Utilizing commercial graphics processors in the real-time geo-registration of streaming high-resolution imagery

Laurence Flath, Michael Kartz, Randall Frank
Lawrence Livermore National Laboratory

GP² Workshop
August 7-8, 2004

UCRL-PRES-205737
Outline

• Introduction
• Problem
  – Real-time video streaming
• Solution
  – Real-time geo-registration using GPU
• Implementation on graphics processor
• Experimental results
• Demonstration
• Conclusion
Introduction

- Real-time video processing is a trade-off
  - Image size [pixels / frame]
  - Image rate [frames / second]

- General-purpose central processing units (CPUs) are very fast, but total system throughput often is not fast enough
  + Superscalar architecture, vector units
  - Inadequate memory bandwidth

- Graphics Processing Units (GPUs) are not as fast, but are specialized for the task
  + Huge memory bandwidth
  - Not ideal for generalized algorithms
Graphics Processing Work at LLNL

• GPU Discovery Project
  – Develop infrastructure for streaming data processing
  – Research novel data mapping for non-traditional data types
  – Implement algorithms on graphics processors for real-world computational problems
  – Catalog advantages / disadvantages of GPU-based algorithms
  – Investigate mapping of algorithms to higher-level APIs
  – Study impact of GPU implementations on next-generation systems

• Visualization

• Numerous application-based projects for which the GPU is an integral system component
Problem - Viewing Remote Video

Mobile Camera

Ground Link

Scene
Video Transmission Limited by Bandwidth

• High-resolution video generates a huge amount of data
  – $10 \to 100 \text{ MPixels/sec}$
  – $10 \to 400 \text{ MB/sec}$

• Mobile platforms often don’t have access to a high-bandwidth transport medium
  – Microwave, laser comm systems designed for stationary operation
  – Available mobile systems range from 9.6 Kbs to 30 Mb/sec

• Need a $10^4 - 10^5$ compression ratio
  – Generic spatial-temporal algorithms give (at best) 100:1
How To Get $10^4$ Compression Ratio

- Re-scope the problem!
- Ground imagery doesn’t change very often
  - Point camera for a long time at the same spot
  - Transform imagery such that background doesn’t change, even though camera platform is moving
- Only transmit required scene information
  - Moving objects
  - Stationary background (occasionally)
Geo-Registration

- Transform imagery recorded in one perspective into another
- Usually produce ‘nadir-looking’ view
  - Stationary scene ideal for background removal algorithms
  - Result may be used as a map, with features given in GPS coordinates
  - Permits sensor fusion; e.g. visible, infrared, radar, etc.
- Requires inertial navigation system (INS) data from sensor platform
  - Global positioning system (GPS) provides position
  - Inertial measurement unit (IMU) provides attitude; e.g. roll, pitch, heading
Imaging the ground obliquely - y-axis

Treat system like a pinhole camera; i.e. \( Z, G \gg f \)

\[
y_g = \frac{\left( \frac{D^2}{Z} \right)}{\left( \frac{G}{Z} \right) \left( \frac{y_c}{f} \right)} - 1 \left( \frac{y_c}{f} \right)
\]
Imaging the ground obliquely - x-axis

\[
x_g = \frac{D}{G} \left( \frac{y_c}{Z} \right) - 1 \left( \frac{x_c}{f} \right)
\]

"Single-point perspective"
Geo-Registration Algorithm

• Calculate angles and distances from INS data
• Map perspective equations to a homogeneous coordinate transformation
• Propagate source pixels through transformation to output image plane
  – Interpolate / anti-alias
  – Fill-in blank patches
• Remove jitter due to GPS uncertainty, IMU drift
  – Register to known target or previous imagery
  – Shift result to sub-pixel resolution
Muir Flight Experiment

Demonstrate real-time on-board geo-registration and moving target extraction

Helicopter Platform
Wescam 3-Axis Stabilized Gimbal

Commodity PCs
(Twin Apple G5s, ATI Radeon 9600)

MRC RF Comm System (TCP/IP)

Twin 11MPixel, 2Hz Prototype Cameras

GPS/IMU System
System Data Flow - Per Camera

- Six objects (threads) distribute work load between two CPUs:
  - Camera / frame grabber
  - Camera record
  - Geo-registration
  - Motion-detect / blob find
  - Blob record
  - Decimated imagery

- Socket communications
  - Individual control of objects
  - Live data for ground station
Image Processing Steps

Raw Data Collection (Frame Grabber)

Geometry Correction (GPS/IMU)

Scene-Based Registration (Spatial Correlator)

Motion-Detect (Star-Killer)

Blob-Detect (Threshold / Blob-Find)

Data Archival (Transmit and/or Store)

INS Data

Registration From Adjacent FPAs

Per-Sensor Data Pipeline
Image Processing Steps

Raw Data Collection (Frame Grabber)

Geometry Correction (GPS/IMU)

Scene-Based Registration (Spatial Correlator)

Motion-Detect (Star-Killer)

Blob-Detect (Threshold / Blob-Find)

Data Transmit / Archive (Transmit and/or Store)

INS Data

Registration From Adjacent FPAs

GPU-based

CPU-based

Geometry Correction (Image Flip, Distortion)
GPU Processing

- Load image into texture(s)
  - 4K x 2.6K x U16 pixel image
  - Four 2K x 2K tiles (GL_MAX_TEXTURE_SIZE == 2048)
- Calculate transformation matrix based on INS data
- Render registration region
  - Read pixels
  - Perform correlation on CPU
  - Feedback shifts to transformation matrix
- Render entire output image
  - Currently rendering to glX context (not a pbuffer)
  - If desired output greater than 1K x 1K, need to tile output as well
  - Use asynchronous transfer and shared caching modes to reduce readback time
Muir Experimental Results

- Demonstrated on-board, real-time image processing of data
  - Geo-rectification
  - Imagery stabilized by auto-correlation
  - Motion detection and object tracking
- 15 hours of broad area geo-locked imagery
- Stored 5 TB of raw imagery
  - Raw data enable us to repeat the flight in the lab and validate our capabilities
  - Full image collection is not necessary for operation
  - Ground station displays geo-registered imagery and object info
  - Image updates every 20 sec (depending on link bandwidth).
  - Blob information updates at camera frame rate
Conclusion

• LLNL has several efforts aimed at incorporating graphics processors into real-world applications

• GPU-based geo-registration algorithm was demonstrated successfully in the field
  – Real-time transformation of 44 Mpixels/sec, including jitter-removal, motion detection, and blob-finding using commodity hardware
  – Permitted real-time transmission of high-resolution ground imagery
Future Work

• Ongoing efforts for remote video application
  – Map motion detection algorithms to GPU, if found to be practical
    • Fragment shader
  – Remove optical distortions via image warping
    • Vertex shader
  – Add support for digital elevation map (DEM)
    • Wrap texture to 3-D surface
  – PCI Express to reduce readback overhead
Acknowledgements

• NAI
  – Charles Bennett
  – Michael Carter
  – John Marion
  – Robert Sawvel
  – Sheila Vaidya

• Computations
  – David Bremer
  – John Johnson
  – Holger Jones
  – Jeremy Meredith
## CPU Comparison

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Intel Pentium 4</th>
<th>IBM PowerPC 970</th>
<th>ATI Radeon 9700</th>
<th>Sony Emotion Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual address range</td>
<td>32 bits</td>
<td>64 bits</td>
<td>32/128 bits</td>
<td>64 bits</td>
</tr>
<tr>
<td>Real address range</td>
<td>32 bits</td>
<td>42 bits</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Scalar datapath width</td>
<td>32 bits</td>
<td>64 bits</td>
<td>32 bits</td>
<td></td>
</tr>
<tr>
<td>CPU cores per die</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Superscalar execution</td>
<td>4</td>
<td>4 + 1 branch</td>
<td>4 Vertex + 8 pixel</td>
<td></td>
</tr>
<tr>
<td>Pipeline depth (int)</td>
<td>20 stages</td>
<td>16 stages</td>
<td>5 (Vertex)</td>
<td></td>
</tr>
<tr>
<td>Vector extensions</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FPUs</td>
<td>1 + SSE2</td>
<td>2 + AltiVec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1 cache I/D (ways)</td>
<td>12K/8K</td>
<td>64K/32K (DM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2 cache</td>
<td>512K</td>
<td>512K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core frequency (max)</td>
<td>3.0Ghz</td>
<td>1.8Ghz</td>
<td>325MHz</td>
<td>300 MHz</td>
</tr>
<tr>
<td>FSB frequency</td>
<td>200MHz</td>
<td>450MHz</td>
<td>310MHz</td>
<td>150 MHz</td>
</tr>
<tr>
<td>FSB effective bit rate</td>
<td>800MHz</td>
<td>900MHz</td>
<td>620MHz</td>
<td></td>
</tr>
<tr>
<td>FSB width</td>
<td>64 bits</td>
<td>2 x 32 bits</td>
<td>1 x 32 bits (AGP 8x), 4 x 64 bits (DDR)</td>
<td>2 x 16 bit</td>
</tr>
<tr>
<td>FSB data bandwidth</td>
<td>3.2GB/s</td>
<td>2 x 3.2GB/s</td>
<td>2 GB/s (AGP 8x), 19.8GB/s (DDR)</td>
<td>3.2 GB/s (RDRAM)</td>
</tr>
<tr>
<td>Transistors</td>
<td>54 million</td>
<td>52 million</td>
<td>107 million</td>
<td>10.5 million</td>
</tr>
<tr>
<td>IC process</td>
<td>0.13μm</td>
<td>0.13μm Cu + SOI</td>
<td>0.15μm</td>
<td>0.25μm</td>
</tr>
<tr>
<td>Die size</td>
<td>131mm²</td>
<td>118mm²</td>
<td></td>
<td>240mm²</td>
</tr>
<tr>
<td>Voltage (core)</td>
<td>1.5V</td>
<td>1.3V (1.8GHz)</td>
<td></td>
<td>1.8V</td>
</tr>
<tr>
<td>Power (typical)</td>
<td>82W</td>
<td>42W @ 1.8GHz</td>
<td>~50W</td>
<td>13W</td>
</tr>
<tr>
<td>Production</td>
<td>Shipping</td>
<td></td>
<td></td>
<td>Shipping</td>
</tr>
<tr>
<td>Floating-point performance</td>
<td></td>
<td></td>
<td></td>
<td>6.2 GFLOPS</td>
</tr>
</tbody>
</table>
Homogeneous Coordinates

- Homogeneous coordinates are scale invariant:

\[
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}
= \begin{bmatrix}
  ax \\
  ay \\
  az \\
  aw
\end{bmatrix}
\]

- Represent scaled coordinates in 3-space:

\[
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}
\overset{\text{Homogeneous}}{\leftrightarrow}
\begin{bmatrix}
  x/w \\
  y/w \\
  z/w
\end{bmatrix}
\text{  3-D World}
\]
• A homogeneous affine transform:

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix} =
\begin{bmatrix}
  m_0 & m_4 & m_8 & m_{12} \\
  m_1 & m_5 & m_9 & m_{13} \\
  m_2 & m_6 & m_{10} & m_{14} \\
  m_3 & m_7 & m_{11} & m_{15}
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}
\]

• Basic means of all transform operators in higher-level 3-D languages
  – OpenGL
  – DirectX
Scheimpflug Checkerboard

- Virtual overlap of focal plane arrays permits effective pixel counts much larger than commercially available
  - Multiple apertures (maximum required is four)
  - Use inexpensive COTS FPAs

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```