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

Image Pair Selector & Evaluator (IPSE)

- Part of Image navigation and registration performance assessment tool set (IPATS) [1] for Geostationary Operational Environment Satellite – R Series
- Operational-performance C++ tool performs evaluation of Advanced Baseline Imager (ABI) images against provided Landsat truth data, and between two ABI images
- Takes pile of input ABI and Geostationary Lightning Mapper (GLM) background images
- Identifies pairs of images to be compared, and then the locations within the image pair to perform detailed comparison
- OpenCV [2] based image processing

Single science module can compare

- Any two images (ABI to ABI, ABI to Landsat, ABI to GLM)
- At a given location in FGA coordinates
- For a given evaluation window size in pixels
- Evaluation region centered on the given location
- Supports images with different resolutions

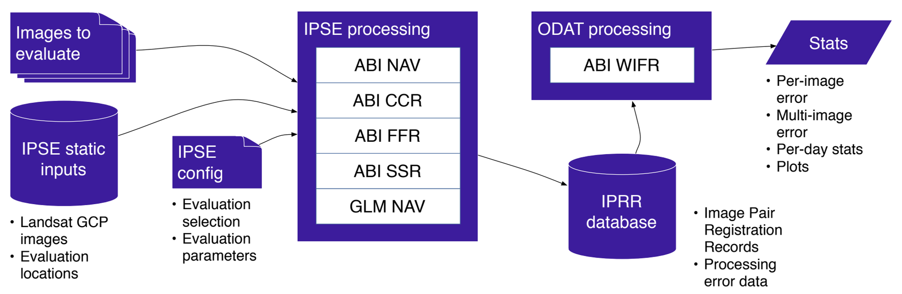
Basis for all test types

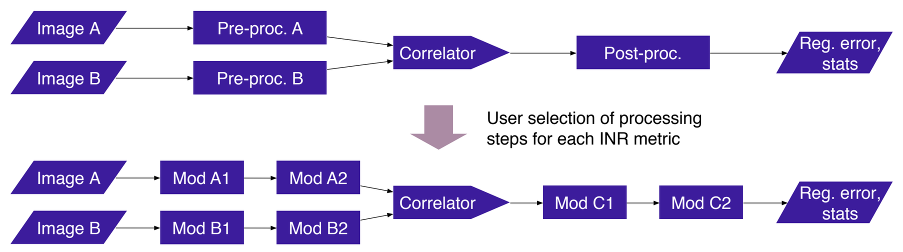
- NAV: Absolute navigation evaluations against Landsat truth data
- BBR: Relative registration evaluation between different bands collected at the same time
- FFR: Relative registration evaluation between consecutive images of the same band/type
- SSR: Relative registration evaluation between the two swaths

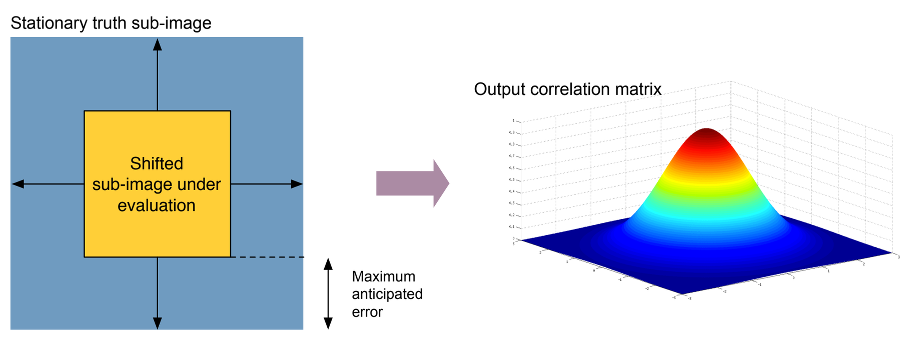
Evaluation type specific code focuses on identifying the evaluations to perform given …

- A list of images to evaluate
- A list of evaluation locations coded for use by ABI resolution and evaluation type
- A list of Landsat chips

Perform as little work as possible
Operationally, IPSE must perform millions of evaluations per day, so processing extra pixels ads up quickly!







IPSE computational performance

- Emphasis was placed on the run-time speed of IPSE
- Must be able to process all operational data at the same rate as collection

Significant parallelization opportunities

- Parallel evaluation locations in a single process
- Parallel evaluation of multiple image pairs in a single process
- Problematic because NetCDF/HDF libraries not entirely thread-safe
- Parallel evaluation using multiple IPSE processes
- Ultimate limitation is the amount of hardware you throw at the problem
- Requires method to command-and-control multiple IPSE processors

- Image files downloaded from customer web server using automated script
- IPSE processing kicked off in batches of 100 image files
- Processing completed within minutes of initiation
- 20 IPSE task queue workers running on one HPC server
- 50-75% load on 96 core Linux server with 1TB RAM

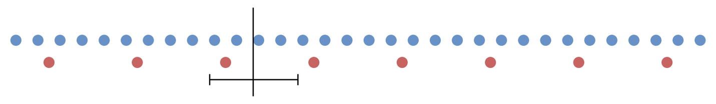
More than exceeds rate of data collection

- **22 TB** of image data over 145 days
- **5,117,361** input ABI images
- **42,383,733** image pairs evaluated (ABI NAV, CCR, FFR)
- **2,945,961,673** individual evaluation window correlation results

Evaluation-type independent pixel processing

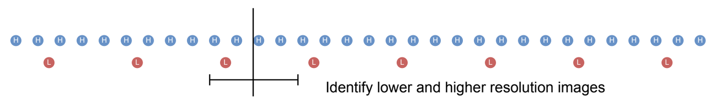
evaluateImagePairAtLocation()

- Compare two images for a given evaluation region size at a location
- Showing 1D illustrations for simplicity
- No assumptions made regarding image resolution; all processing based on metadata from the input file

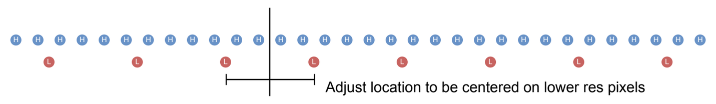


Determine higher resolution image

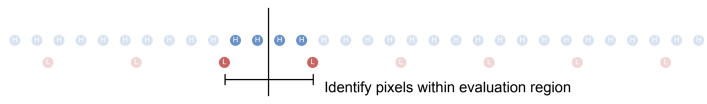
- Image with explicitly smaller pixel size ($14\mu r$ vs. $56\mu r$)
- Image with smaller center wavelength ($0.47\mu m$ vs. $64\mu m$)
- Image collected first
- Image with north-most top



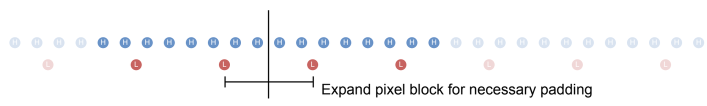
IPSE does not work with fractional pixels, so evaluation region must be perfectly aligned with the centers of pixels in the lower resolution image



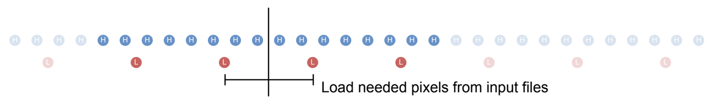
IPSE starts pixel identification by considering only the pixels inside or aligned with the center of the lower resolution pixels at the edges of the evaluation region



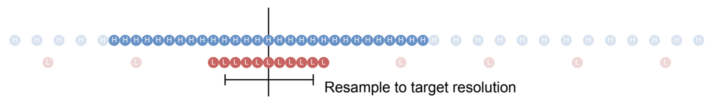
IPSE then expands that evaluation region independently in the lower and higher resolution images to include all necessary pad pixels to avoid resampling artifacts



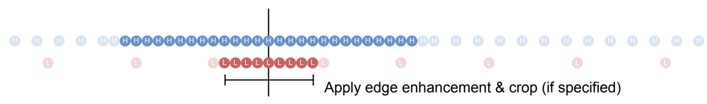
IPSE pulls only minimally necessary pixel data from the input



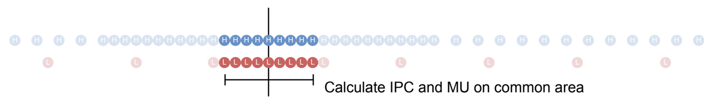
Resample images to target evaluation resolution; pad pixels inherently cropped away as part of resampling operation



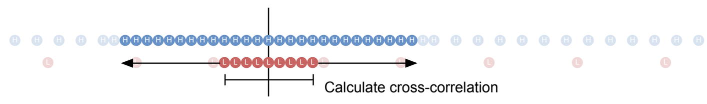
Edge enhancement performed if requested; once filtering kernel is applied, additional pad pixels are cropped away



Cross-correlation is expensive, so first screen the resampled data for similarity and measurement uncertainty; stop processing if images don't sufficiently match

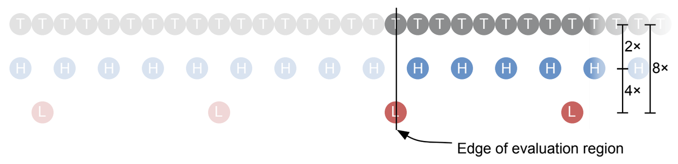


Finally, run the cross-correlator, and then perform peak refinement on the cross-correlation matrix

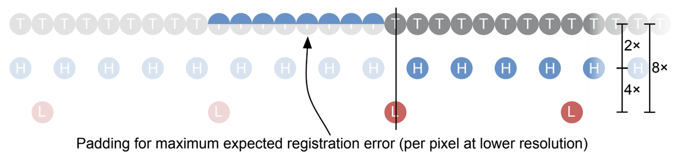


Avoiding edge artifacts by calculating minimal padding

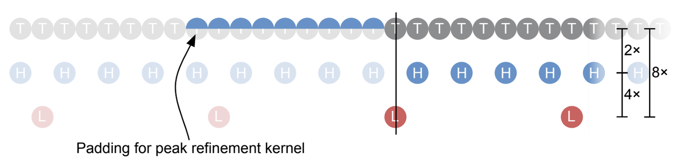
- Padding to determine the evaluation region
- Padding to avoid edge artifacts when resampling and filtering
- Calculations are split between the target and source resolutions



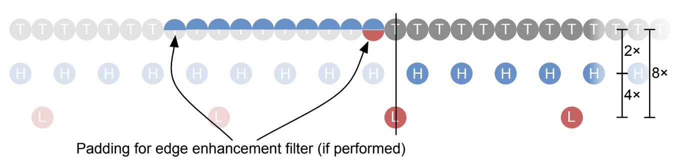
User specifies maximum expected registration error with respect to the lower resolution image; IPSE translates that value to target resolution



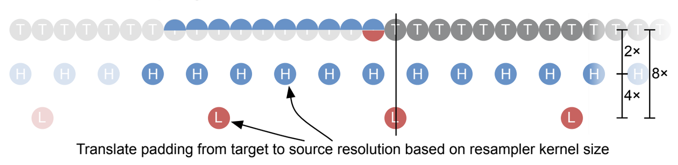
IPSE adds pixels dependent on width of peak refinement kernel to ensure that peak refinement works all the way out to the edge of the user's specified maximum error



If edge enhancement (Sobel or Roberts) is specified, IPSE adds padding to both images; calculations performed at target resolution; this ensures that all pixels for the evaluation window are calculated from real input data, not replicated or reflected pixels

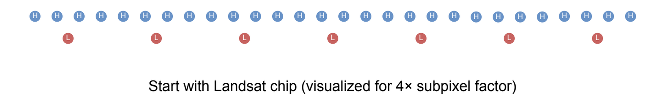


IPSE then translates from target to source resolution; if up-sampling, IPSE considers the width of the resampling kernel; similar consideration when downsampling

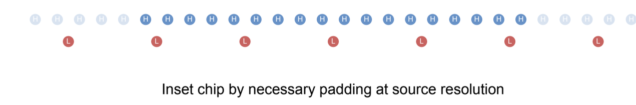


Working backwards to calculate evaluation region and size for NAV

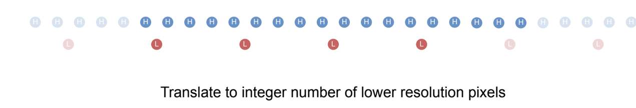
Landsat chips contain only the pixels for the evaluation location; no extra pixels for padding! IPSE must work backwards to calculate the actual evaluatable region, so the common code can then pad it back out without exceeding the size of the chip



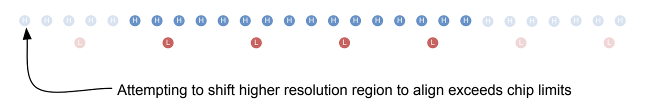
Start with the entire Landsat chip, but inset by the necessary padding for cross-correlation and resampling on each side

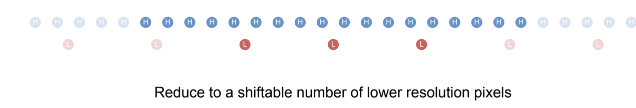


The number of pixels left from the Landsat chip is then translated down to the largest number of ABI pixels that completely fit within that size; we're not selecting ABI pixels yet! Just calculate the size of the evaluation region in terms of the lower resolution ABI image

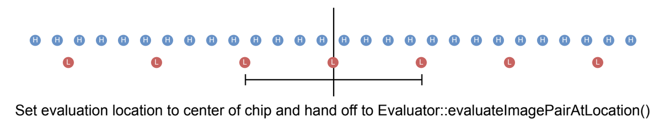


One gotcha! If the number of whole ABI pixels switches between an even and odd count from the original unpadded size, the evaluation size is too big and the necessary padding will overflow the Landsat chip. Shift the Landsat region by 1/2 ABI pixel is greater than the number of unused pixels because of rounding down to an integer number of ABI pixels



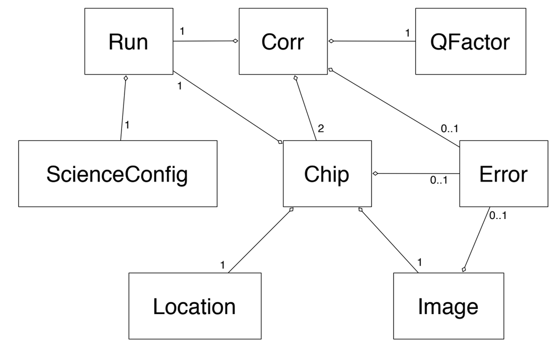


IPSE places the evaluation location at the center of the Landsat chip and then hands the ABI image and Landsat chip off to the common evaluateImagePairAtLocation(); Processed exactly if it were any other image pair and location!



Correlation output stored relational database (SQLite3 or PostgreSQL)

Evaluation configuration, image list and evaluation results are stored in a relational database, using either SQLite3 [3] for smaller data sets, or PostgreSQL [4] for operational data sets. 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.



Corr: a 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.

ScienceConfig: 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

QFactor: 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)

Chip: the pixel region extracted from an image under evaluation, as well as the center of the chip in fixed grid angular coordinates.

Image: the filename and key metadata extracted from a single image under evaluation

Location: additional information about the ground location of the chip.

Error: 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.

Run: the time of execution and information about the version of IPSE being used.

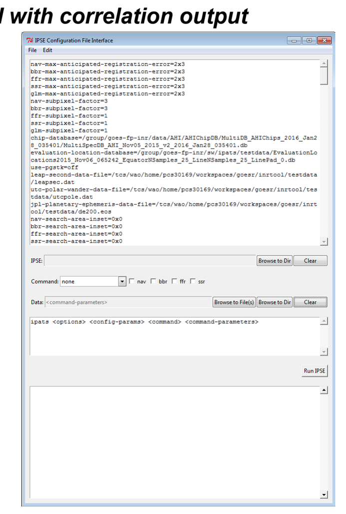
This structure of the IPRR database allows IPSE to generate the necessary data for large volumes of individual evaluations without inducing bloat on the database. For example, the band-to-band evaluations for one day of ABI imagery could result in millions of individual evaluations.

Output Data Analysis Tool (ODAT)

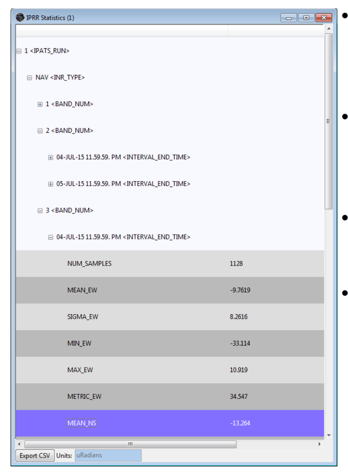
- Allows analyst to query correlation database and perform follow-on analysis
- Reads SQL database and outputs IPSE results
- Easy-to-use interface that allows end-user to export results to CSV, generate statistics and generate plots
- Python 2.7 [5] with NumPy, SciPi, Matplotlib and Pandas [6,7,8,9,10]

View IPSE configuration associated with correlation output

- Addresses need to view configuration file content from specific IPRR record(s)
- Configuration file is now stored in IPRR database
- Allows the configuration file content to be edited and saved as needed
- Additionally allows IPSE to be run from within GUI
- Note: ODAT must be run on Linux server in order to to make additional IPSE runs

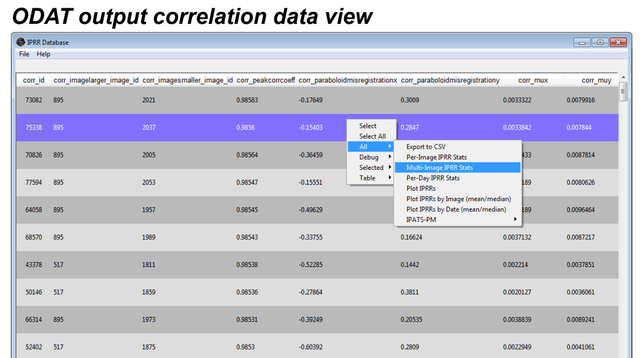


Uses Python Pandas DataFrame objects

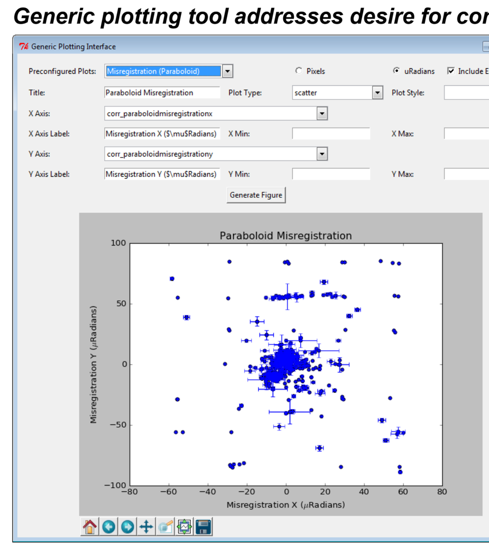


- Pandas group by functionality is used to group data by combinations of the input parameters, such as image date and band
- Stock statistics provided by Pandas are used such as the min, max, mean, median and standard deviation of the registration error
- Custom statistics are provided by extending the Pandas DataFrame object with custom code
- Several outlier rejection methods are implemented by extending the DataFrame objects

ODAT output correlation data view



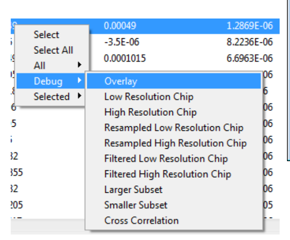
Generic plotting tool addresses desire for configurable plotting

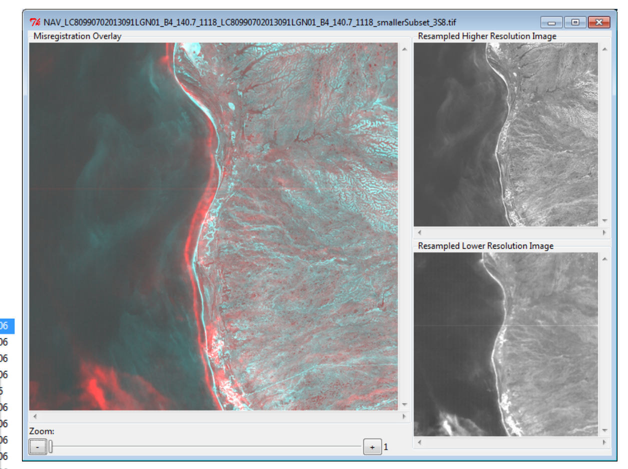


- Any column in the correlation database can be used as either X or Y axis
- Line, scatter, box, histogram, etc, …
- Axis ranges can be modified
- Title, axis labels can be edited
- Conversion between pixel and fixed-grid angle coordinate spaces
- Error bars
- Can save plot image in multiple formats (e.g., tif, png, eps, jpg)
- Provides all plot styles available within local SciPy installation
- Pan, zoom, reset of figure, and cursor position
- Preconfigured plots available through menu lets analyst bring up standard plots quickly
- New plot configurations can be added

Intermediate image viewer
View any of the pre-defined IPATS intermediate images from debug mode

Overlay the higher resolution image on the lower resolution image after being resampled to the same resolution





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).

Contact info
Scott Houchin
Senior Project Leader
The Aerospace Corporation
14301 Sullyfield Circle, Unit C
Chantilly, VA 20151-1622

571-307-3914
scott.houchin@aero.org