Prototype and Metrics for Data Processing Chain Components of IPM

Vuong Ly

HyspIRI Symposium
Intelligent Payload Module Session
June 5, 2014
Representative IPM Data Processing Chain

Test Data Source

AMS on Citation Forest Service ~Data TBS

GliHT on UC-12 Langley ~350 Mbps

ChaiV640 on UC-12 Langley ~350 Mbps

ChaiV640 on Bussmann Helicopter ~350 Mbps

EO-1 Hyperion Simulated data rate

Ingest/Level 0

Level 1R

FLAASH AC

Level 1G

WCPS

Level 2 SAM

Level 2 Vectorizer

Radiometric Correction

Geometric Correction

Atmospheric Correction

Classifiers & Other Algorithms
# Platforms and Algorithms

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Ground</th>
<th>Level 0</th>
<th>Level 1R</th>
<th>FLAASH Atm Corr</th>
<th>Level 1G Geocorr</th>
<th>GCAP Geocorr</th>
<th>Coreg Geocorr</th>
<th>WCPS SAM</th>
<th>WCPS Potrace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maestro Multicore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tilera 64 Multicore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tilera Pro Multicore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TileGX Multicore</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>SpaceCube 1.5</td>
<td></td>
<td>😊</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Csp ARM/FPGA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 😊: Tested
- 😊: In process
## Platforms and Algorithms

<table>
<thead>
<tr>
<th>Platforms Airborne</th>
<th>Level 0</th>
<th>Level 1R</th>
<th>FLAASH Atm Corr</th>
<th>Level 1G Geocorr</th>
<th>GCAP Geocorr</th>
<th>Coreg Geocorr</th>
<th>WCPS Hot Pixel</th>
<th>WCPS Potrace</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMS/IPM/Citation</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
</tr>
<tr>
<td>Platforms Airborne</td>
<td>Level 0</td>
<td>Level 1R</td>
<td>FLAASH Atm Corr</td>
<td>Level 1G Geocorr</td>
<td>GCAP Geocorr</td>
<td>Coreg Geocorr</td>
<td>WCPS SAM</td>
<td>WCPS Potrace</td>
</tr>
<tr>
<td>GliHT/IPM Cessna</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
</tr>
<tr>
<td>ChaiV640/IPM/Heli</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
</tr>
<tr>
<td>ChaiV640/IPM/B200</td>
<td></td>
<td></td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
</tr>
<tr>
<td>Platforms Airborne</td>
<td>Frame Grabber</td>
<td>Jellyfish Det</td>
<td>Autonom Navigation</td>
<td>🎨🎨</td>
<td>🎨🎨</td>
<td>🎨🎨</td>
<td>🎨🎨</td>
<td>🎨🎨</td>
</tr>
<tr>
<td>Camera/MiniIPM/UAV Heli</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td>😊😊</td>
</tr>
</tbody>
</table>

😊 Tested  
😊 In process
Overview of Effort

- Year 1 Build hardware, begin writing software and arrange flights
- Year 2 Run simulated science scenarios (e.g. instrument calibrations)
- Year 2 Begin Flight tests
- Year 2 Investigate software benchmarks
- Year 3 More flight tests and benchmarks based on results of year 2
- Year 3 Recommendations for future missions
Key Methods to Accelerate Onboard Computing for a Space Environment

- Intelligent onboard data reduction
- Parallel processing, multicore processors
- Use of FPGA as co-processor to accelerate portion of algorithms
Example 1 of Intelligent Onboard Data Reduction: Autonomous Modular Sensor Onboard Processing
Existing Autonomous Modular Sensor (AMS) Pre-processing/Product Relationships

DC Radiance → Reflectance → Temp. → Hot Spots. → Multi-Class Shapefile

FRP

Raw data
Intermediate
Derived Output
“Real” Output

Scaled Visual Product.
Burn Area Emergency Rehabilitation Imagery

BAER Bands:
10 – 7 - 9
Linear Stretched 2%
CCRS (Hot Pixels Algorithm)
Hot Pixels as Topojson on Github

Vectorized Hot Pixs to Topojson format (50% simplification)
File Size: 6KB (2KB .tgz)

Topojson converted to C++ from javascript

Potrace integrated into WCPS

Displayed on MapBox TopoMap

Can be shared on Facebook/Twitter…
All Open Source

http://geojson.io/#id=gist:cappelaere/770dc8388c021ca6091b&map=14/33.3553/-116.5015
Metrics

- Original AMS files size 3.1 Mbytes
- Potrace – converts to raster to vector with the output being GeoJSON
  - Geographic Javascript Object Notation (GeoJSON) file size is 31 kbytes
- Topographic Javascript Object Notation (TopoJSON) converts GeoJSON to TopoJSON format
  - file size is 6 kbytes (choose 50% simplification of vectors)
  - User selectable to about 90%
- Compress TopoJSON using ZIP
  - Compressed size is 2 kbyte
- Compression of 1000:1
- Download 2 kbyte GitHub then can visualize on built in map visualized (Mapbox)
  - OpenStreetMap compatible
  - Viewable in browser
  - Shareable on Facebook and other social media
  - Github is used for versioning on maps and thus will store user annotation to map
Example 2 for Intelligent Onboard Data Reduction: Running Coregistration with Chips
Global Land Survey Maps

- A collection of Landsat-type satellite images from USGS
  - Near complete global coverage
  - Orthorectified
  - Each image has cloud cover of less than 10%
- Ground truth for the registration programs was drawn from the GLS 2000 and can be updated when the GLS 2010 is completed
Currently “chip database” created (in a brute-force fashion) by extracting successive 256x256 sub-images of all GLS scenes and storing them according to path and row.
Automatic Registration of EO1 Scenes Using Global Land Survey (GLS) Database

1. Find Chips that correspond to the Incoming Scene
2. For Each Chip, Extract Window from input scene using UTM coordinates
3. Eliminate Windows with insufficient information
4. Smooth and Normalize gray values of both Chip and Window using a Median Filter
5. Register each (Chip,Window) Pair using a wavelet-based automatic registration: get a local rigid transformation for each pair
6. Eliminate Outliers
7. Compute Global Rigid Transformation as the median transformation of all local ones
8. Compute Correct UTM of 4 Scene Corners of input scene
9. If desired, Resample the input scene according to the global transformation
Scene 1 Before Automatic Registration
Superimposed onto Google Earth
Scene 1 After Automatic Registration
Superimposed onto Google Earth
Scene 2 Before Automatic Registration
Superimposed onto Google Earth
Scene 2 After Automatic Registration
Superimposed onto Google Earth
Conclusions and Future Work

- Results visually acceptable
- Computations very fast and real-time
- RMS still too high (Translation errors between 0.4 and 2.5 pixels) because:
  1. Chips and windows need to be pre-selected based on the information content (e.g., using an entropy measure)
     - Registration would be more accurate because transformation would only be computed on pairs that have a significant amount of features
     - Registration would be faster because less local registrations
     - Chip database would be smaller to be stored onboard
  2. Global transformation should be computed by taking the list of original corners coordinates of each window and their corresponding corrected coordinates, and treat them as a list of ground control points and their corresponding points => after outlier elimination, global transformation can be computed using a rigid, an affine or a polynomial transformation.
  3. Masks for clouds and water should be included, so registration would not use cloud or water features that are often unreliable
- Onboard, computations can be performed on SpaceCube or hybrid processor
Representative IPM Data Processing
Chain & Metrics
Building to Helicopter Experiment
## Hyperspectral Image Processing

<table>
<thead>
<tr>
<th></th>
<th>Radiometric Correction (CHAI data)</th>
<th>*Atmospheric Correction (FLAASH) (EO1 Hyperion data)</th>
<th>Geometric Correction (GCAP) (GLiHT data)</th>
<th>*WCPS (vis_composite) (EO1 Hyperion data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>864 MHz TILEPro64 (1 core)</td>
<td>121.95</td>
<td>2477.74</td>
<td>183.42</td>
<td>72.39</td>
</tr>
<tr>
<td>864 MHz TILEPro64 (49 cores)</td>
<td>23.83</td>
<td>1744.13</td>
<td>4.59</td>
<td>21.63</td>
</tr>
<tr>
<td>1.0 GHz TILE-Gx36 (1 core)</td>
<td>57.22</td>
<td>897.71</td>
<td>28.51</td>
<td>19.93</td>
</tr>
<tr>
<td>1.0GHz TILE-Gx36 (36 cores)</td>
<td>9.21</td>
<td>588.71</td>
<td>1.41</td>
<td>8.72</td>
</tr>
<tr>
<td>2.2GHz Intel Core i7</td>
<td>2.09</td>
<td>58.29</td>
<td>0.169</td>
<td>2.26</td>
</tr>
<tr>
<td>Virtex 5 FPGA Image data:</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
</tbody>
</table>

**Notes:**
- Unit is in seconds
- TILEPro64 – No floating point support
- TILEGx36 – Partial floating point support
- * Indicates time includes file I/O

**Image data:**
- GLiHT: 1004 x 1028 x 402 (829,818,048 bytes)
- Hyperion (EO1H1740732001151111K3): 256 x 6702 x 242 (830,404,608 bytes)
- Chai640: 696 x 2103 x 283 (828,447,408 bytes)
# FLAASH Parallelization Effort

<table>
<thead>
<tr>
<th>Task</th>
<th>Walltime (s)</th>
<th>System (s)</th>
<th>User (s)</th>
<th>Reflect::</th>
<th>RadtoRef</th>
<th>YES</th>
<th>Parallelized?</th>
<th>Original Walltime (s)</th>
<th>Parallel Speedup</th>
<th>Total Time (h:m:s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surf reflectance</td>
<td>32.7952</td>
<td>5.1398</td>
<td>27.6554</td>
<td>YES</td>
<td></td>
<td></td>
<td></td>
<td>197.317</td>
<td>6.016642679</td>
<td>0:10:46</td>
</tr>
<tr>
<td>Cube smoothing</td>
<td>145.741</td>
<td>18.1373</td>
<td>127.6037</td>
<td>mini_cube- &gt;Smooth; FFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cube reduction</td>
<td>112.363</td>
<td>13.9834</td>
<td>98.3796</td>
<td>mini_cube- &gt;Condense</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cube load &amp; distrib</td>
<td>89.0128</td>
<td>11.0775</td>
<td>77.9353</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cube gather &amp; write</td>
<td>83.2988</td>
<td>10.3664</td>
<td>72.9324</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aerosol Retrieval</td>
<td>52.6236</td>
<td>6.54892</td>
<td>46.07468</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Water col retrieval</td>
<td>34.7984</td>
<td>4.3306</td>
<td>30.4678</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectral Polishing</td>
<td>33.5795</td>
<td>4.17891</td>
<td>29.40059</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor calibration</td>
<td>28.5097</td>
<td>3.54799</td>
<td>24.96171</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Images and Masks</td>
<td>17.1781</td>
<td>2.13779</td>
<td>15.04031</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectral Resampling</td>
<td>8.72743</td>
<td>1.08611</td>
<td>7.64132</td>
<td>Smile_Resampler::Cube_Copy</td>
<td>YES</td>
<td></td>
<td></td>
<td>241.669</td>
<td>27.69074057</td>
<td></td>
</tr>
<tr>
<td>Cloud Masking</td>
<td>5.93287</td>
<td>0.738336</td>
<td>5.194534</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor slit function</td>
<td>0.764612</td>
<td>0.0951547</td>
<td>0.6694573</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modtran Tables</td>
<td>0.416416</td>
<td>0.0518223</td>
<td>0.3645937</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>un-categorized</td>
<td>0.0397966</td>
<td>0.00495262</td>
<td>0.03484398</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flaash setup</td>
<td>0.000163794</td>
<td>2.04E-05</td>
<td>1.43E-04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total time</td>
<td>645.7813884</td>
<td>81.425006</td>
<td>564.3563824</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1:641422344</td>
</tr>
<tr>
<td>Total time (h:m:s)</td>
<td>0:10:46</td>
<td>0:01:21</td>
<td>0:09:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Original Wall time: 1060.0:17:40
Time spent in FLAASH components

- Surf reflectance: 23%
- Cube smoothing: 17%
- Cube reduction: 14%
- Cube load & distrib: 13%
- Cube gather & write: 8%
- Aerosol Retrieval: 5%
- Water col retrieval: 4%
- Spectral Polishing: 5%
- Sensor calibration: 5%
- Images and Masks: 5%
- Spectral Resampling: 14%
- Cloud Masking: 13%
- Sensor slit function: 5%
- Modtran Tables: 5%
- un-categorized: 4%
- Flaash setup: 0%

Total: 100%
Conclusion

- Examining a variety of methods to speed up onboard processing chain to meet needs of low latency users
- Dovetailing efforts and metrics with High Performance Space Computing (HPSC) effort sponsored by NASA Office Chief Technologist
- IPM data processing effort applied to multiple future mission needs