IMAGE NAVIGATION AND REGISTRATION PERFORMANCE ASSESSMENT EVALUATION TOOLS FOR GOES-R ABI AND GLM

Scott Houchin, Brian Porter, Justin Graybill, Philip Slingerland

The Aerospace Corporation, 14301 Sullyfield Circle, Unit C, Chantilly, VA 20151-1622

ABSTRACT

The GOES-R Flight Project has developed an Image Navigation and Registration (INR) Performance Assessment Tool Set (IPATS) for measuring Advanced Baseline Imager (ABI) and Geostationary Lightning Mapper (GLM) INR performance metrics in the post-launch period for performance evaluation and long term monitoring. IPATS utilizes a modular algorithmic design to allow user selection of data processing sequences optimized for generation of each INR metric. This novel modular approach minimizes duplication of common processing elements, thereby maximizing code efficiency and speed. Fast processing is essential given the large number of sub-image registrations required to generate INR metrics for the many images produced over a 24 hour evaluation period. This paper describes the software design and implementation of IPATS and provides preliminary test results.

Index Terms—Image registration, Image Navigation, Software Design, Automation

1. INTRODUCTION

The Image Navigation and Registration (INR) Performance Assessment Tool Set (IPATS) was developed to measure INR performance metrics of the Advanced Baseline Imager (ABI) and Geostationary Lightning Mapper (GLM), onboard GOES-R, in both the post-launch period for performance evaluation and for long term monitoring. IPATS utilizes a modular algorithmic design to allow user selection of data processing sequences optimized for generation of each INR metric.

The evaluation tool component of the entire IPATS toolset consists of two main software applications: The Image Pair Selector and Evaluator (IPSE) and the Output Data Analysis Tool (ODAT).

Given one or more ABI or GLM Background Images to be evaluated, IPSE identifies other images against which each newly received image should be evaluated. Those other images can come from within that input image set, from within a database of previously received ABI or GLM Background images, or from within a database of truth images derived from Landsat data. It then identifies appropriate chips within the overlapping geographic region of that image pair to evaluate. The navigation and registration results for each individual evaluation region for each image pair, in terms of correlation error and measurement uncertainty, are then stored in either a SQLite3[3] or PostgreSQL[4] database for further analysis. The prime focus of IPSE is performing the core analysis on each individual image pair at sufficient speed to keep up with data collection (process a day's data within a day) using limited computing hardware. IPSE is a C++ command line application with both standalone and client/server components. To perform some of the image processing, IPSE uses the open source OpenCV toolkit[2] to manage image data and perform many but not all image processing operations.

ODAT, an analyst focused, graphical user interface-based Python application[5], allows each analyst to query the database generated by IPSE for specific portions of the analyzed imagery (e.g., navigation error for all Band 2 images collected on a specific day, or all frame-to-frame registration error output for all Band 3 images collected over the previous week). ODAT then allows the user to export raw analysis results, generate additional statistics across multiple analysis results, generate plots of the results, and rerun specific evaluates through IPSE using debug modes or using alternate evaluation parameters.

2. THE IMAGE PAIR SELECTOR AND EVALUATOR OVERVIEW

The Image Pair Selector and Evaluator (IPSE) performs the bulk of the scientific analysis in the IPATS toolset. Within IPSE, data analysis is divided into three main components. Image pair identification, evaluation location identification, and then finally the actual scientific evaluation.

2.1. Image pair identification

First, for each specific analysis type (e.g., navigation, band-to-band registration), IPSE examines metadata for the input imagery and identifies which images are to be compared against which other images. The rules for image pair identification vary from evaluation to evaluation.

Navigation evaluation is performed on every single input ABI and GLM Background image received. For the ABI images, the input image is compared against a set of truth chips, derived from Landsat data, and the image pair

identification step is skipped. For GLM Background image navigation, each GLM image is compared with the single previous and single next collected ABI image from a single configurable band.

For band-to-band registration (BBR), IPSE identifies all of the available images from a single collection, grouping images into collections by the start time in the file name. The specific pairs of bands that are to be compared is a configurable parameter.

Frame-to-frame registration (FFR) pair identification is performed by finding the previously collected matching ABI image (same band, same satellite position, same type, etc.).

Swath-to-swath registration (SSR) pair identification is the most complicated, because from an official requirements perspective, swath-to-swath registration evaluation is performed on specifically tasked collection. In order to provide automation for SSR, IPSE looks for two Mesoscale images from the same band and same satellite position, and that were collected approximately 30 seconds apart. IPSE then loads the geographic metadata from the image files in order to determine the covered ground area in GOES-R fixed grid angular coordinates. If the images overlap by approximately one swath, then SSR evaluation is performed on that image pair.

2.2. Evaluation location identification

Once an image pair is identified, it then identifies specific chips of the whole image on which to perform the evaluation. Given the computational cost of performing a cross-correlation operation and the likelihood that navigation and registration error varies across the image (particularly for the large full-disk and continental US images), evaluation is performed on a sequence of small chips extracted from the images.

For ABI NAV, the database of Landsat truth images is considered to be the locations on which the ABI image should be evaluated. IPSE calculates the location and size of the portion of the ABI image (in pixels) covered by the truth image, taking into account any pixels needed in either image to minimize edge effects and for padding. IPSE then treats the truth image as if it was just another ABI image for the purpose of the scientific evaluation, thus allowing all evaluation types to be treated as the comparison of two images at one or more locations. The list of Landsat chips is provided to IPSE in the form of an SQLite3 database, or as a comma separated value (CSV) file, which IPSE then imports into an in-memory SQLite database at program startup.

For all other evaluation types, including GLM Background image navigation, IPSE determines the area on the ground covered by both images in the pair. It then searches a database of predetermined locations that are sufficiently inside the intersecting area and are flagged for the specific evaluation type.

ABI L1b imagery is resampled to an angular fixed-grid coordinate system; given a virtualized satellite position (e.g., over the equator at 89.5° W longitude, neighboring pixels in either the X or Y direction have a fixed angular separation. This is unfortunately not true for the GLM background image, which is not resampled to a fixed grid; individual X and Y coordinates are provided for every single pixel. To minimize overhead, since image data is not loaded from any image at this phase of processing, IPSE treats the GLM background image as if it is an ABI full-disk image from the perspective of ground coverage.

2.3. Scientific evaluation

The vast majority of the real work performed by IPSE is the actual evaluation of an image pair at a single location. IPSE is structured such that this work is performed by a single function, regardless of evaluation type. This function does assume that the input images are on a fixed grid, but it does not require that the grid be the same for both input images; as long as the resolution ratio is an integer, IPSE can compare the images. This presents a challenge for the GLM Background images, since they are *not* resampled to the fixed grid; that resampling must be done inside IPSE. To compensate, the C++ class model implements the GLM image loader as a subclass of the ABI image loader. From the perspective of the common evaluator function, all image data is loaded on-demand, and only for specific pixel regions as requested, based on the location of and size of the evaluation region. This allows the GLM image loader to perform the necessary resampling to a fixed-grid only for the portions of the image that are covered by the identified evaluation locations, saving significant processing time.

The details of the evaluation algorithm are fully covered in [1]; in summary, IPSE determines the overlapping area in the two images, aligns them on a common fixed grid, and then expands the necessary region of each image a sufficient amount to ensure that when interpolation is performed to generate pixels of both images at a target evaluation resolution, that interpolation is performed using only real pixels. As shown in Figure 1, this processing results in the two images being resampled to a common resolution. Additional resampled padding pixels are extracted from the higher resolution truth image; this allows for the lower resolution image (the one under evaluation) to be moved around inside the larger truth region to find the best correlation.

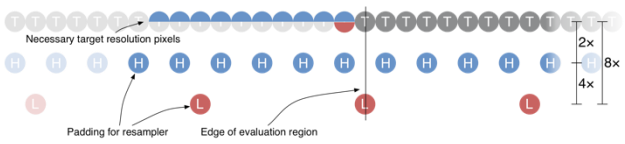

Figure 1. IPSE resampling truth and evaluation to common resolution

Once the two chips are extracted and resampled to a target evaluation resolution, one of three correlators can be used to calculate the error between the two images: A position-based correlator, using the OpenCV matchTemplate function, a correlator using normalized-mutual information, and an FFT-based phase correlator. The initial raw correlation output can then be refined by either centroid or parbolic fit methods. That measured and refined error data is then output into the IPSE evaluation results database.

3. IPSE EVALUATION RESULTS DATABASE STRUCTURE AND CAPABILITIES

In order to minimize size and maximize creation speed of the Image Pair Registration Record (IPRR) database, the database is divided into several tables, linked together through an ID column on each row in each table. This allows information that is common across many rows (from hundreds to millions of rows) to be stored only once in the database, but be correctly linked to the record for each individual location evaluated for a given pair of images.

Figure 2 shows the relationships between the key tables in the IPRR database, using the Unifed Modeling Language (UML). At a high level, a diamond-tipped line shows that a row in the table at the diamond-end of the line has a reference to a row in the table at the plain end of the line. The number on the line indicates the multiplicity of that relationship. For example, a Corr row points to two Chips, and can also point to 0 or 1 Errors.

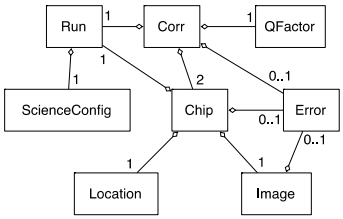

Figure 2. UML class diagram showing the key tables in the IPRR database

Key tables in the database are:

- Rows in the Corr table each contain a single correlation output in terms of both raw and refined registration error, for a single location within a single pair of images, for a single run. This table links back to other tables that specify the configuration parameters, the images under evaluation, and the chips extracted from those images.
- Rows in the ScienceConfig table each contain the specific scientific parameters (e.g., the subpixel factor, interpolation method and correlation method) used for a given set of evaluations. This data is generated indirectly from the command line and configuration parameters specified by the user to ensure that if two users specify the same configuration, either intentionally or coincidentally, the resulting correlation output records all link back to the same configuration. In addition, this table allows for configurations to be named, simplifying the process for an analyst to use a known configuration.
- Rows in the QFactor table specify the quality factors used for the band pair of the images under evaluation to determine whether the images were similar enough to compare (e.g., to exclude a cloud covered image from evaluation against a cloud free image)
- Rows in the Chip table each specify the pixel region extracted from an image under evaluation, as well as the center of the chip in fixed grid angular coordinates.
- Rows in the Image table each specify the filename and key metadata extracted from a single image under evaluation
- Rows in the Location table specify additional information about the ground location of the chip.
- Rows in the Error table specify additional error information, for either a correlation, chip or image. For example, if correlation fails, a chip is too close to the edge of an image, or if an image file is corrupt and cannot be loaded, the error will be recorded.
- Rows in the Run table specify the time of execution and information about the version of IPSE being used.

This structure of the IPRR database allows IPSE to generate the necessary data for large volumes of individual evaluations without inducing bloat on the database. For example, the band-to-band evaluations for one day of ABI imagery could result in millions of individual evaluations.

4. IPSE COMPUTATIONAL PERFORMANCE

In our current processing environment, IPSE can be run using a PostgreSQL database, with the processing spread across 80 cores.

Using a library of simulated 2240 ABI L1b full disk image files (140 sets of 16 bands, each image 10848×10848 pixels), IPSE is able to perform NAV evaluation on all images in about 5 minutes, using 378 Landsat chips to determine the evaluation locations. BBR evaluation on all 140 sets can be completed in about 80 minutes, but we expect planned improvements to drop that time down under 60 minutes. BBR evaluation on that image set results in 6,498,053 attempted evaluations, with 5,445,058 complete IPRR records generated after outlier rejection. We do not have time figures for FFR or SSR at this time, but expect FFR to be on the same order of magnitude as NAV, and for SSR to require minimal resources due to the highly restricted data sets due to manual tasking of SSR pairs.

5. OUTPUT DATA ANALYSIS TOOL

While the IPRR database allows for IPSE to perform large volumes of evaluations in small amounts of time, the IPRR database does not lend itself for easy direct analysis. To aid

the end-user in evaluating the results and performing cross-result comparisons, the IPATS tool set contains the Output Data Analysis Tool (ODAT). ODAT was developed with two goals in mind: to provide a standard interface to the IPATS IPRRs, and to provide an easy-to-use graphical interface for users who simply want to compute a set of stock statistics and plots. ODAT has been developed in Python (version 2.7 or higher). Python is becoming increasingly popular for data analysis due to its powerful language constructs (it is a fully object-oriented language), its increasingly refined data analysis packages, and the fact that it is open source and free [5][6]. In addition, Python is multi-platform, enabling easy deployment to Windows, Linux, and Mac environments[5]. In order to keep the number of external Python packages to a minimum, ODAT uses stock Python 2.7 modules such as Tkinter for the graphical user interface (GUI) and the SQLite3 connector for the database.8 The only required external package is SciPy, and ODAT makes heavy use of the SciPy sub-modules NumPy, Matplotlib, and Pandas for the data manipulation and analysis portions of the code due to the flexibility and performance these packages offer [6][7][8][9][10].

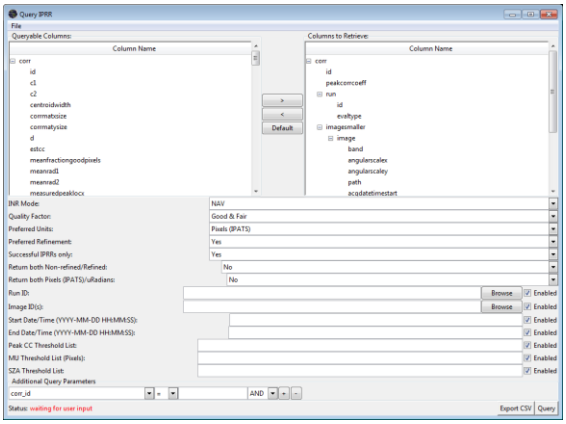

Figure 3. ODAT Query IPRR screen

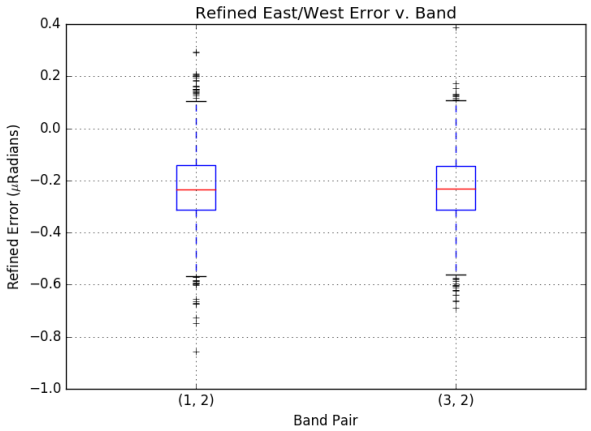

Figure 4. ODAT example refined East/West error vs. band plot

In addition to viewing and analyzing already generated results. ODAT allows the analyst to reprocessing specific images at specific locations in either debug modes or with alternate scientific parameters. In the event of out of family results, the analyst can then perform additional testing without having to manually identify the images and locations, and then manually run IPSE on those locations with altered settings.

6.
7. CONCLUSION

The IPSE and ODAT software tools, part of the IPATS tool set, provide a high performance, automated processing mechanism for evaluating navigation and registration error for GOES-R, along with easy to use tools for the analyst to examine the results and generate additional statistics and metrics. These tools are an essential part of the task to verify registration and navigation performance of the GOES-R instruments.

8. REFERENCES

[1] De Luccia, F., S. Houchin, B. Porter, J. Graybill, E. Haas, P. Johnson, P. Isaacson, A. Reth, Image navigation and registration performance assessment tool set for the GOES-R Advanced Baseline Imager and Geostationary Lightning Mapper, Proc. SPIE 9881, Earth Observing Missions and Sensors: Development, Implementation, and Characterization IV, 988119 (May 2, 2016); doi: 10.1117/12.2229059

[2] Open Source Computer Vision (OpenCV), *About OpenCV*, 02 October 2014, http://opencv.org/ (04 February 2016)

[3] SQLite Consortium, *About SQLite*, SQLite, 2016, https://www.sqlite.org/about.html (02 February 2016)

[4] PostgreSQL, *PostgreSQL: The world's most advanced open source database*, https://www.postgresql.org/about/, (2016)

[5] Python Software Foundation, *Python Language Reference, version 2.7, 2016*, http://www.python.org (02 February 2016).

[6] McKinney, W., *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*, O'Reilly Media, (2012).

[7] Van der Walt, S., Colbert, C., and Varoquaux, G., *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science & Engineering, 13, 22-30 (2011).

[8] Hunter, D. J., *Matplotlib: A 2D Graphics Environment*, Computing in Science & Engineering, 9, 90-95 (2007).

[9] McKinney, W., *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, 51-56 (2010).

[10] Jones E., Oliphant E., Peterson P., et al., *SciPy: Open Source Scientific Tools for Python*, 2001-, http://www.scipy.org/ (02 February 2016).