of the tests that have been carried out to characterize the newly developed orthorectification system.

2 Project Description

2.1 Background

The Ortho Demonstrator project was performed in cooperation a PCI reseller who is a geospatial data and software provider. They offer value-added production services and are pursuing large opportunities with agencies that wish to process several million square kilometres' worth of satellite imagery per year; they wish to use PCI technology as part of their solution.

In order to convince the agency contacts that PCI technology could achieve the required throughput, a collaborative approach included the purchase of a modern workstation equipped with a multi-core processor and Graphical Processing Units and development of orthorectification code that leverages this hardware. Required performance figures were not included in the agreement; PCI was expected merely to put forth a best effort to see what is achievable.

2.2 System Hardware

The system purchased was an HP Blackbird 002 LCi machine, pictured in Figure 1. It features an Intel® Core2™ Extreme Quad-Core 3.0GHz QX9650 CPU, 8 GB RAM, one 7200 RPM and two 10,000 RPM SATA hard drives, and two NVIDIA GeForce GTX-280 GPUs with 1GB of GDDR3 SDRAM. The total cost of the system was $7600 USD.

Figure 1: Orthorectification workstation

2.3 Development Environment and Tools

For development of the orthorectification software, openSuSE Linux 10.3 (64-bit) was installed on the workstation. All development was done using the Intel C++ compiler (version 10.1), using the OpenMP 2.5 libraries for multithreading and NVIDIA's CUDA SDK release 2.0 for programming the GPUs.

2.4 Test Datasets

Several types of product with different bit depth, number of channels, ground sample distance and source sensor were used for the development. Table 1 lists the major different products that were included in the test procedures. For some tests, they were grouped into datasets with common bit depth, processing level and number of channels.

Table 1: Development Test Datasets

Product, Type, Resolution	Dataset	Raster Type	Channels	Num Scenes	Total GB
WorldView-1 Level 1B Panchromatic *0.5 metre*	16U_PAN	16U	1	24	37.3
QuickBird Level 1B Panchromatic *0.6 metre*	16U_PAN	16U	1	13	19.3
QuickBird Level 1B Multispectral *2.4 metre*	16U_MS	16U	4	9	3.4
IKONOS Geo Ortho Kit Panchromatic *1.0 metre*	16U_PAN_IK	16U	1	18	17.9
SPOT-5 Level 1A panchromatic *2.5 metre*	8U_PAN	8U	1	143	85.0
QuickBird OrthoReady PanSharpened *0.6 metre*	16U_PS	16U	4	69	105.7

From these large test datasets, a subset of scenes (representing a progression in output filesize) was used for many of the tests. These scenes are listed in Table 2.

Table 2: Primary Test Scenes

Scene	Image Filename	Product	Ortho Size (GB)
Australia04	08APR07004925-P1BS_R2C1-005757713010_01_P001.TIF	WorldView-1 Level 1B	0.51
Vanc03	07NOV21190422-P1BS-005728476030_01_P003.TIF	WorldView-1 Level 1B	0.86
Vanc06	07NOV21190538-P1BS-005728476030_01_P006.TIF	WorldView-1 Level 1B	1.07
MadridPan03	po_2618661_pan_0030000.tif	IKONOS Geo Ortho Kit	1.23
Denver03	08JAN20175253-P1BS-005728476010_01_P003.TIF	WorldView-1 Level 1B	1.97
Denver04	07DEC12174947-P1BS-005728476010_01_P004.TIF	WorldView-1 Level 1B	2.29
SanFran	07NOV26185329-P1BS-005695617010_01_P001.TIF	WorldView-1 Level 1B	2.49
Vanc02	07NOV21190420-P1BS-005728476030_01_P002.TIF	WorldView-1 Level 1B	2.68
Beijing	07NOV27024615-P1BS-005701980010_01_P001.TIF	WorldView-1 Level 1B	3.01
Vanc04	07NOV21190535-P1BS-005728476030_01_P004.TIF	WorldView-1 Level 1B	3.13
Vanc05	07NOV21190537-P1BS-005728476030_01_P005.TIF	WorldView-1 Level 1B	3.34

3 Test Results

In order to fully characterize the performance of the system, five categories of tests have been carried out. Their names and definitions can be found in Table 3.

Table 3: Test Categories

Test Category	Definition
A. System Throughput	Describes the overall speed of the system after optimizations have been carried out
B. Accuracy Tests	Determines the effect of certain optimizations on the geometric accuracy of the generated orthos
C. Performance Tuning	Determines the best combination of tunable parameters to yield the highest processing speed
D. Specific Optimizations	Characterizes the relative effect of specific optimizations on the processing speed
E. Other Tests	Answers other questions related to the developed system

Results from the different tests are described in the following sections.

3.1 System Throughput

The system throughput tests were the last tests to be carried out, but since they describe the item of greatest interest they are presented first.

The throughput tests were carried out to answer two major questions:

1. How much data (of a given product type) can be processed in a given period of time?
2. How does this compare to PCI's existing commercial offering?

With reference to configuration parameters described in following sections, the system configuration used for these tests is summarized in Table 4.

Table 4: System configuration for throughput tests

Configuration parameter	Value
Number of GPUs:	2
Max memory to use on GPUs:	900 MB
Max memory to use on host	4 GB
Processing strategy	TP_ACCEL: All possible processing, including math model calculations, is done on the GPU. A custom thread pool implementation is used to associate processing threads with particular GPUs.
Latitude/longitude grid spacing:	100 pixels
Row/column grid spacing:	1 pixel

3.1.1 Overall Data Volume

In order to test the overall data processing capability, datasets from the different data categories described in Table 1 were each processed for several hours, and the results were used to derive processing rates. The resultant values are shown in Table 5 below.

Table 5: Processing Throughput

Dataset	Product Type	MB/Sec	GB/Min	TB/Day
16U_PAN	WorldView-1 / QuickBird Level 1B	39.59	2.32	3.26
16U_PAN_IK	IKONOS Geo Ortho Kit	35.67	2.09	2.94
16U_PS	QuickBird OrthoReady 4-channel pan sharpened	42.67	2.50	3.52
8U_PAN	SPOT5 Level 1A 2.5 metre	24.23	1.42	2.00
16U_MS	QuickBird Level 1B 4-channel multispectral	55.47	3.25	4.57

Due to the nature of the orthorectification process, the data volume that can be processed is largely defined by the number of bytes per pixel for a given ground location. While the 16U_MS dataset occupies 8 bytes per pixel (4 channels at 2 bytes per pixel per channel), the 8U_PAN dataset occupies only one byte per pixel. Thus, the 8U_PAN dataset represents nearly the worst-case scenario in terms of performance. The only worse case would be 8U data from sensor that collects in a North/South direction rather than in the orbital path direction, because it has less blackfill in the resultant ortho.

Timing results from individual scenes are shown as a function of file size in Figure 2. The processing time is observed to be a roughly linear function of file size.

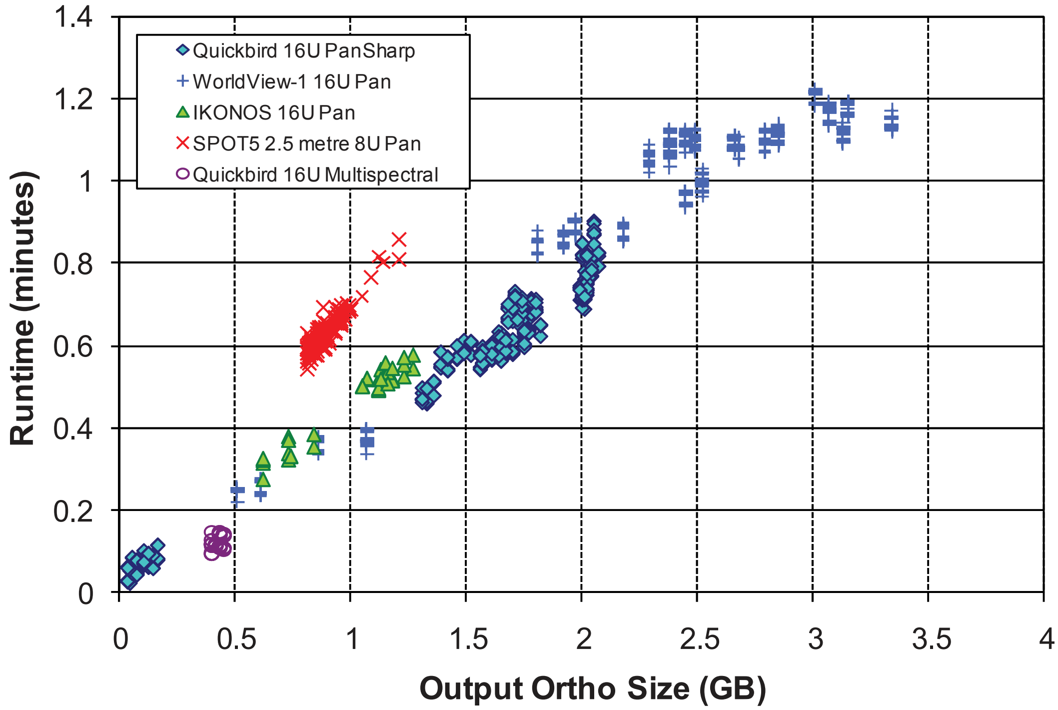

Figure 2: Processing time vs. file size

3.1.2 Comparison with SDK

The PCI SDK was installed on the workstation used for development of the Ortho Module and several scenes were processed using the ortho PPF. The same scenes were processed using the new Ortho Module. The results are shown in Figure 3 and Figure 4.

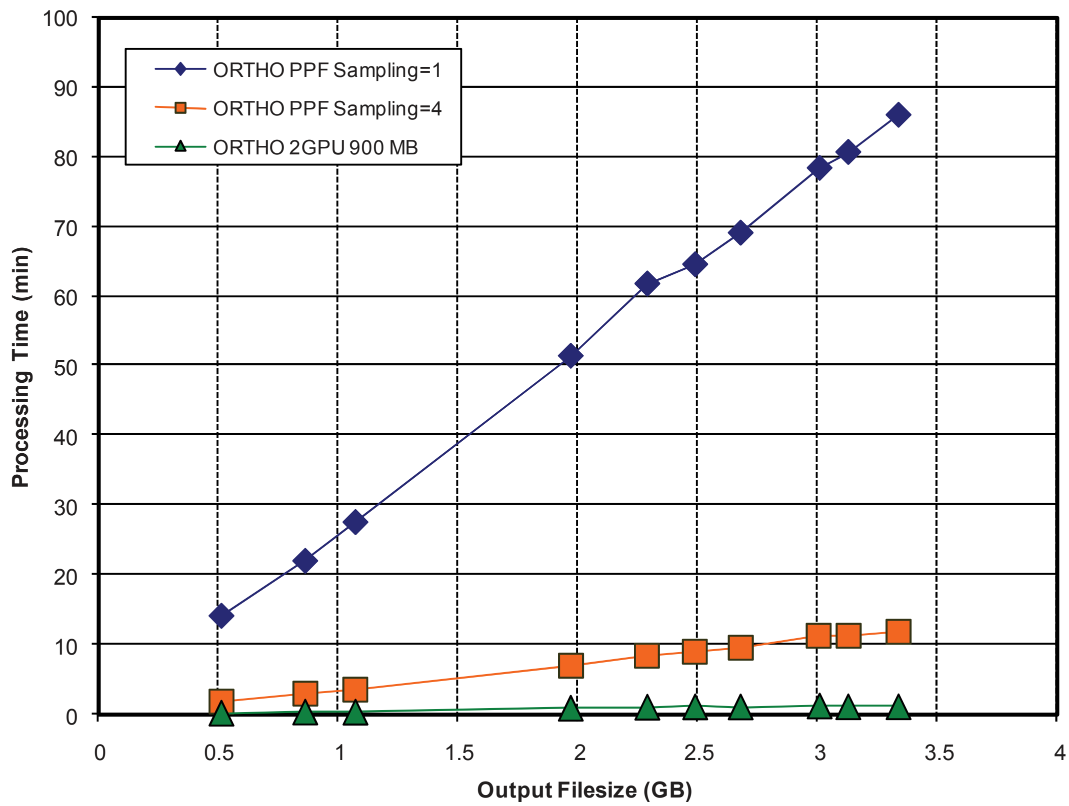

Figure 3: Runtime comparison of ProSDK (PPF) vs. Ortho Module

In terms of the amount of processing being done, the fairest comparison is between the new module and with the ortho PPF at a grid spacing of 1. However, as can be seen from the figure, the processing runtime at a grid spacing of 1 is too long to be practical. Most users will use a grid spacing of 4 or higher. This may result in some localized accuracy issues, as will be discussed in Section 3.2.

When a sampling interval of 1 is used, the average speedup of the new module is 65x with respect to the ortho PPF. When the ortho PPF is used at a sampling interval of 4 and the new module uses a sampling interval of 1, the average speedup is 8.9x.

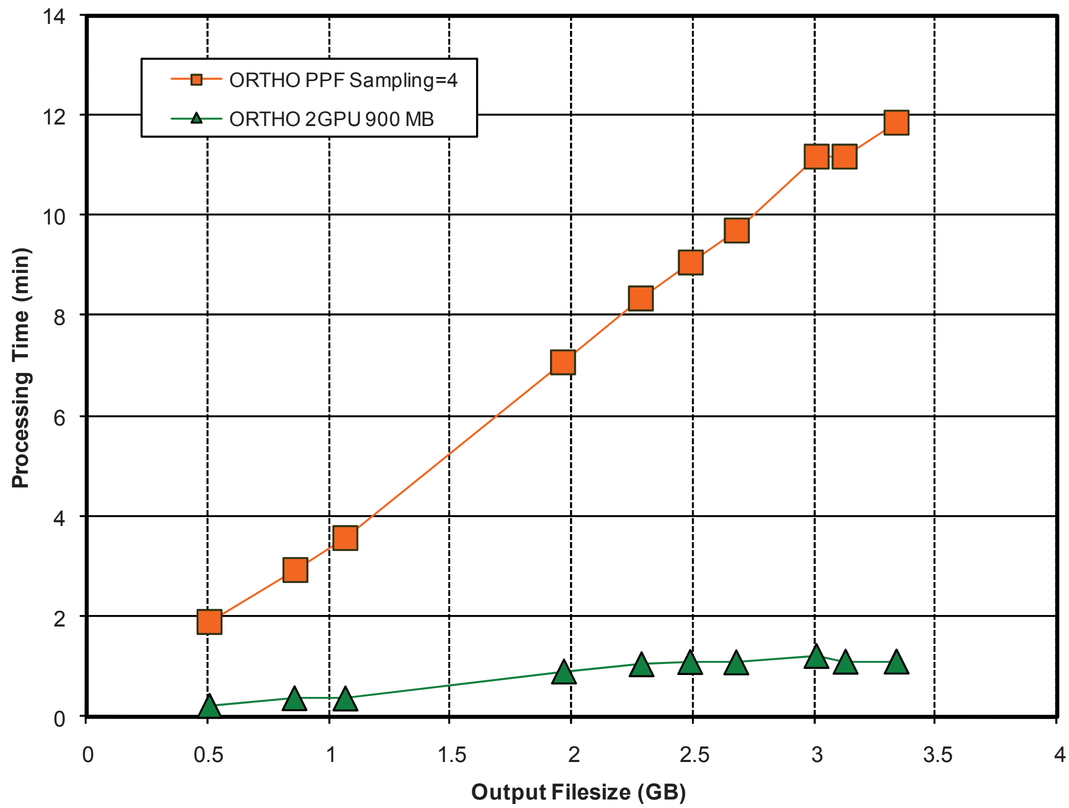

Figure 4:   Runtime comparison of ProSDK (PPF) (spacing = 4) vs. Ortho Module

3.2 Accuracy

To understand the purpose of the accuracy test, it is helpful to understand how orthorectification works. The orthorectification process can be summarized as follows:

a) A blank orthoimage covering an area of interest is created. For each pixel of this blank orthoimage, we know the Easting and Northing value because we have defined the orthoimage layout.

b) The height associated with each of these Easting and Northing values is interpolated from a DEM. In many cases, the Easting and Northing have to be converted to Longitude and Latitude in order to do this calculation.

c) The Easting, Northing, and Height values are used to compute which pixel in the source image would have imaged this position – this in general gives a set of non-integer row and column coordinates. In many cases, the Easting and Northing have to be converted to another projection in order to do this calculation.

d) Image interpolation is carried out to determine the pixel value at this fractional location. The interpolated pixel value becomes the ortho pixel value.

In order to speed up the calculations, there are thus two processing steps that may be carried out using approximations rather than rigorous calculations.

1. Map Transformation: Computation of the Latitude and Longitude coordinates (or coordinates in a different projection) corresponding to a given location in the orthoimage.
2. Math Model: Computation of the source pixel location given a ground coordinate.

Within the PCI ortho PPF, these two items are expressed via the 'sampling' parameter. For instance, in the PPF, a 'sampling' value of 4 instructs the software to perform rigorous calculations for every 4^{th} pixel and to perform bilinear interpolation in between. In the Ortho Module, these two steps can be controlled separately, but the effect is similar.

In general, it is valid to use a wide grid spacing for the map transformation because this transformation is a very smooth function. However, the accuracy of bilinear interpolation for the math model is highly dependent on both the image-to-ground viewing geometry and on the terrain roughness.

To assess the amount of error introduced by sampling rather than rigorous calculation, various different grid spacings were used for the map transformation and for the math model. The source pixel locations that were computed using this grid spacing were compared with another set of source pixel locations that had been computed rigorously for each pixel. For each spacing value, the resultant error reflects the positional error that has been introduced in the orthoimage.

The accuracy tests were repeated for several different datasets. The results are summarized in the figures below.

Figure 5 and Figure 6 illustrate the effect of varying the grid spacing of the map transformation. It can be seen that the error introduced by interpolating the map transformation value is negligible; the worst-case error is no more than 0.005 pixels in any of the tests. This is good news, because the map transformation can be a very computationally-intensive operation. By default, the Ortho Module has thus been configured to use a spacing of 100 pixels by default.

Figure 7 and Figure 8 illustrate the effect of varying the grid spacing of the math model. It can be seen that this error can grow quickly in some cases. Even with a DEM post spacing of approximately 90 metres, the Vanc_05 dataset shows positional errors up to 4 pixels when the grid spacing is 16 pixels (approximately 8 metres). Normally, users of OrthoEngine (Geomatica desktop software) will employ a grid spacing of 4 pixels; in this case, the maximum error is still significant, at 1.5 pixels. Although these positional errors would be restricted to limited areas with rapidly varying height, it makes it difficult to guarantee that the generated product will meet a specified level of accuracy. For that reason, the Ortho Module performs the math model calculation for every pixel of the ortho, with no interpolation.

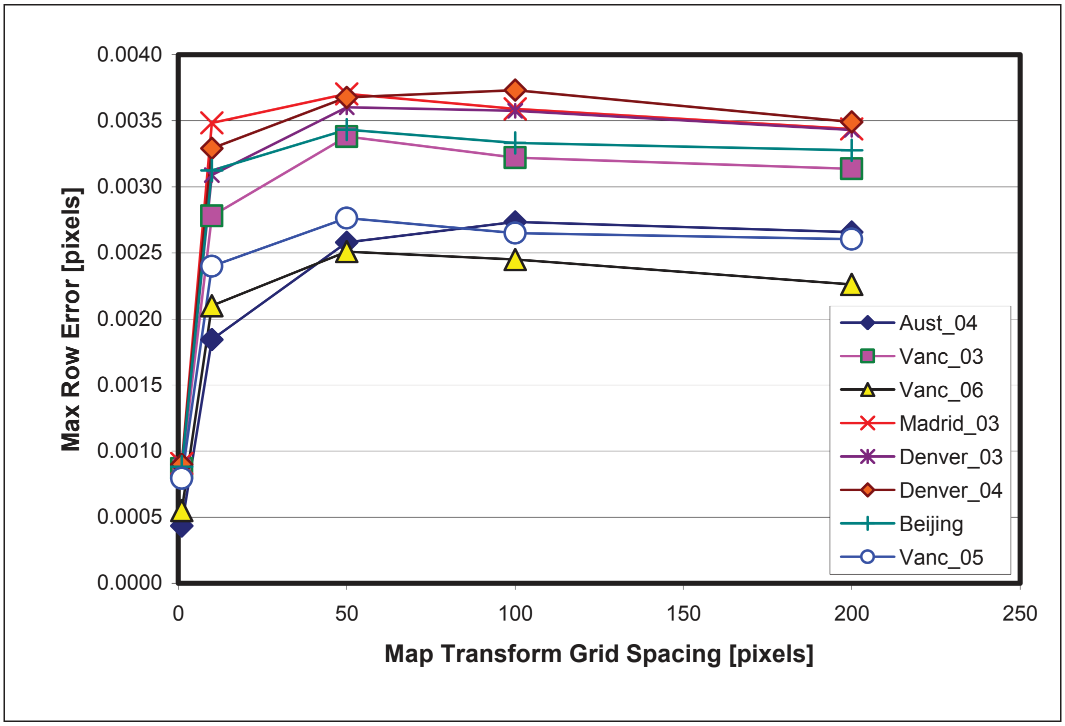

Figure 5: Row error vs. map transform grid spacing

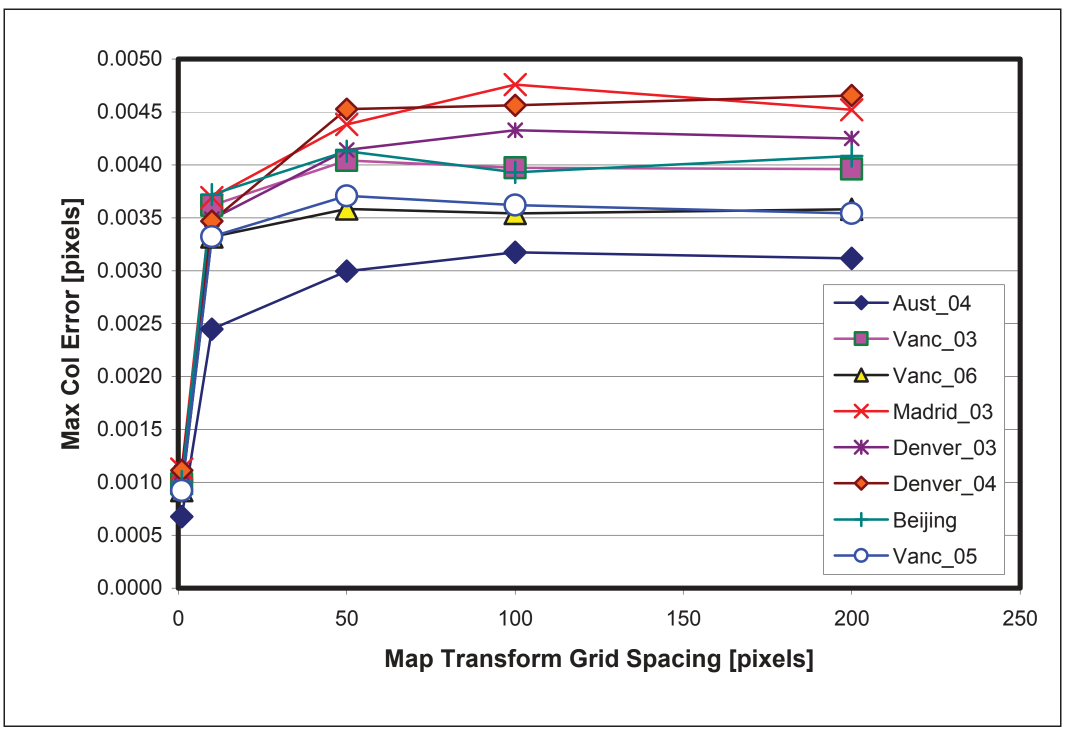

Figure 6: Column error vs. map transform grid spacing

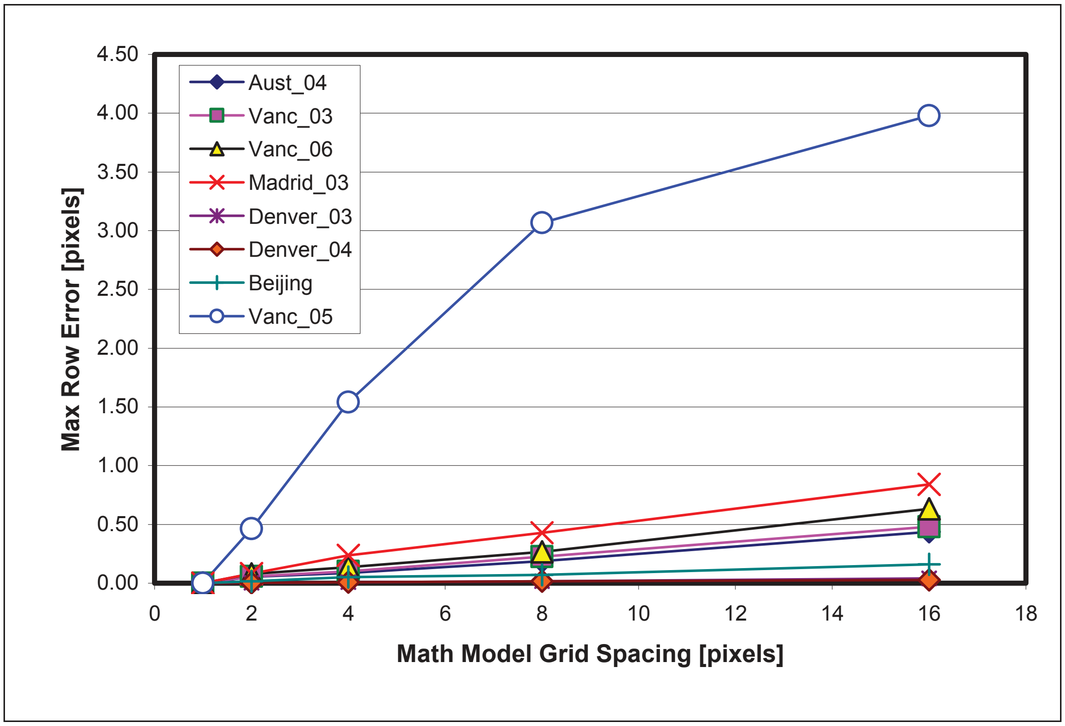

Figure 7: Row error vs. math model grid spacing

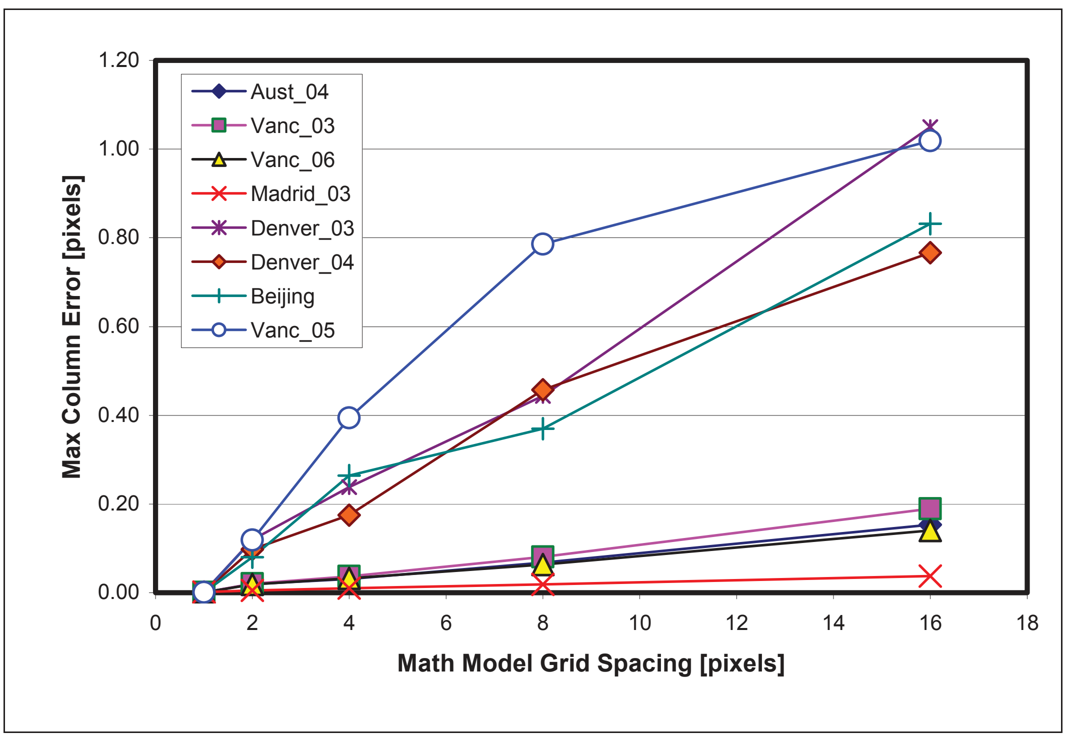

Figure 8: Column error vs. math model grid spacing

Appendix A and Appendix B contain more detailed plots illustrating the errors for individual scenes.

In addition to these accuracy tests, the orthos generated by the Ortho Module were visually compared with orthos generated by the PCI ortho PPF. When using a 'sampling' value of 1 in the ortho PPF, the resultant orthos were confirmed to be consistent to the sub-pixel level.

3.3 Performance Tuning

As alluded to in the previous sections, the orthorectification module contains a number of tuneable parameters that may be modified by the user at runtime. These parameters include the following:

1. The number of GPUs used
2. The amount of memory available for use on the GPU and on the host
3. The allocation of particular processing steps to the GPU or to the host

The objective of performance tuning is to find the combination of parameters that produces the maximum data throughput. At the same time, we can discover how sensitive the system is with respect to the different parameters.

Each of the performance tuning tests was run on eight of the primary test scenes. In each case, the test would be run by setting the tuning parameter to a specific value, running all eight scenes at that value, and then setting the tuning parameter to the next value. If the test had been run by cycling through all the parameter settings in turn for each scene, the operating system would have cached at least part of the source file, thus affecting the computed result.

3.3.1 Memory Usage

The user can instruct the Ortho Module to limit its memory usage (memory used for raster buffers) to a specified amount. It is possible to specify a limit for both the host memory and the GPU memory used; the module will allocate raster buffers that are small enough to fit within the more restrictive limit.

The purpose of this test was to answer two questions:

a) How does the performance vary with the amount of memory used?
b) What amount of memory yields the best performance?

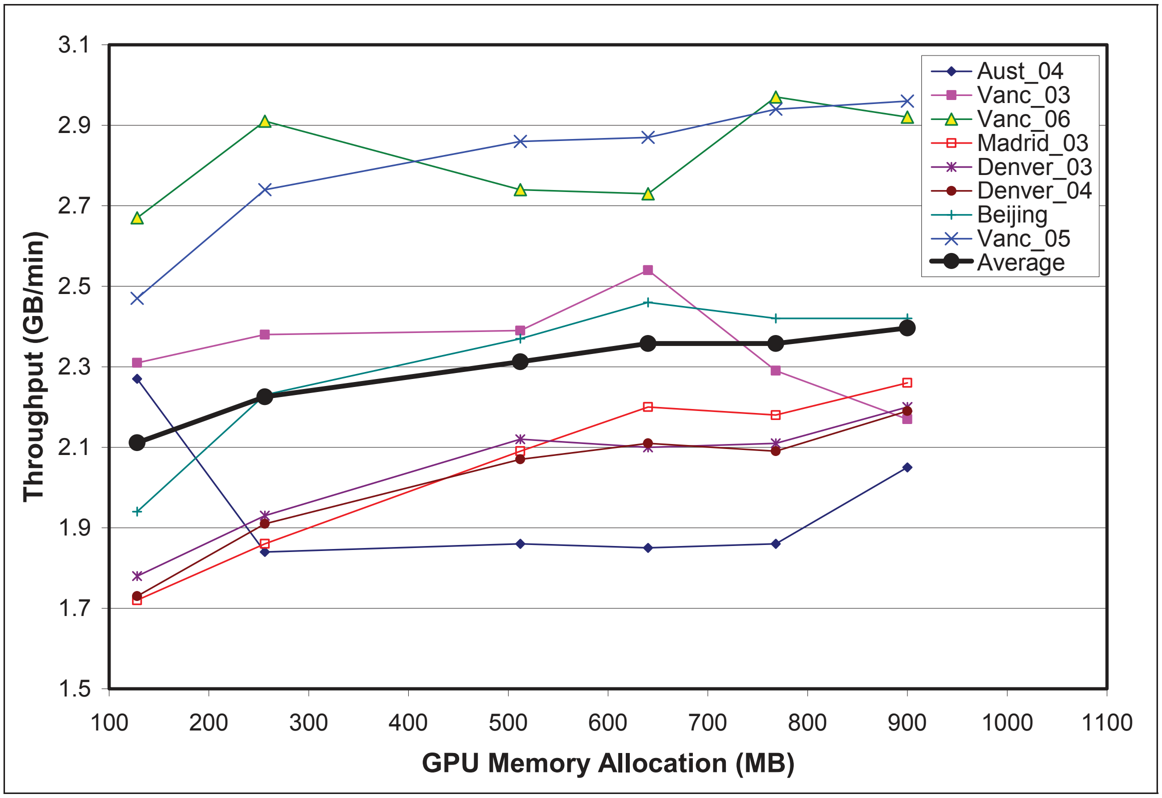

Figure 9: System throughput vs. memory usage

Looking at the average trend, it can be seen that the system is not highly sensitive to the amount of memory allocated. However, there is a clear trend toward increased performance with increased memory usage. The system has been configured to use 900 MB of GPU memory by default.

3.3.2 Number of GPUs Used

The machine contains two GPUs which can be used in parallel for orthorectification. The purpose of this test is to answer the question:

a) Does it help (for this application) to have two GPUs rather than one?

If, for instance, the system performance is limited entirely by disk performance, it may not be helpful to use both GPUs.

To do this test, the datasets were processed first using GPU 1, then using GPU 2, then using both GPUs together. The results were interesting:

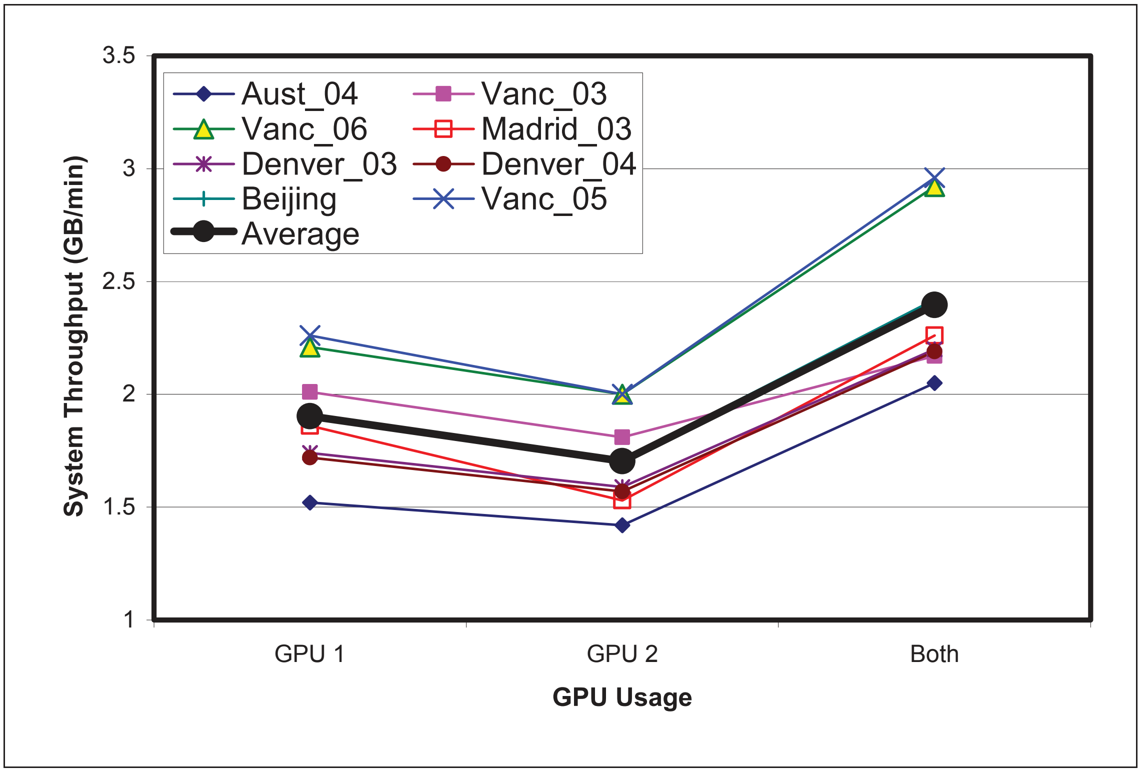

Figure 10: System throughput vs. GPU allocation

On average, use of two GPUs gives a performance boost of approximately 25% when compared to using just one. However, if a single GPU is used, GPU 1 is consistently faster than GPU 2. The reason for this finding is not known.

3.3.3 Processing Strategy

The Ortho Module can be configured to run various combinations of steps on the host and on the GPU. This is specified by selecting a particular "strategy" when processing the ortho. There are five different strategies, as outlined in Table 6.

Table 6: Processing Strategies

Strategy	**Description**
HOSTONLY	All processing is done on the host.
OMP	Several processing steps, except for math model calculations, are done on the GPU. OpenMP is used to associate processing threads with particular GPUs.
TP	Several processing steps, except for math model calculations, are done on the GPU. A custom thread pool is used to associate processing threads with particular GPUs.
OMP_ACCEL	Like OMP, except that math model calculations are also done on the GPU.
TP_ACCEL	Like TP, except that math model calculations are also done on the GPU.

The 'custom thread pool' used in the TP and TP_ACCEL strategies is implemented as a class CudaJobRunner which maintains a processing thread separate from the main thread. For connection to two different GPUs, the system would keep two of these CudaJobRunner objects and 'bind' each one to its respective GPU. The binding is done by calling a SelectDevice() function in the processing thread. The thread will remain bound to the selected device until it goes out of scope or SelectDevice() is called with a different device ID. The TP implementations were included because SelectDevice() takes a finite time to run and it does not appear to be guaranteed that OpenMP will always keep the same thread alive between processing loop iterations (thus requiring SelectDevice() calls before each CUDA function). The TP implementations use pthreads, allowing the developer to know exactly which threads are bound to which devices.

The question to be answered by this test was the following:

a) Which processing strategy is the fastest?

Results from testing all strategies with the test scenes are shown below.

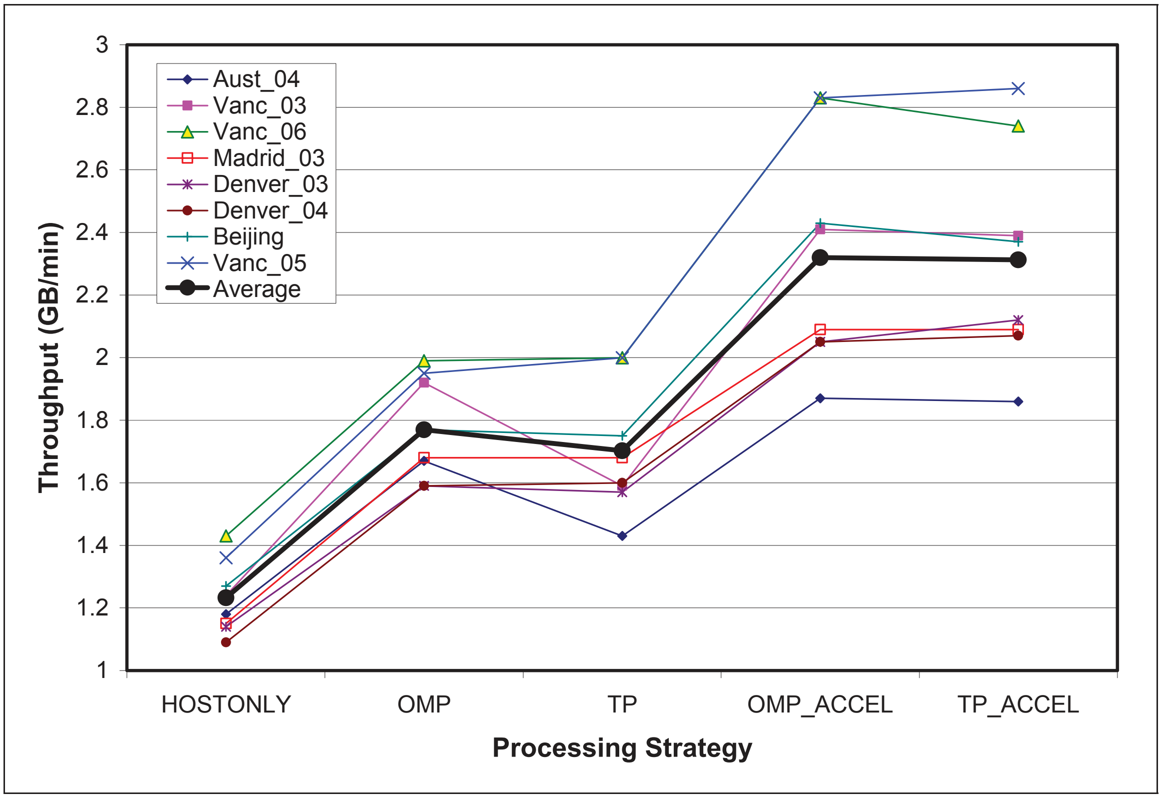

Figure 11: Results from different processing strategies

As expected, host-only processing was slowest while the _ACCEL strategies were fastest. The TP implementation was sometimes slightly slower than OMP, but we found that for some reason, OMP (and OMP_ACCEL) would fail with out-of-memory errors when 900 MB of GPU memory was requested. The TP and TP_ACCEL implementation, despite ultimately using the same memory (and the same function calls), could successfully use more memory. Thus, TP_ACCEL has been chosen as the default processing strategy.

3.4 Specific Optimizations

During development of the Ortho Module, numerous techniques were used to increase the processing speed, and the achieved performance figures reflect the combination of all of these things. The purpose of this test was to investigate how much impact each different optimization had on the final result.

In general, the sequence of algorithmic improvements proceeded as follows:

1. The code was rewritten so that the same grid spacing is not required for the map transform and math model transform.
2. The raster I/O library was improved.
3. To minimize disk contention, different hard drives were used for input and output data.
4. The code was compiled in multithreaded mode, enabling the interleaving of I/O and

processing as well as loop parallelization

5. Most of the per-pixel processing steps (except math model calculations) were moved to the GPU
6. Math model calculations were moved to the GPU.

Results from the different optimizations are shown in Table 7 below. That table shows the result when using two GPUs; results are shown for the single-GPU case in Table 8. The tests yielded some interesting observations:

- Improving the raster I/O library reduced the runtime by over two minutes, even in single-threaded mode. If there is no concern about running out of disk space, it is not necessary to initialize the full output file (i.e., write zeroes to the whole file) before processing. This saved about 30 seconds in our implementation. It is also very important to have a library that does not access the disk when it doesn't need to! For instance, if the image is stored in tiles and we wish to overwrite an entire tile, there is no need to first read it from the disk.
- It is not reflected in the tables, but multithreaded processing is helped a lot by using different hard drives for input and output. This is because input and output operations happen simultaneously in multithreaded mode, potentially causing disk contention.
- Even while processing entirely on the host, performance is helped enormously by intelligent use of OpenMP multithreading. For this scene, processing time was reduced by 5.5 minutes.
- Moving all of the operations possible onto the two GPUs further reduced processing time by another 60 seconds. We found during implementation that it was best to combine several successive processing steps into one CUDA function so that memory copies to and from the GPUs were minimized; this made a difference of about 15% in runtimes with respect to the non-combined processing steps (i.e., datasets that were running at about 2.0 GB/min were improved to 2.3 GB/min).
- As seen in previous tests, it does help to have two GPUs rather than one, but the improvement in performance is not dramatic. This is because the system is partially limited by I/O performance; some blocks of the image (not all) are delayed while waiting for I/O to finish. Having a faster processor does not help in that case.

Table 7: Progressive performance improvement for 3083 MB Beijing scene (2 GPUs)

Run	Strategy	Grid Spacing	Threading	Raster I/O	Same I/O dir?	Time (sec)	Time Speedup	Speedup w.r.t. Base
1	SDK/Py	4	Single	PIX	Yes	706.36		
2	HOSTONLY	1	Single	TiffSlow	Yes	608.59	97.77	1.2x
3	HOSTONLY	1	Single	TiffFast	Yes	486.32	122.27	1.5x
4	HOSTONLY	1	Single	TiffFast	No	475.71	10.61	1.5x
5	HOSTONLY	1	Multi	TiffFast	No	141.48	334.23	5.0x
6	2 GPUs (No Math Model)	1	Multi	TiffFast	No	95.56	45.93	7.4x
7	2 GPUs (All)	1	Multi	TiffFast	No	79.74	15.82	8.9x

Table 8: Progressive performance improvement for 3083 MB Beijing scene (1 GPU)

Run	Strategy	Grid Spacing	Threading	Raster I/O	Same I/O dir?	Time (sec)	Time Speedup	Speedup w.r.t. Base
1	SDK/Py	4	Single	PIX	Yes	706.36		
2	HOSTONLY	1	Single	TiffSlow	Yes	608.59	97.77	1.2x
3	HOSTONLY	1	Single	TiffFast	Yes	486.32	122.27	1.5x
4	HOSTONLY	1	Single	TiffFast	No	475.71	10.61	1.5x
5	HOSTONLY	1	Multi	TiffFast	No	141.48	334.23	5.0x
6	1 GPU (No Math Model)	1	Multi	TiffFast	No	117.39	24.09	6.0x
7	1 GPU (All)	1	Multi	TiffFast	No	94.50	22.89	7.5x

Because the I/O and processing are interleaved in the multithreaded implementations, it can be difficult to separate the effect of optimizations in these two categories. To get a better idea of the true effect of the optimizations, the tests were run in single-threaded mode only, and the results are shown in Table 9.

Table 9: Progressive performance improvement for 3083 MB Beijing scene (1 GPU, single-threaded operation)

Run	Strategy	Grid Spacing	Threading	Raster I/O	Same I/O dir?	Time (sec)	Time Speedup	Speedup w.r.t. Base
1	SDK/Py	4	Single	PIX	Yes	706.36		
2	HOSTONLY	1	Single	TiffSlow	Yes	608.59	97.77	1.2x
3	HOSTONLY	1	Single	TiffFast	Yes	486.32	122.27	1.5x
4	HOSTONLY	1	Single	TiffFast	No	475.71	10.61	1.5x
6	1 GPU (No Math Model)	1	Single	TiffFast	No	230.54	245.17	3.1x
7	1 GPU (All)	1	Single	TiffFast	No	112.87	117.67	6.3

Another view of the effect of the different performance improvements (in the multithreaded processing cases) is shown in Figure 12. As mentioned above, this chart does not give an entirely fair assessment of the effect of raster I/O improvements, because the multithreaded implementation takes much greater advantage of them than does the single-threaded implementation. Each of the improvements results in a significant performance improvement; however, many users could be satisfied with the CPU-only implementation. This is good news for the prospect of incorporating OpenMP into PCI's COTS products.

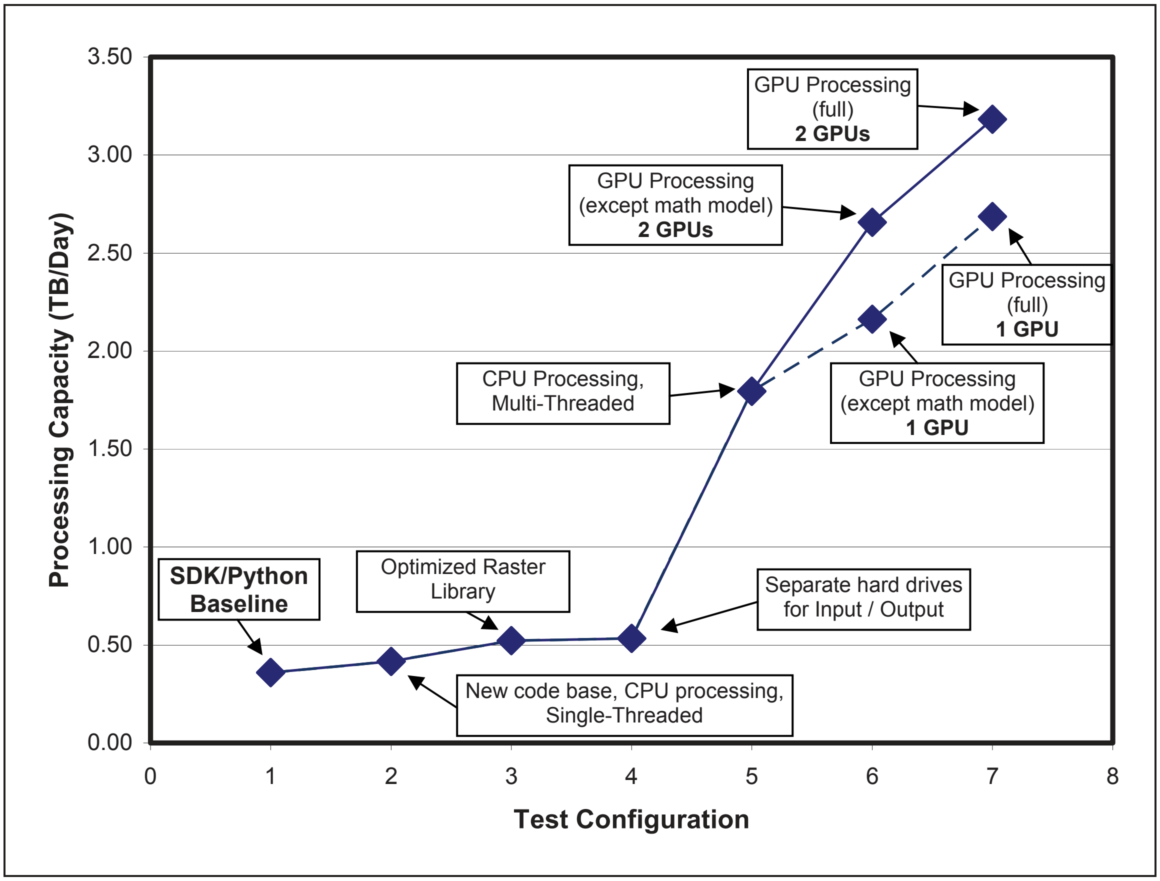

Figure 12: Successive performance improvements

3.5 Other Tests

3.5.1 Disk Bound or Compute Bound?

It is common to describe a processing module as either "compute bound" or "disk bound", meaning that its performance is hindered either by CPU performance or by disk performance. Having some idea of this characteristic allows the developer to decide where further optimization should be focused.

In order to evaluate whether this application is primarily disk bound or compute bound, block-by-block timing results from each of the processed orthos were collected. For each block of each ortho, one of the threads (I/O or computation) finishes first. The delay caused by the remaining process was added up to yield an overall delay figure for each scene. For each scene, both the total I/O delay and the total processing delay are thus computed.

The results shown in Figure 13, Figure 14, and Figure 15 thus indicate how much time is available to be gained by improving either I/O or computation speed for some of the different datasets tested. It can be seen from the results that performance improvements will be achieved if either aspect is improved. For 16-bit unsigned panchromatic datasets, there is more time to be gained by improving processing speed; for 16-bit unsigned pan-sharpened datasets, on the other hand, I/O delays have a larger influence.

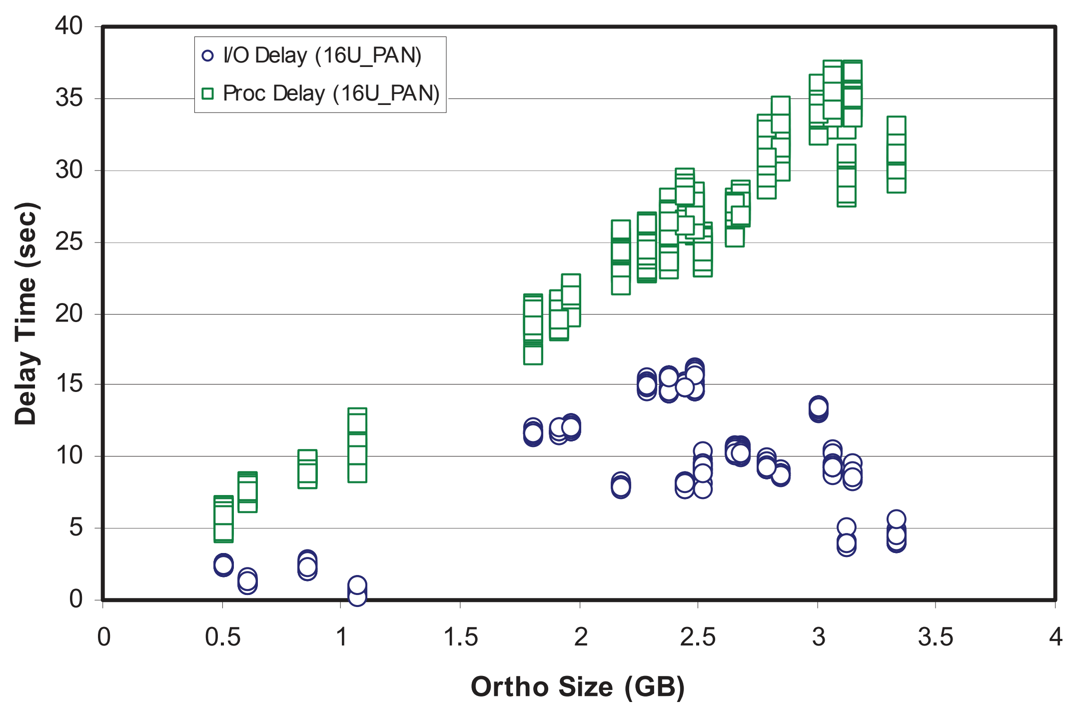

Figure 13: Processing and I/O Delays (16U_PAN)

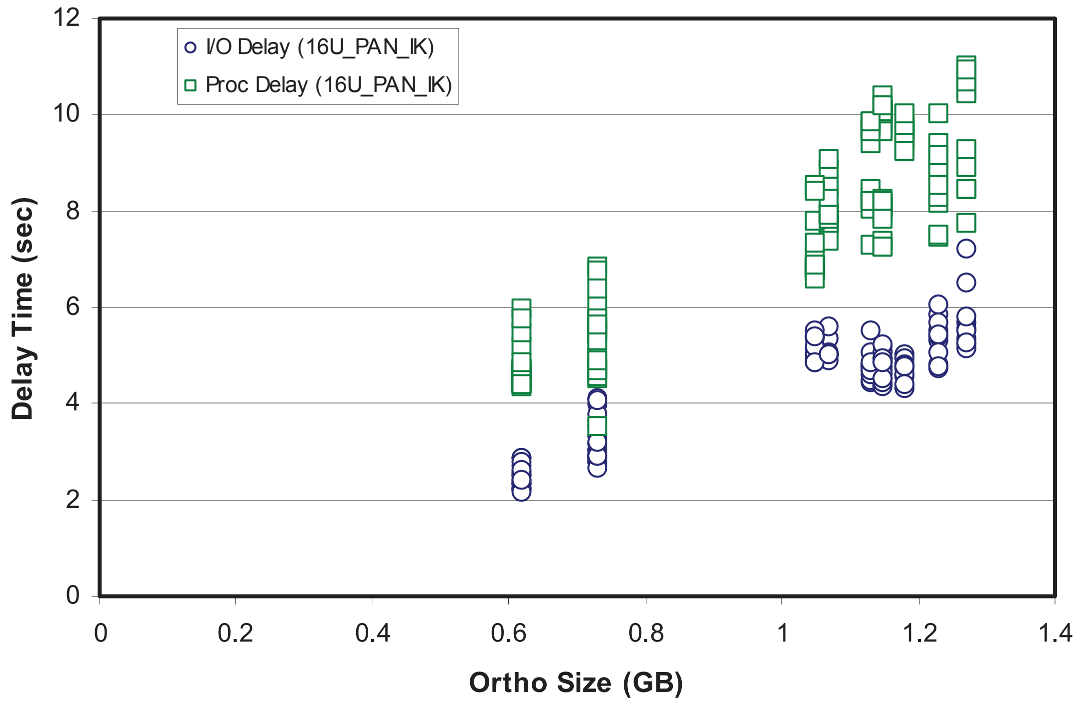

Figure 14: Processing and I/O Delays (16U_PAN_IK)

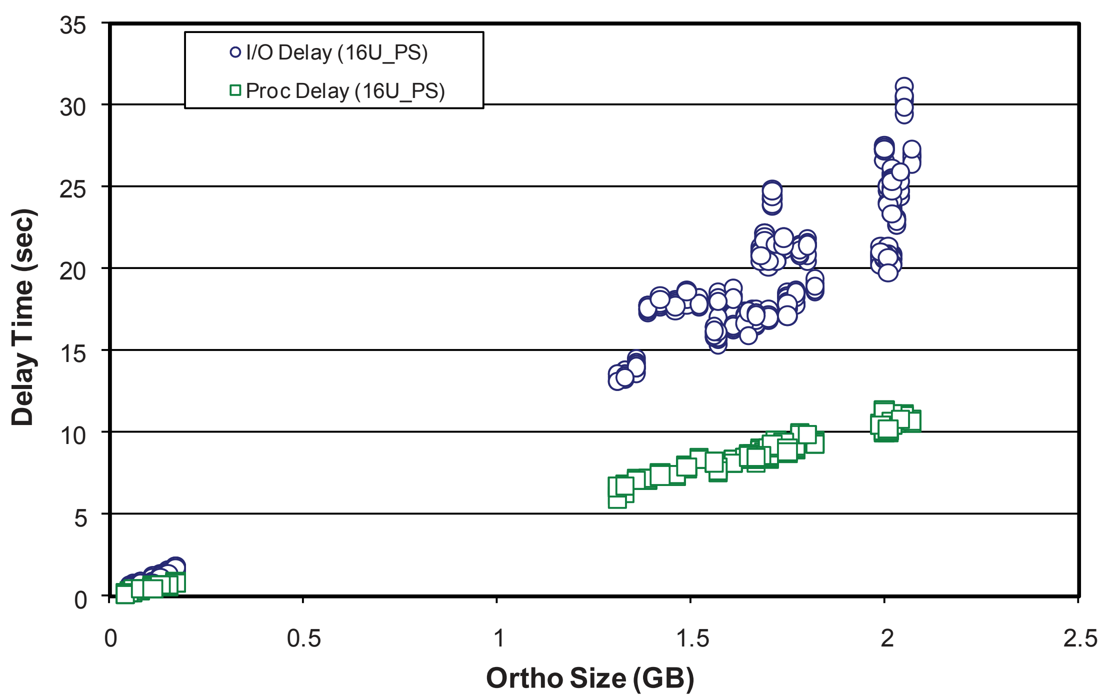

Figure 15: Processing and I/O Delays (16U_PS)

4 Appendix A. Lon/Lat Grid Plots

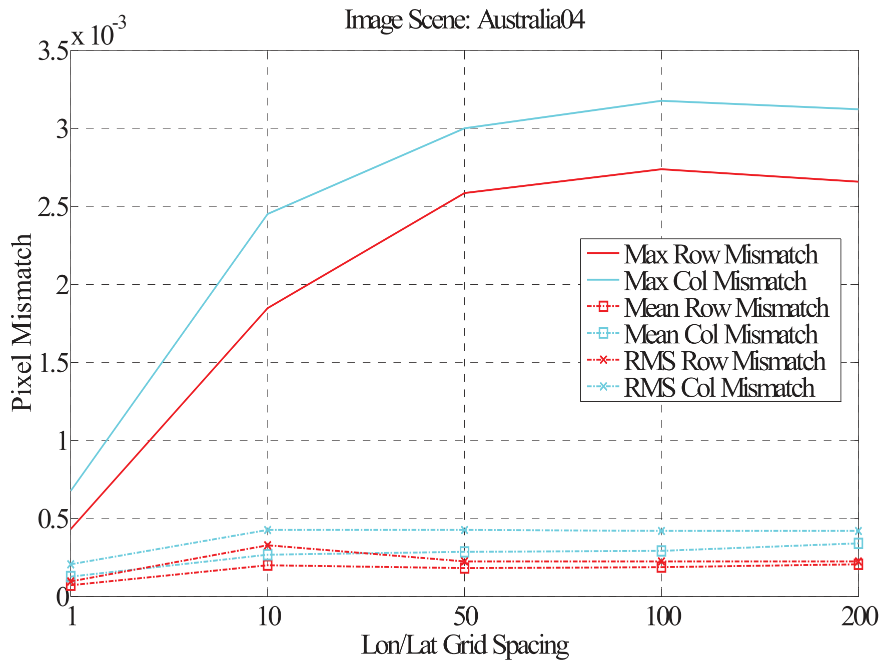

Fig. 1 Accuracy test (Lon/Lat) for Australia04

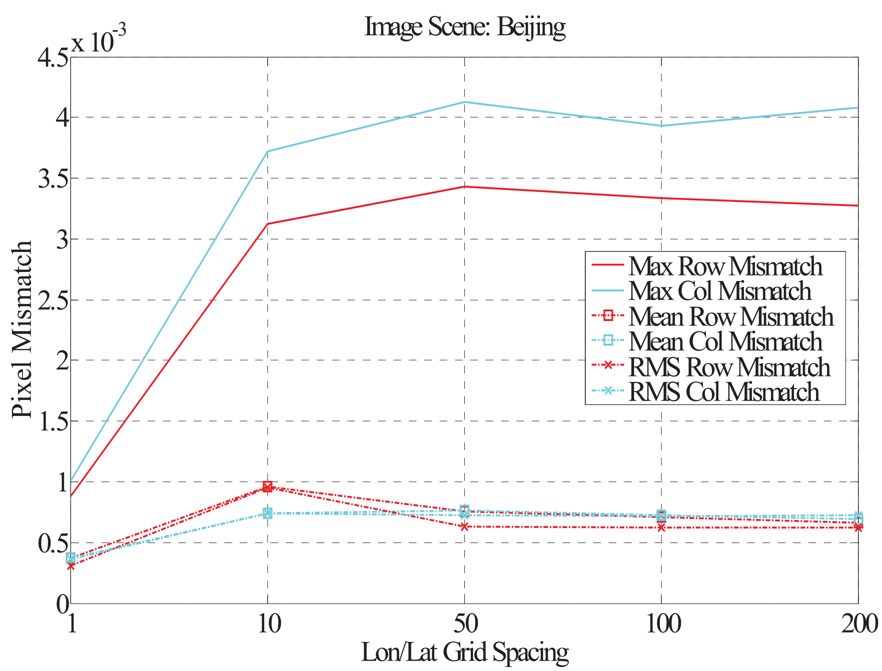

Fig. 2 Accuracy test (Lon/Lat) for Beijing

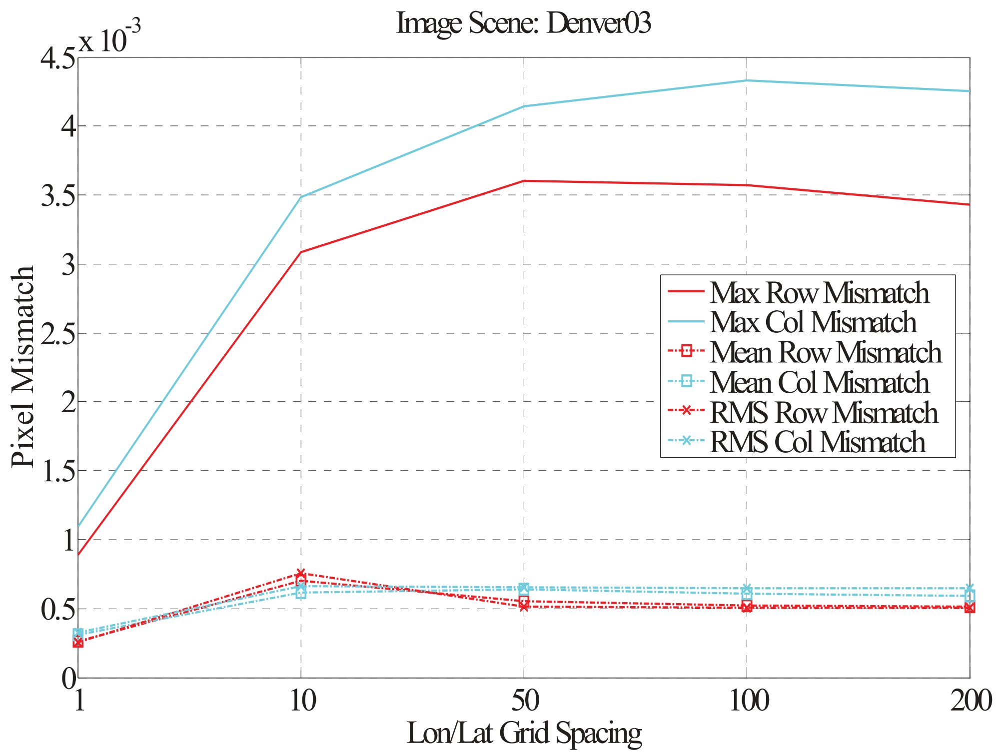

Fig. 3 Accuracy test (Lon/Lat) for Denver03

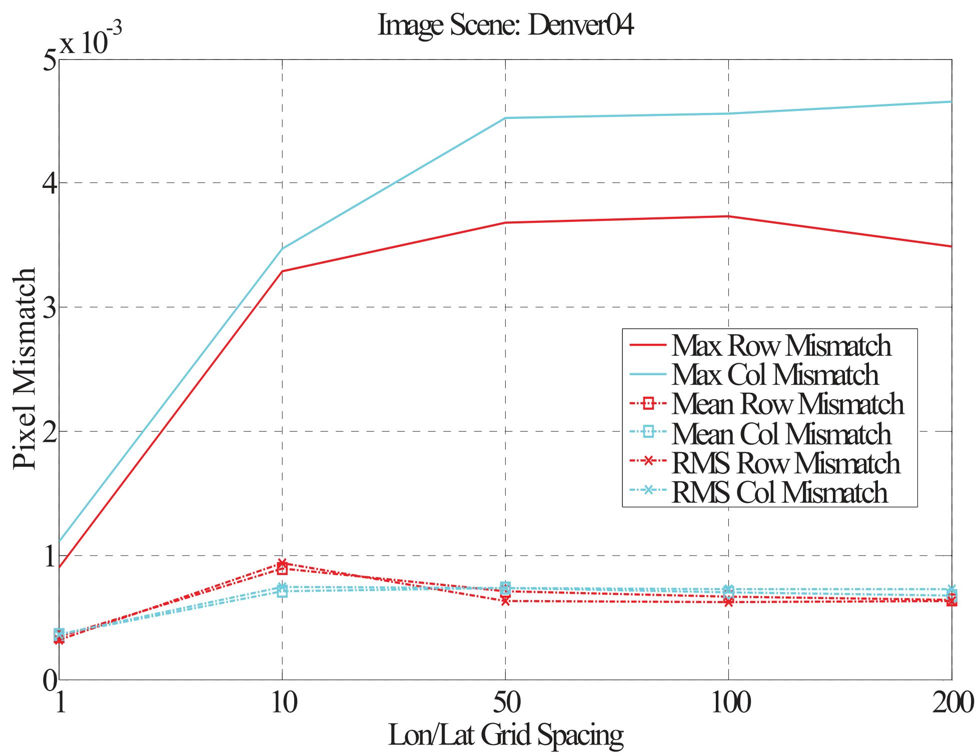

Fig. 4 Accuracy test (Lon/Lat) for Denver04

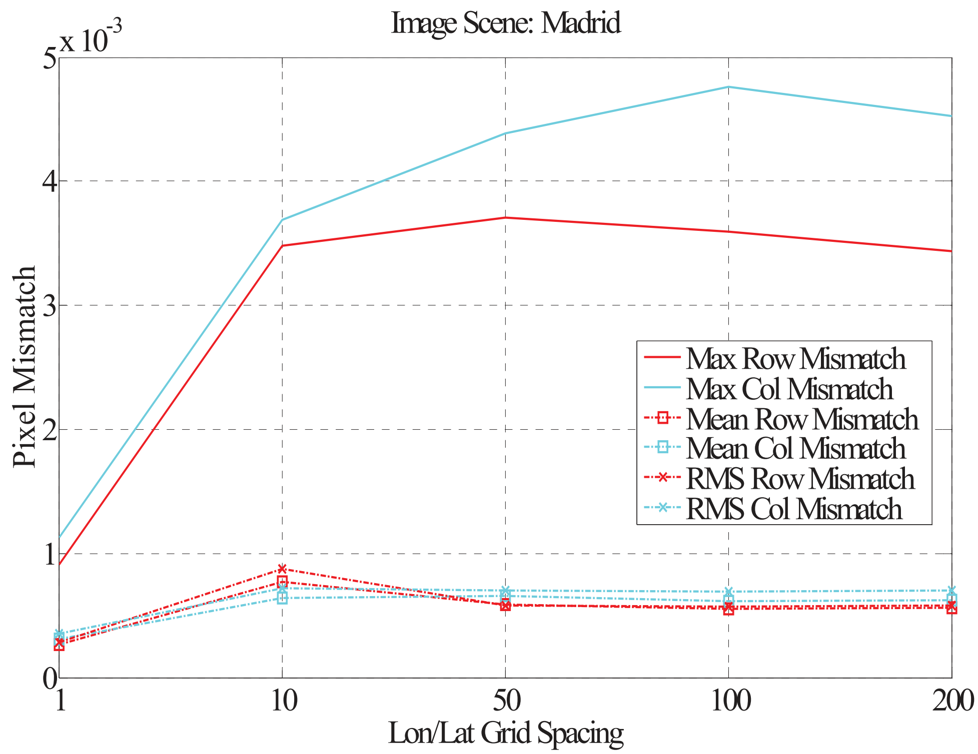

Fig. 5 Accuracy test (Lon/Lat) for Madrid

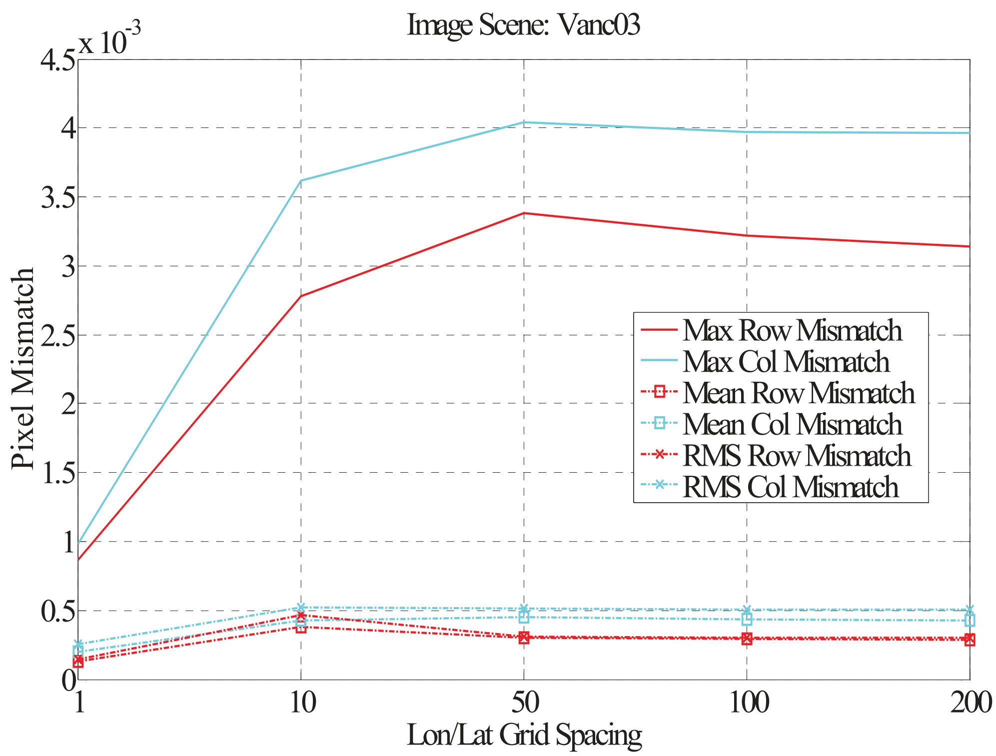

Fig. 6 Accuracy test (Lon/Lat) for Vanc03

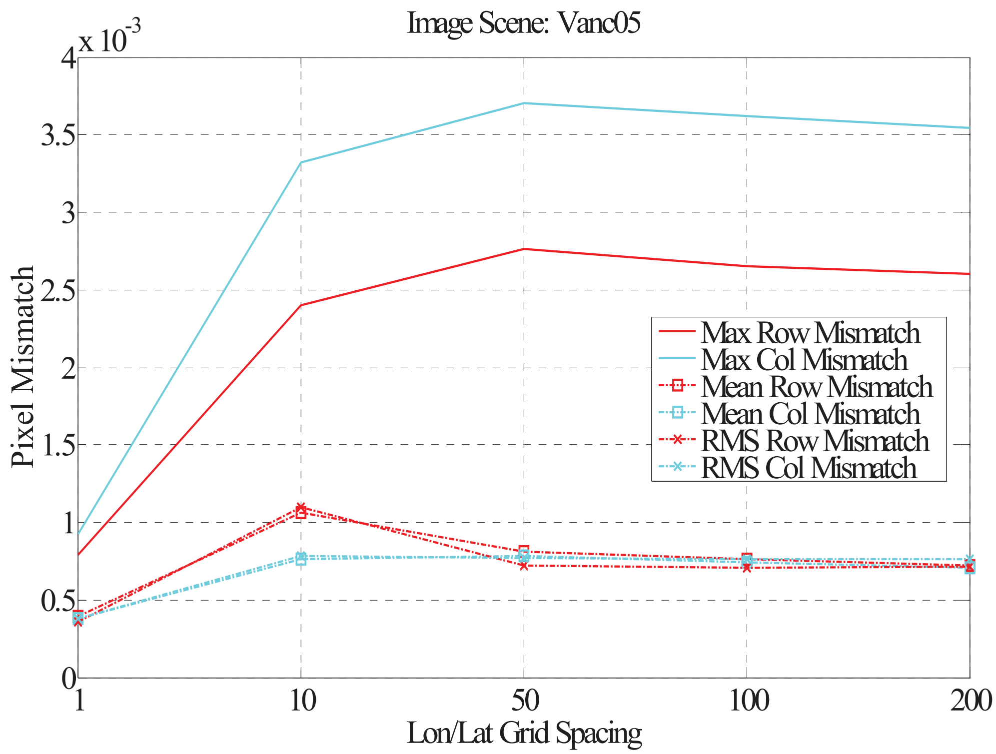

Fig. 7 Accuracy test (Lon/Lat) for Vanc05

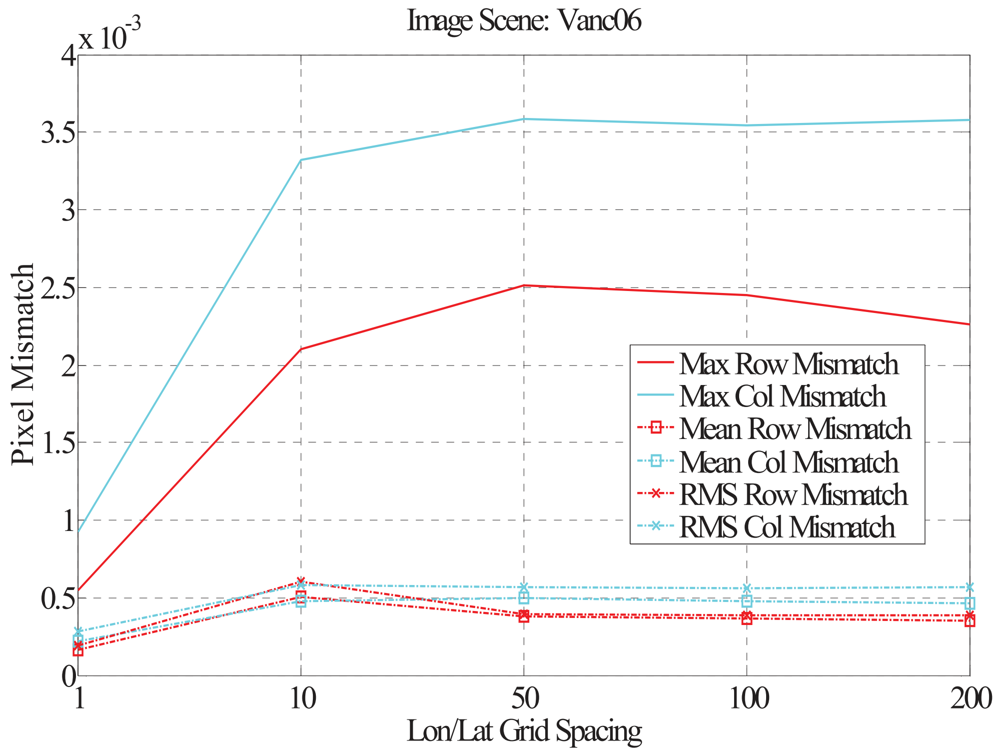

Fig. 8 Accuracy test (Lon/Lat) for Vanc06

5 Appendix B. Row/Col Grid Plots

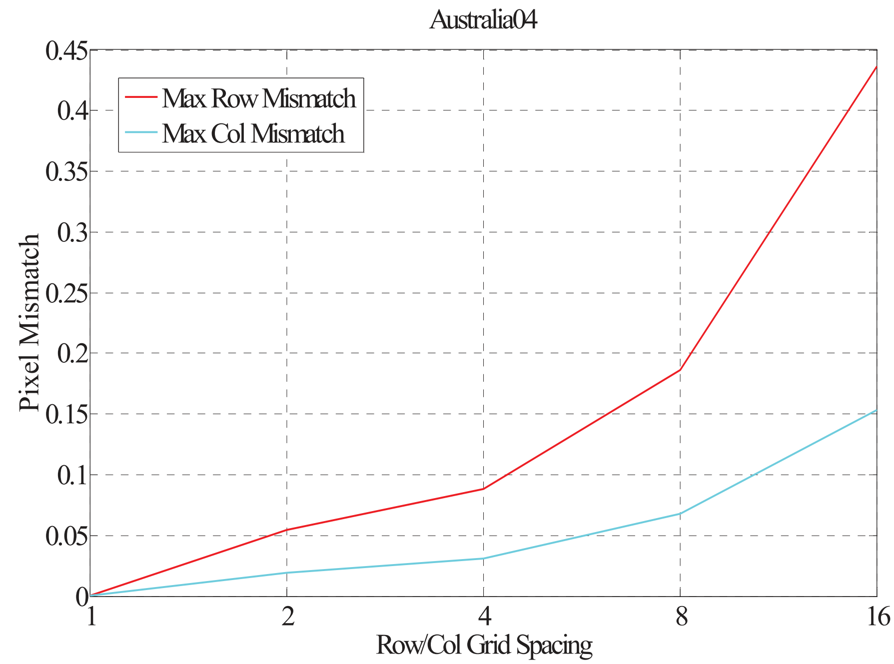

Fig. 9 Accuracy test (Row/Col) for Australia 04

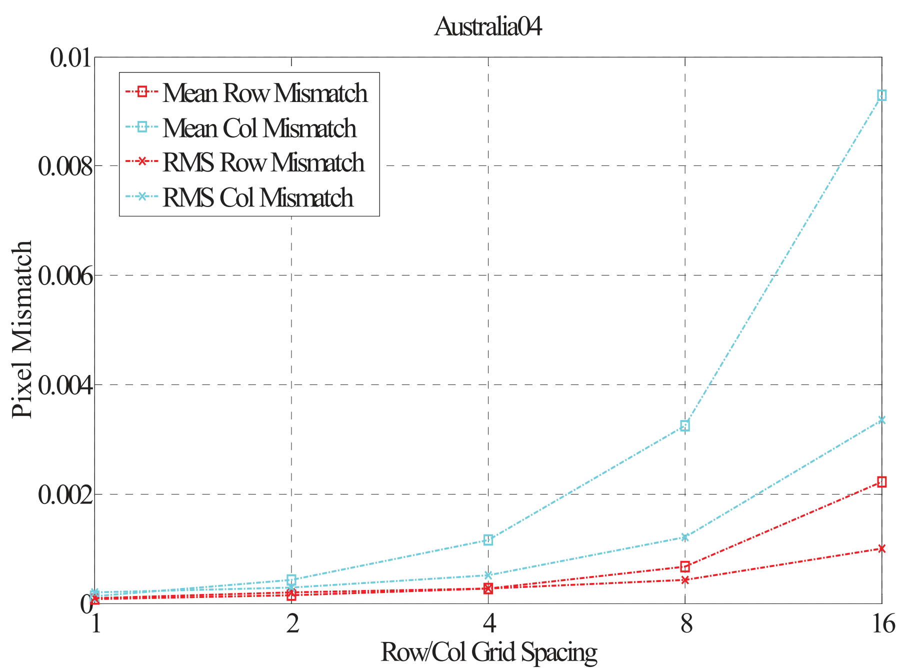

Fig. 10 Accuracy test (Row/Col) for Australia04

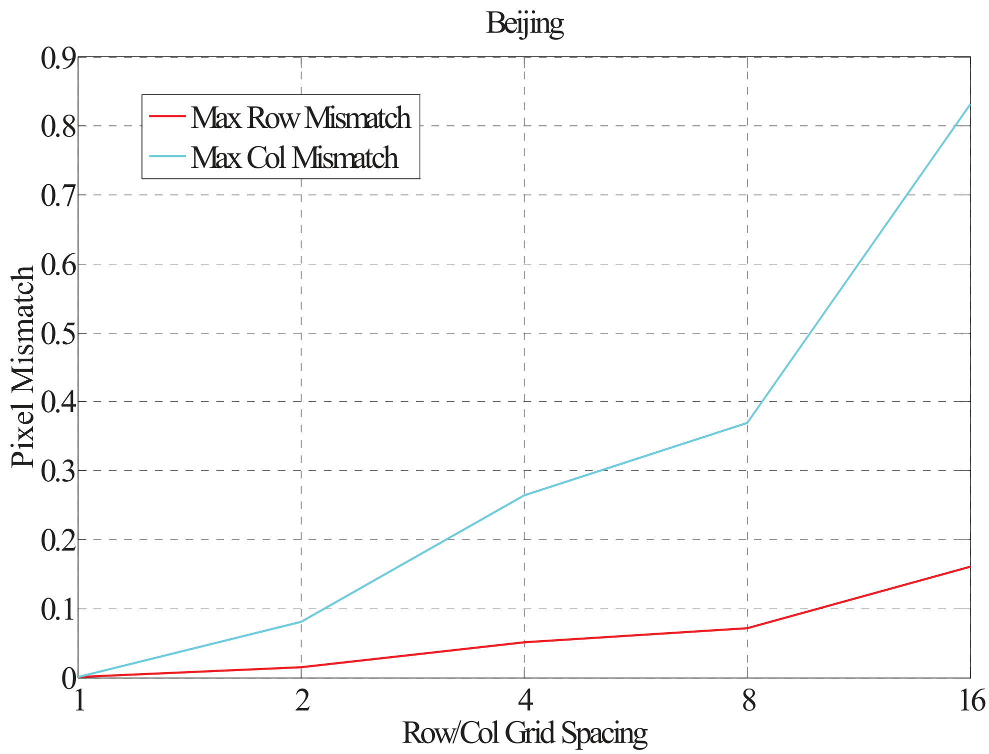

Fig. 11 Accuracy test (Row/Col) for Beijing

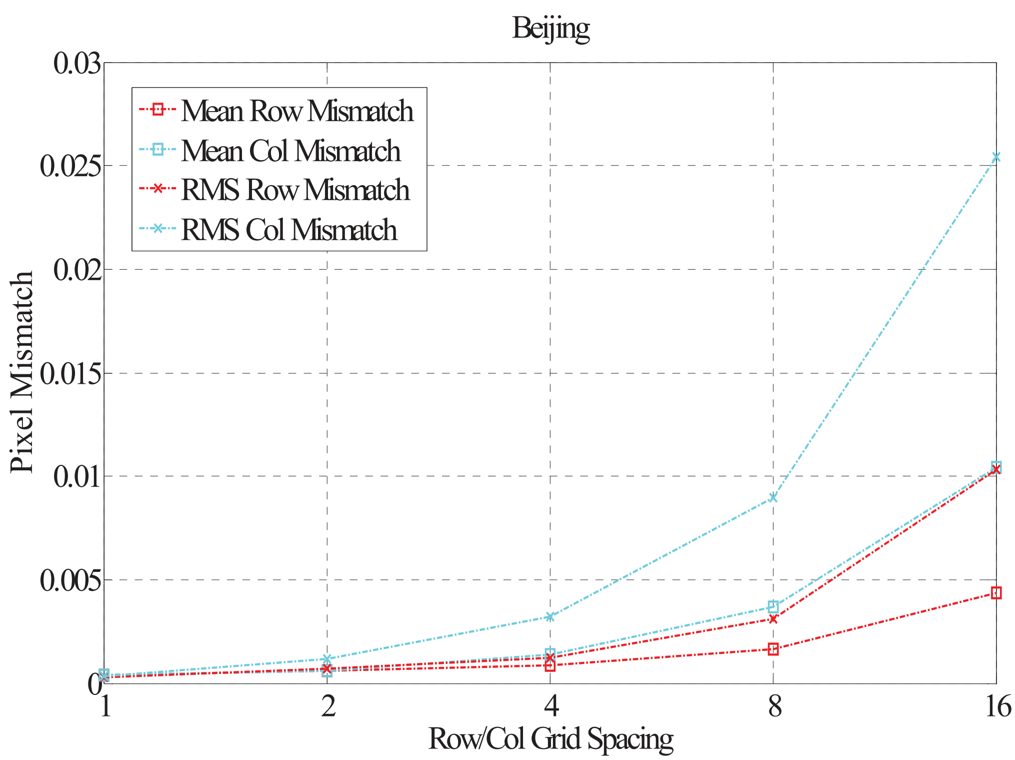

Fig. 12 Accuracy test (Row/Col) for Beijing

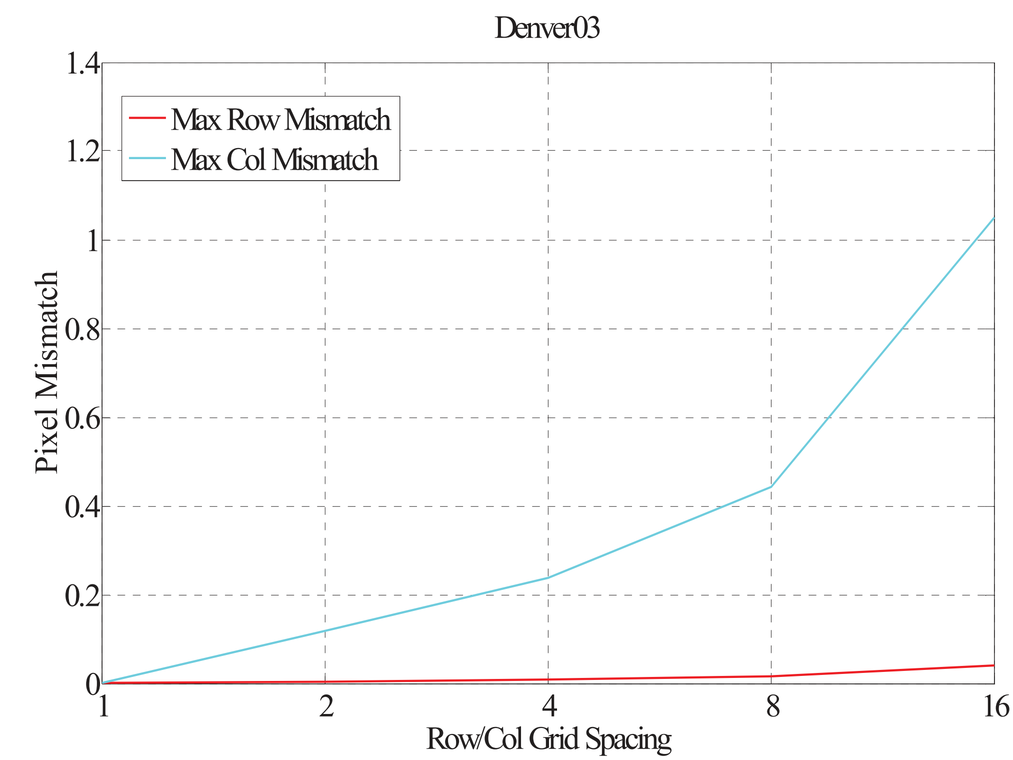

Fig. 13 Accuracy test (Row/Col) for Denver03

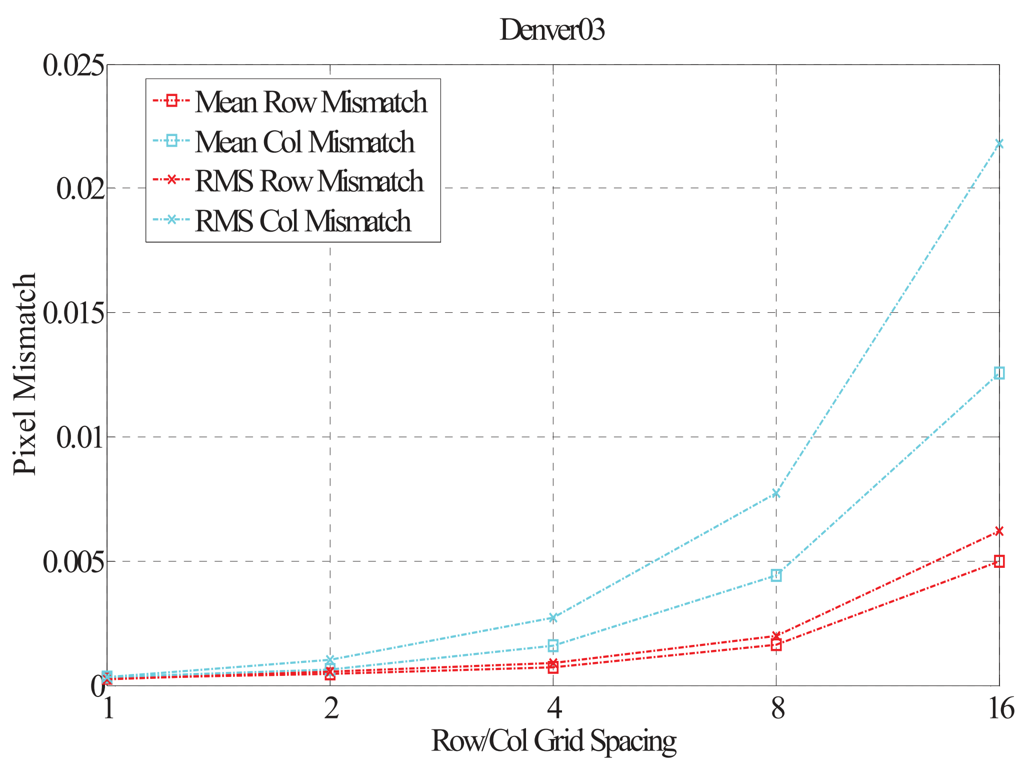

Fig. 14 Accuracy test (Row/Col) for Denver03

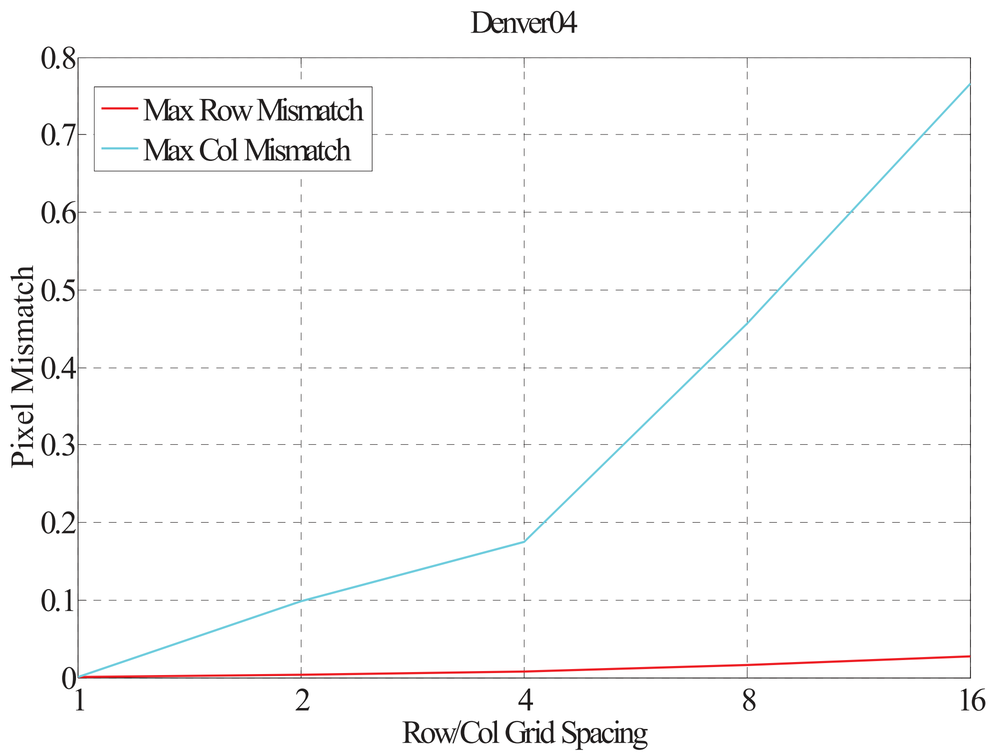

Fig. 15 Accuracy test (Row/Col) for Denver04

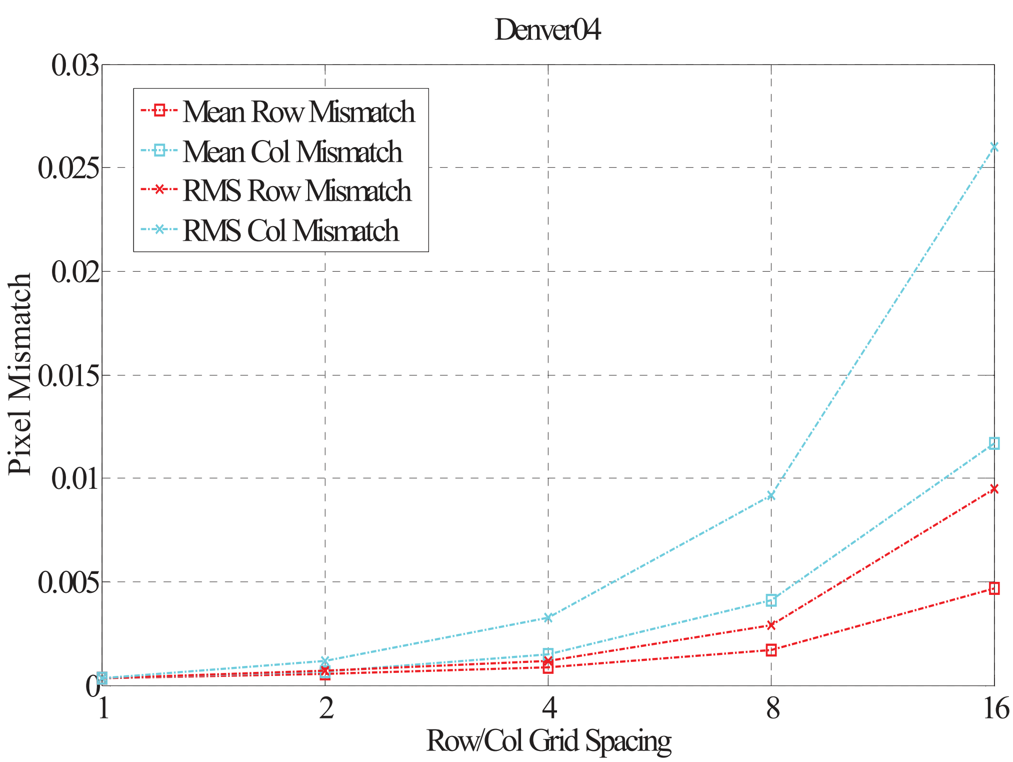

Fig. 16 Accuracy test (Row/Col) for Denver04

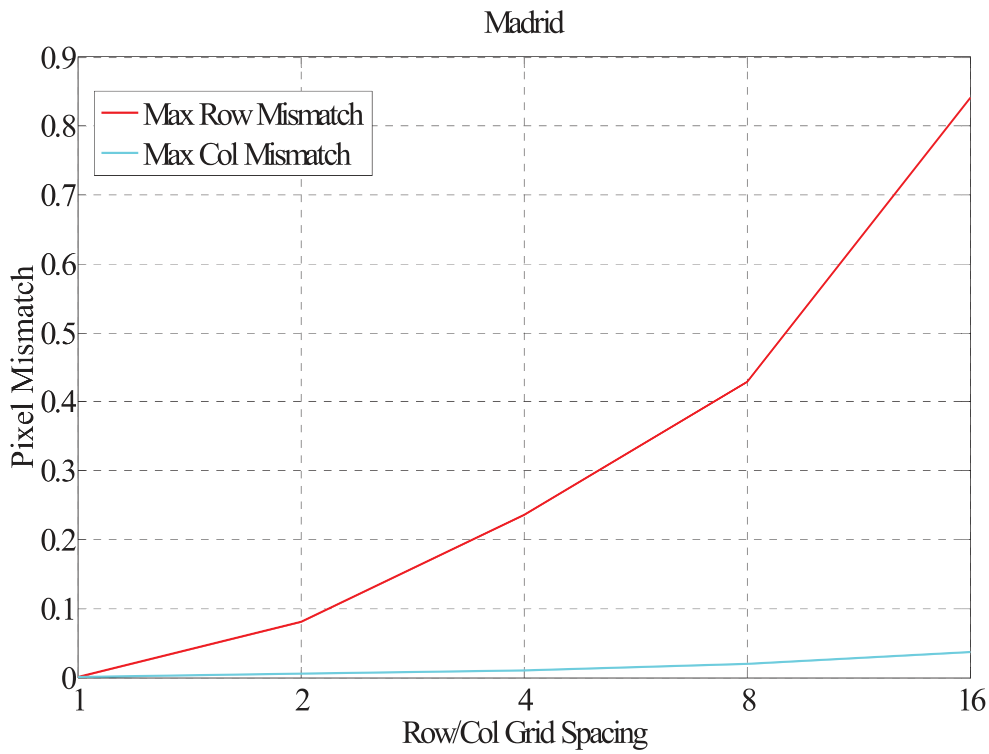

Fig. 17 Accuracy test (Row/Col) for Madrid

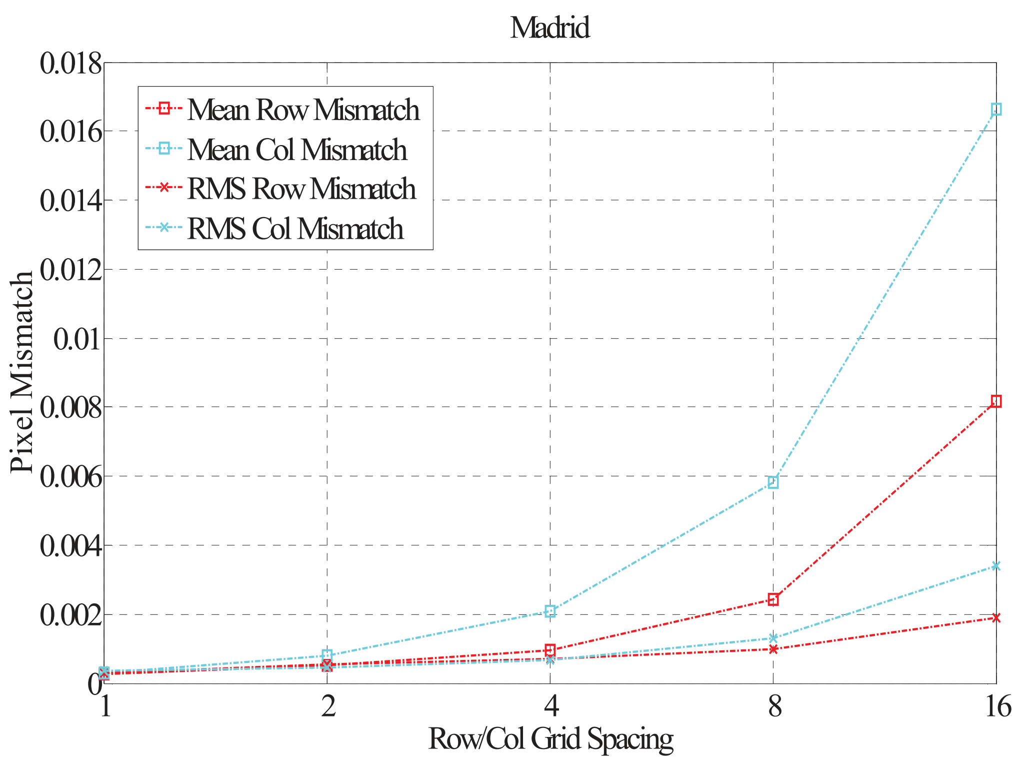

Fig. 18 Accuracy test (Row/Col) for Madrid

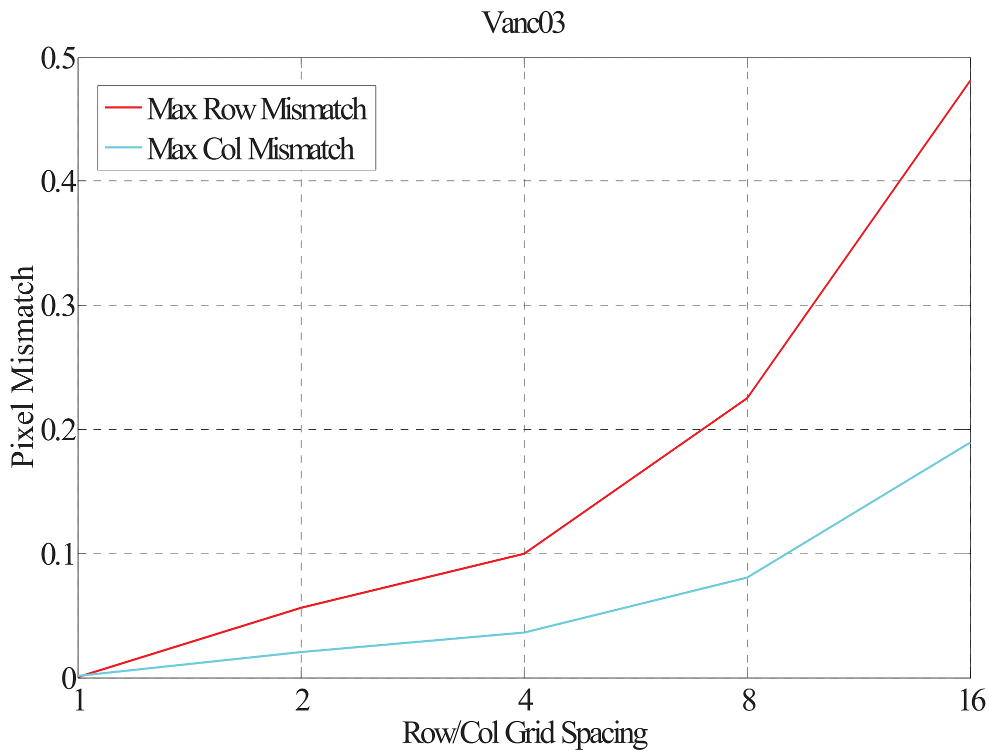

Fig. 19 Accuracy test (Row/Col) for Vanc03

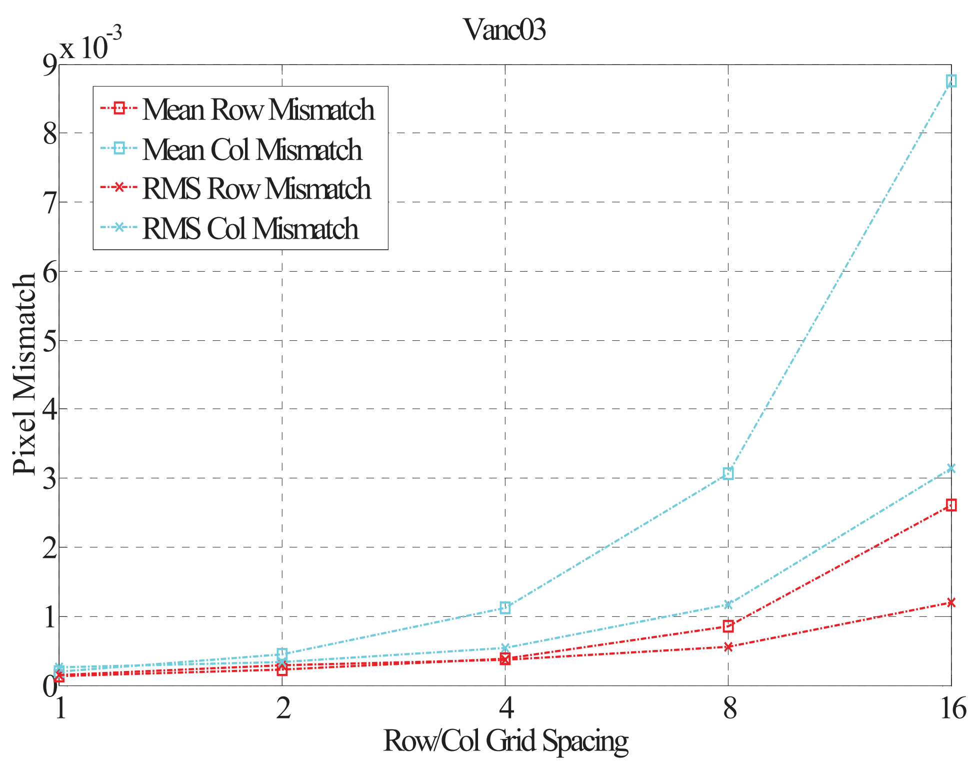

Fig. 20 Accuracy test (Row/Col) for Vanc03

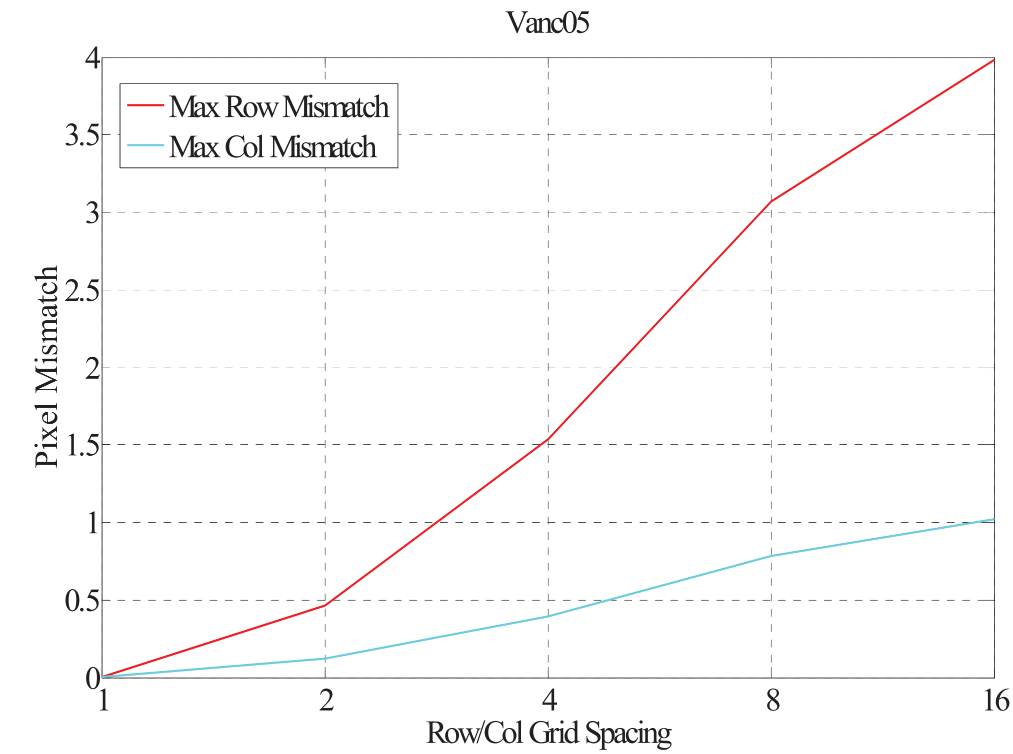

Fig. 21 Accuracy test (Row/Col) for Vanc05

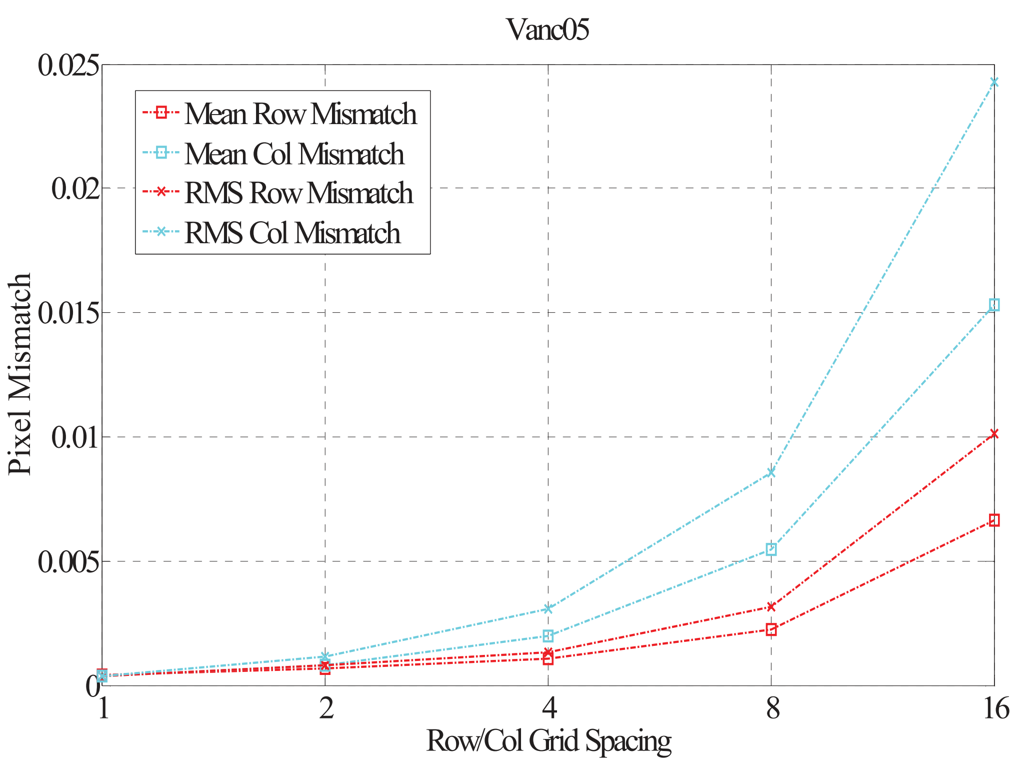

Fig. 22 Accuracy test (Row/Col) for Vanc05

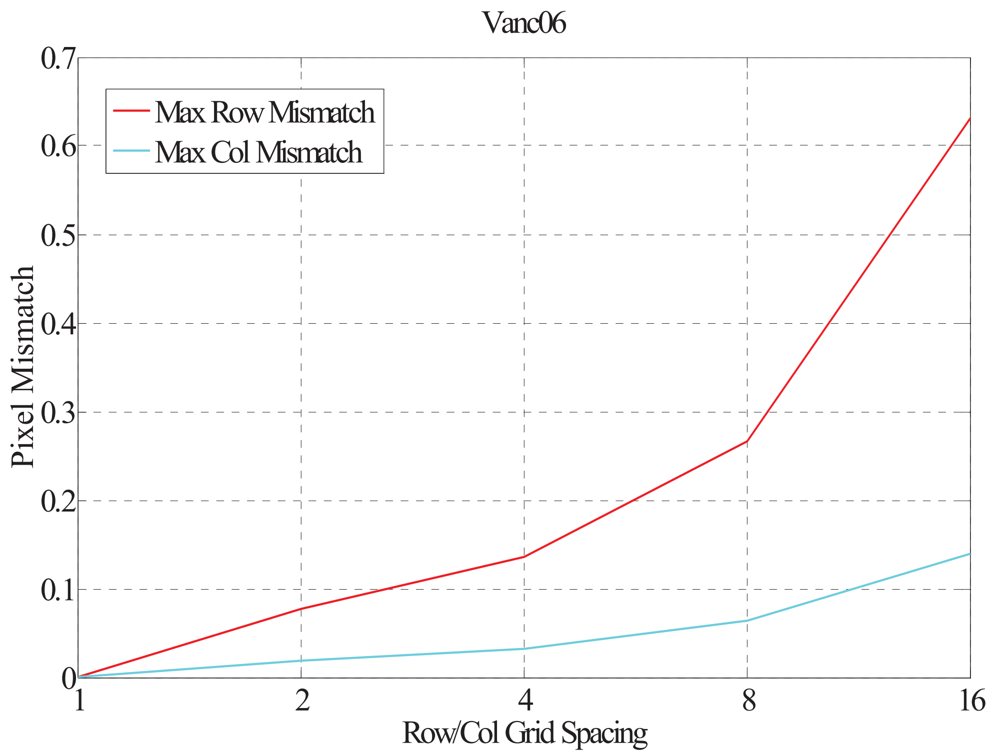

Fig. 23 Accuracy test (Row/Col) for Vanc06

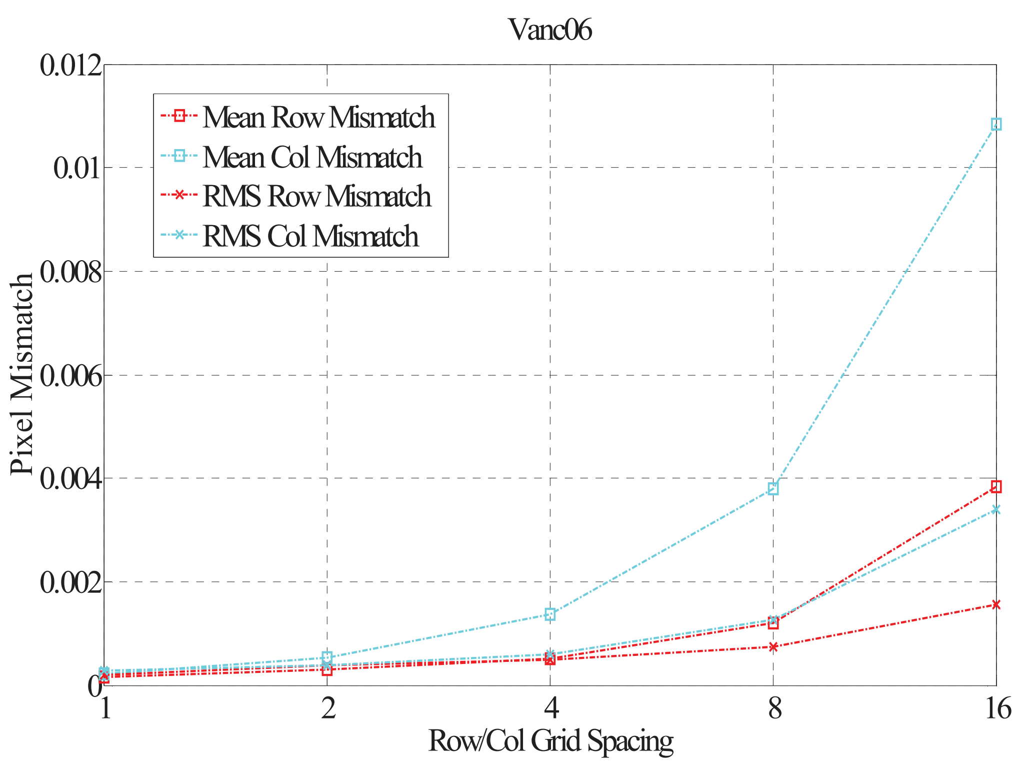

Fig. 24 Accuracy test (Row/Col) for Vanc06

6 Appendix C. System Specifications

Hardware

HP Blackbird 002 LCi

1x Intel® Core2™ Extreme Quad-Core 3.0GHz QX9650 CPU

2x 4GB RAM

1x 7200 RPM Sata HDD

2x 10,000 RPM SATA HDD

2x NVIDIA GeForce GTX-280 GPU with 1GB of GDDR3 SDRAM

Development Enviroment

openSuSE Linux 10.3 (64-bit)

Intel C++ compiler (version 10.1)

OpenMP 2.5 libraries for multithreading

NVIDIA CUDA SDK release 2.0 for programming the GPUs

GeoImaging Accelerator Ortho Performance Test Results
A PCI Geomatics Whitepaper

Authors: Lutes et al.

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