

Data Processing and Management of Diwata-1 Imagery Using Free and Open Source Software Stack

Tupas, Mark Edwin A.¹; Tamondong, Ayin M.¹; Magallon, Benjamin Jonah P.²; Aranas, Romer Kristi D.²; Sempio, Julius Noah H.²; Bauzon, Ma. Divina Angela I.²; Jiao, Benjamin Joseph D.²; Yu, Carl Earvin A.²

¹Department of Geodetic Engineering, University of the Philippines, Diliman, Quezon City 1101

matupas@up.edu.ph

²PHL-Microsat Project 3 (Data Processing, Archiving and Distribution), University of the Philippines, Diliman, Quezon City 1101

KEYWORDS: Open Source, FOSS4G, Diwata-1, data processing, data management

ABSTRACT

Diwata-1, the first Philippine microsatellite, has been operating for the past 15 months and is continuously providing imagery for earth observation. Concurrent with this development, the ground control and receiving station capacity is also being laid out. This includes the capability to automate data processing of acquired images for faster turnaround times after image acquisition and reception. In this paper, we discuss the efforts of the PHL-Microsat program to process and manage these images using a FOSS (Free and Open Source Software) stack.

Upon reception, quality assessments and radiometric calibration of the images are performed using custom scripts based on GDAL, Rasterio, and OpenCV. These scripts include (but not limited to) computations for pixel dropout, signal to noise ratio (SNR), and cloud cover assessments. Pointing constraints on the small spacecraft necessitated improvements on the geolocation procedures, which include the development of an automated georeferencing platform using a PostgreSQL/PostGIS-based keypoint database derived using SIFT descriptors from OpenCV. While this is currently in progress, manual operations proceed using QGIS and its GDAL-based Georeferencer plugin. Overarching these are operations and management tasks, for which we developed an indigenous browser based platform utilizing FOSS web technologies such as Django and Leaflet. The system capability includes an orbit tracker, image management dashboards, and data distribution endpoint.

Customizing for fit of use, performance, and overall improvements, these developments have already undergone several iterations and FOSS changes. The lessons learned, and systems and platforms developed have increased our operational capacity in preparation for Diwata-2 and other upcoming Philippine satellites.

BACKGROUND

Diwata-1 is a 50kg earth observation microsatellite jointly developed by Hokkaido University, Tohoku University, University of the Philippines, and Department of Science and Technology. It is viewed as the first microsatellite built by Philippine engineers under the guidance of Japanese professors. It was successfully deployed in April 27, 2016 and has been operating since, providing imagery for earth observation. Diwata-2, its successor, a microsatellite of the same class is currently under development and is planned for deployment by the end of 2018.

Concurrent with these developments, the ground control and receiving station (GRS) is also being developed to support satellite mission operations. This initiative includes not just the setup of GRS hardware and software, but also research and development on data processing and management automation of acquired images for faster turnaround times after image acquisition and reception. As a developmental microsatellite, Diwata-1's images have unique characteristics to be accounted for e.g. non-sun synchronous orbit, low pointing accuracy, unique SMI payload characteristics (Paringit et al, 2016), etc. While proprietary and commercial software are available to serve this purpose, the cost and fit of purpose necessitates indigenous development (Aranas et al., 2016). The approach of "not reinventing the wheel" has been espoused by a number of system development ideologies with the aim of

improving and accelerating development. To hasten our development process, reuse of existing freely available open source software was encouraged.

One group of open-source software of particular interest for satellite earth observation are Free and Open-source Software for Geospatial Applications (FOSS4G). FOSS4G covers various applications for GIS and Remote Sensing (RS). These applications are usually categorized into three main groups 1) Desktop Applications, 2) Web Server Applications, and 3) Processing Libraries. One of the main selling points of these software are their interoperability, thus they are used on top of one another, hence the term stack.

In this paper, we focus on our efforts to process and manage DIWATA-1 images using a FOSS (Free and Open Source Software) stack.

RELATED WORK

Recent reviews suggest FOSS when utilized together, provides comparable performance to proprietary software. The work of Hall and Leahy (2008), provides detailed insights on the various aspects of the FOSS4G stack, it describes the applicability of these software for sharing, storing, and processing geospatial data for various use cases.

In the context of ground receiving station satellite operations, FOSS is primarily used for image processing however utilization of web applications and processing libraries have been used in other aspects such as planning and operations. Ground Receiving Station operations for earth observation have standard processes and protocols. These often include 1) mission planning and tasking, 2) image restoration, 3) image processing, 4) image management, and 5) image distribution (Rau, et al., 2003; Leptoukh, 2005).

Image restoration involves reconstruction of images received by the GRS, specifically data depacketing, and image generation. These images may not be completed in one pass, as missing data needs to re-downloaded from the satellite, hence this process includes processes for gap filling. Image pre-processing, on the other hand, involves geolocation or georeferencing, image correction, and calibration.

The use of FOSS for image reconstruction and pre-processing have been demonstrated by an early work by Maathuis and Retsios (2006) for meteorological satellites using Geospatial Data Abstraction Library (GDAL) in C++. In their work, a cost effective GRS data management system which allows for decompression, geolocation, and data conversions was also demonstrated.

One of the most common use of FOSS is in image processing. Zhao (2013) comprehensively reviewed free and open software for RS image processing. One of these most commonly used software is QGIS (QGIS Development Team, 2010), while primarily a GIS software, QGIS provides plug-ins and interfaces with different applications with more mature image processing capabilities such as GDAL, GRASS GIS, and Orfeo Toolbox (Grizonnet et al., 2017), among others. Another multifaceted application is ILWIS, a stand alone desktop GIS and RS application, was used for higher order data processing used by Maathuis and Retsios (2006).

While stand alone applications are easily deployed and can be used out of the box. Another approach is custom built systems using programming interfaces with FOSS4G libraries. An example of these is Ling et al.'s (2011) approach, where GEOS a geospatial manipulation and processing library, and Proj.4 a popular map projection library was used to develop their own desktop application for image processing. Whereas another method is the use of these libraries for a web based processing platform (Yoon, Kim and Lee, 2016), which allows browser based operation. Their work is an image processing platform utilizing open web standards and open source software, specifically Zoo and Orfeo Tool Box.

Image management and distribution are concerned with efficient storage and retrieval of data to oversee the above mentioned process and to provide access for stakeholders and end users. This can be accomplished by simple folder

based work structures or in larger organizations automated and workflow managers such as Catena by DLR and OOTD by NASA (Aranas, 2016). A simpler approach is to utilize Spatial Database management systems such as PostgreSQL/PostGIS, that allows for storing, indexing and querying of geospatially referenced image datasets.

CONCEPTUAL FRAMEWORK

Standard operations of Ground Receiving Station which receives the DIWATA-1 imagery involves 1) mission operations, 2) data retrieval, and 3) data processing, archiving and distribution. Figure 1, shows these tasks relative to the microsatellite and stakeholders. The basic concept of operations, include providing processed satellite imagery with complete metadata to our stakeholders. To achieve this, mission and operations team tasks the satellite, image retrieval and download is then scheduled based on priority criteria as defined from the microsatellite’s mission.

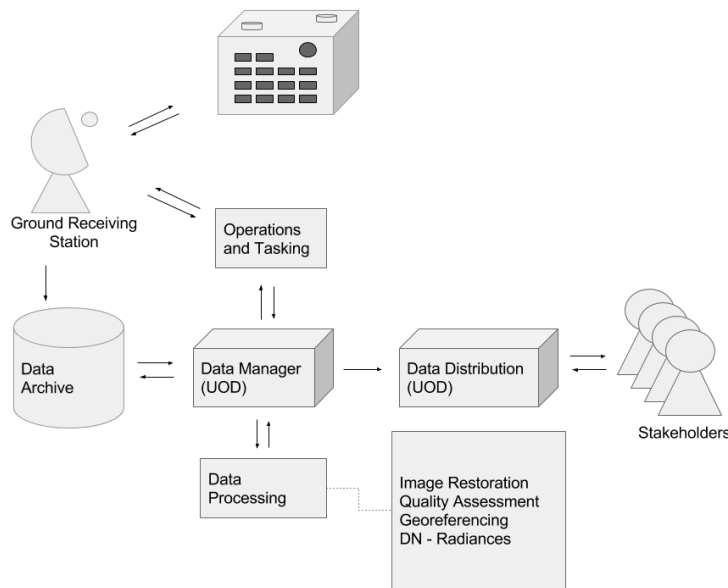


Figure 1. Conceptual Framework of PHI-Microsat Ground Receiving Station Operation

Once data packets are downloaded, images are reconstructed and metadata information is tracked. At this stage metadata information from the satellite such as acquisition mode, time of capture, etc are recorded. These along with images are then stored in a data archive. Archived images are then quality checked, thus additional information such as packet loss, pixel dropout, and cloud cover percentages are appended on its metadata. The images are then categorized based on the above mentioned quality metrics. Images are then subsequently processed. Images undergo geo-referencing and DN to radiance correction. These images are then bundled and again stored in the archive, and tagged as ready for distribution. Images ready for distribution is then cataloged and displayed in our distribution platform.

IMPLEMENTATION

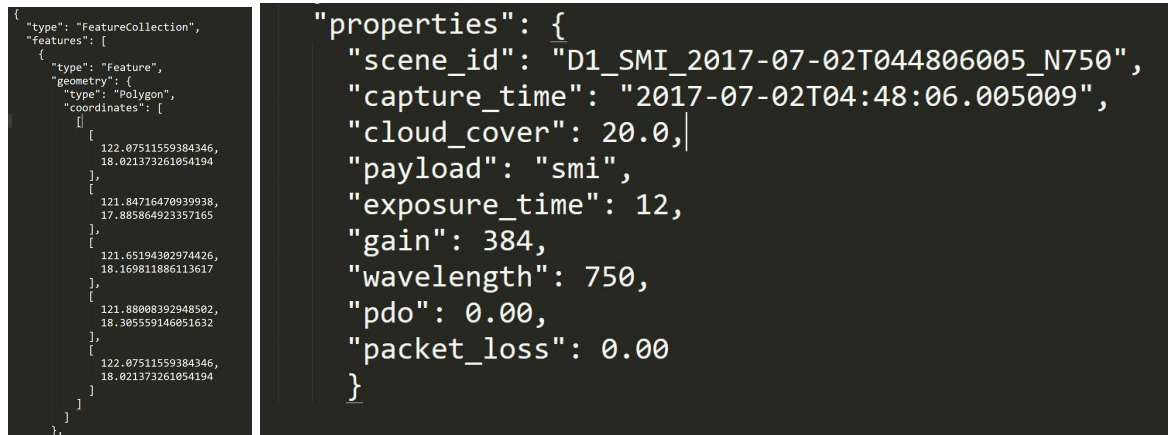
Mission Operations

In support of tasking and operations, satellite tracking and simulations are performed using Python (Rossum, 1991) based scripts using PyEphem. PyEphem is an open source Python library for ephemeris computations for natural and man-made satellites. It implements the Simple Generalized Perturbation (SGP) model, specifically SPG4 which utilizes Two Line Elements (TLE) to generate orbital simulations. From these simulations, satellite eclipse time,

satellite orbit projections, among others, are performed to assist in planning acquisitions and data retrieval scheduling.

Image Reconstruction and Quality Assessments

Upon data reception, data packets are converted into image format. Packet losses are tagged and metadata files are generated in GeoJSON format. GeoJSON is a human-readable format for representing geographical features and their associated attributes. GeoJSON, extends the Javascript Object Notation (JSON) and is gaining popularity as an exchange format for web based applications. When loaded in a GIS, DIWATA-1's metadata can be read as vector data for interoperability.



```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              122.07511559384346,
              18.021373261054194
            ],
            [
              121.84716470939938,
              17.885864923357165
            ],
            [
              121.65194302974426,
              18.169811886113617
            ],
            [
              121.88008392948502,
              18.305559146651632
            ],
            [
              122.07511559384346,
              18.021373261054194
            ]
          ]
        ]
      },
      "properties": {
        "scene_id": "D1_SMI_2017-07-02T044806005_N750",
        "capture_time": "2017-07-02T04:48:06.005009",
        "cloud_cover": 20.0,
        "payload": "smi",
        "exposure_time": 12,
        "gain": 384,
        "wavelength": 750,
        "pdo": 0.00,
        "packet_loss": 0.00
      }
    }
  ]
}
```

Figure 2. Sample Metadata in GeoJSON Format. It is composed of the Geographic Component (left) and the Attributes (right)

Subsequently, quality assessments of the images are performed using custom scripts based on GDAL, Rasterio (MapBox, 2013), and OpenCV (Bradski, 2000). These scripts include (but not limited to) computations for Pixel Dropout (PDO), Signal to Noise Ratio (SNR), and cloud cover assessments. While initial iterations of the system checked for packet losses and cloud cover, PDO and SNR assessments were added to better characterize image quality. To initialize this process, GDAL library is utilized using Python bindings e.g. Rasterio, to convert images into GeoTiff files for better interoperability. Throughout these processes GDAL/Rasterio is used to abstract the images as arrays that can be manipulated in other libraries such as NumPy (Ascher et al., 2001), SciPy, and OpenCV (Bradski, 2000).

Quality assessments are done throughout the processing steps. From the depacketing process, transmission losses are noted. Initial screening is performed based on packet losses, with those failing being tagged for redownloads. The next layer of check, involves PDO computations. To perform this, Laplacian filters are used from OpenCV. PDO is done to account for salt and pepper noise, once again a threshold is applied to PDO computations. Signal to Noise Ratio (SNR) is added to the metadata at this stage. SNR is calculated using the digital number standard deviation analysis and the homogeneous block method implemented in OpenCV. And lastly the cloud cover percentage for the image is computed with the the same set of Python libraries using Otsu method.

Semi-automated Georeferencing

Pointing constraints on the small spacecraft necessitated improvements on the geolocation procedures which include the development of an automated georeferencing platform using a PostgreSQL/PostGIS-based keypoint database derived using SIFT descriptors from OpenCV. An feature matching based approach to georeferencing is implemented with strong features from Master and Slave imageries serve as gcps for image to image registration. In our case, Diwata-1 SMI (slave) images are referenced with Landsat (master) images. For the system development,

extracted features are converted into point objects with its descriptors as its properties. These are then stored as point collections using PostGIS, hence serving as a GCP database.

While this is currently in progress, manual operations proceed using QGIS and its GDAL-based Georeferencer plugin. Due to DIWATA-1's orbit, successive frames are taken at each pass. These images contain significant overlaps. Image matching and feature extraction tests give operational results when matching with same imagery. To minimize manual operations, a pass stitching method is implemented wherein a couple of images per pass are manually georeferenced using QGIS. Then an OpenCV-based script utilizing FAST-SIFT-RANSAC combination is used to automatedly georeference the whole pass, using the georeferenced images as master.

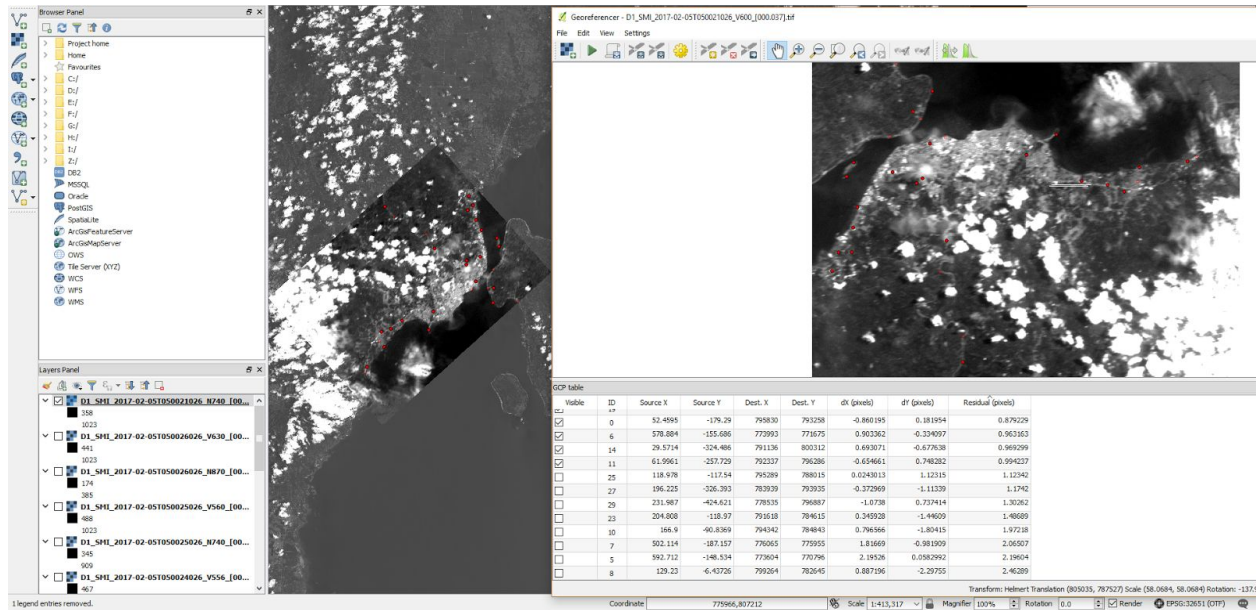


Figure 3. QGIS - Georeferencing (GDAL) Plugin used for Manual Georeferencing of SMI Images

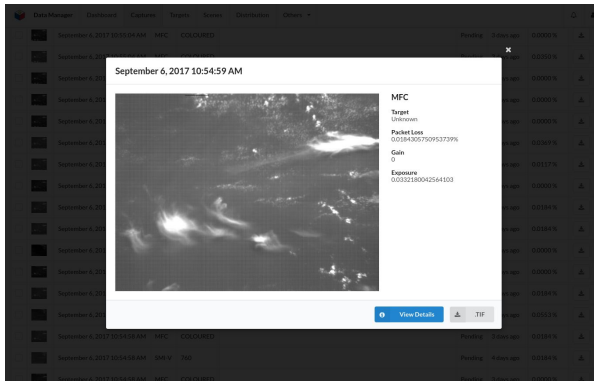
Image Management and Distribution

Overarching these are operations and management tasks, for which we developed an in-house browser based platform utilizing FOSS web technologies such as Django, Leaflet, among others. The system capability includes an orbit tracker, image management dashboards, and data distribution end point.

The Unified Operations Dashboard is composed of two main parts - the API and the Web Frontend. The API is a RESTful API built using Django. The Django REST Framework library was used to enforce RESTful features. Postgres with the PostGIS plugin was used as the main data storage for relational data. Data preprocessing and image reconstruction tasks (mentioned above) and bundling are being run asynchronously using Celery, a distributed task queue based on Python. The Web Frontend, which mainly works as a graphical user interface for the API, is a lightweight web application being served via Node.js using the ExpressJS framework. The frontend layout and styles were done using Semantic UI, while most of the its user interface handling were implemented using a combination of JQuery and Vue.js. Figure 4 shows selected tabs on this interface.

The above mentioned data quality metrics scripts are integrated within this data manager. For example, when images are received and uploaded, the system tracks image capture the unique identifiers based on its timestamp and payload. Each uploaded version of an image is cross referenced and merged using Python scripts to eliminate packet losses. The interface provides for promotion mechanisms to lead the images along the processing chain, with specific scripts semi-automatedly initialized. Simulations and projections are also graphed or mapped out using Leaflet and other graphing libraries.

a)



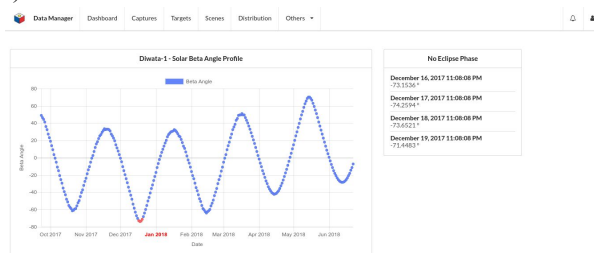
b)

Data Manager Dashboard Captures Targets Scores Distribution Others

CIV (Global) 11313 targets

Capture Time	Band	Wavelength	SRJ Address	Name	Status	Added
July 14, 2017 8:49:43 AM	SM-N	750	233	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:43 AM	SM-V	740	236	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:43 AM	MFC	0	445	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:43 AM	SM-V	680	235	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:43 AM	SM-N	740	232	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:43 AM	MFC	0	444	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:42 AM	SM-V	740	234	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:42 AM	MFC	0	443	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:42 AM	SM-N	750	231	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:42 AM	SM-N	740	230	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:42 AM	MFC	0	442	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:42 AM	SM-V	680	233	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:41 AM	SM-N	740	232	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:41 AM	MFC	0	441	Steel Island, Sulu	Pending	a month ago
July 14, 2017 8:49:41 AM	SM-N	750	229	Steel Island, Sulu	Pending	a month ago

c)



d)

Data Manager Dashboard Captures Targets Scores Distribution Others

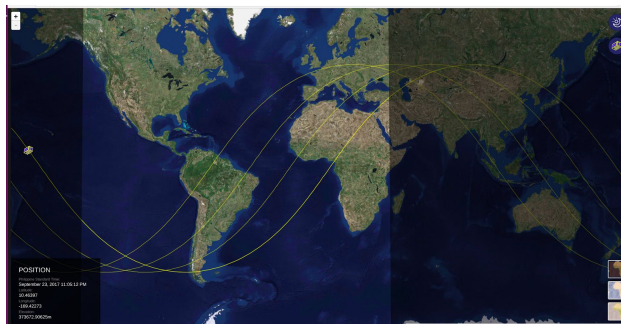
7411 captures

Capture Time	Band	Wavelength	Target	Status	Added	Packet Loss
Jun 27, 2016 11:37:21 AM	HPT-G	HPT-G		Pending	15 days ago	0.0000%
Jun 25, 2016 7:07:05 AM	SM-V	740		Pending	15 days ago	0.0000%
Jun 25, 2016 7:07:05 AM	SM-N	SM-N		Pending	15 days ago	0.2476%
Jun 23, 2016 6:19:44 PM	SM-N	SM-N		Pending	15 days ago	0.0000%
Jun 23, 2016 6:19:44 PM	SM-V	740		Pending	15 days ago	0.0000%
Jun 23, 2016 6:33:44 PM	SM-V	740		Pending	15 days ago	0.0000%
Jun 23, 2016 6:33:44 PM	SM-N	SM-N		Pending	15 days ago	0.0000%
Jun 23, 2016 3:58:52 PM	SM-N	SM-N		Pending	15 days ago	0.0000%
Jun 23, 2016 3:58:52 PM	SM-V	740		Pending	15 days ago	0.0000%
Jun 23, 2016 1:25:57 PM	SM-V	740		Pending	15 days ago	0.0000%
Jun 23, 2016 1:25:57 PM	SM-N	SM-N		Pending	15 days ago	0.0000%
Jun 23, 2016 12:06:20 PM	SM-V	740		Pending	15 days ago	0.0000%
Jun 23, 2016 12:06:20 PM	SM-N	SM-N		Pending	15 days ago	0.0129%
Jun 23, 2016 9:21:55 AM	SM-N	SM-N		Pending	15 days ago	0.0049%

Figure 4. Sample Panels of Unified Operation Dashboard. a) Image Viewer, b) Targets Interface - tracking of acquired images, c) Solar Beta Angle - tracking eclipse satellite phase, d) Captures Interface - tracking downloaded images, shows percent

Processed images are tagged on the interface for distribution once all metadata entries are accomplished and the image rectified. These are then published on a distribution platform based on the same technologies e.g Django, Leaflet. This is published on the main website of the research program including the ther application for tracking and image acquisition requests. The tracking application based on open source web mapping solutions utilizes a custom built API leveraging PyEphem and updated NORAD Two Line Elements (TLE). This interface provides a visual pseudo real time representation of the current location of the satellite. Figure 5. Shows the applications.

a)



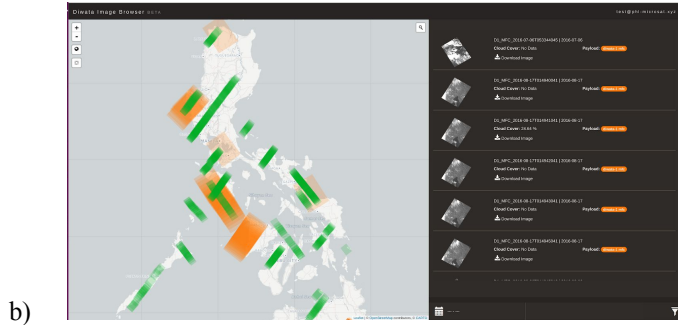


Figure 5. PHL-Microsat Applications a)Satellite Tracker, b) Data Distribution Site

CONCLUSIONS

Customizing for fit of use, performance, and overall improvements; these developments have already undergone several iterations and FOSS changes. Initial developments using processing and management systems such as Apache OODT has proven cumbersome and too complex for microsatellite operations. Instead, an in house developed simplified data management system was developed, the Unified Operations Dashboard (UOD). UOD integrates open source Python scripts using various science libraries such as PyEphem, NumPy, and OpenCV. These scripts handle primary tasks including image restoration, quality assessment and pre-processing. These scripts are augmented by manual processing performed using stand alone desktop applications such as QGIS for higher level tasks.

In all we utilized open source software for 1) scientific computing- for custom assessments and processing, 2) databases- data storage and management, 3) web development- for an integrated dashboard type interface and stakeholder facing applications. Theses softwares interoperate and form the bulk of the tasks in our concept of operations. With the lessons learned, systems and platforms developed; increases our operational capacity in preparation for Diwata-2 and other upcoming Philippine satellites.

ACKNOWLEDGEMENTS

This work is supported by the Department of Science and Technology - Philippine Council of Industry, Energy, and Emerging Technology Research and Development (DOST-PCIEERD) under the Development of the Philippine Scientific Earth Observation Microsatellite (PHL-MICROSAT – Project 3) Program.

REFERENCES

- Paringit, E. C., Viray, F. M., Maestro, M. M. & Calvo, J. S. (2016) Implementation of a Solar Spectral Model for the Calibration of the Spaceborne Multispectral Imager (SMI) of DIWATA 1. Proceedings of Asian Conference on Remote Sensing
- Aranas, R. K. D., Jiao, B. J. D., Magallon, B. J. P., Ramos, M. K. F., Amado, J. A., Tamondong, A. M., and Tupas, M. E. A. (2016) Design of a Free And Open Source Data Processing, Archiving, And Distribution Subsystem For The Ground Receiving Station Of The Philippine Scientific Earth Observation Micro-Satellite, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLI-B7, 909-911.
- Ling H., Fada T., Minglu L. (2011) Design and Implementation of the Image Processing Algorithm Framework for Remote Sensing. In: Liu C., Chang J., Yang A. (eds) Information Computing and Applications. ICICA 2011. Communications in Computer and Information Science, vol 244. Springer, Berlin, Heidelberg

Grizonnet, M., Michel, J., Poughon, V. et al. (2017). Orfeo ToolBox: open source processing of remote sensing images. *Open geospatial data, softw. stand.* (2017) 2: 15.

Yoon, G., Kwangseob Kim and K. Lee, (2016) Design and implementation of geo-spatial image processing system using OGC WPS 2.0 and Web framework on Openstack cloud, *2016 4th International Workshop on Earth Observation and Remote Sensing Applications (EORSA)*, Guangzhou, 2016, pp. 132-135.

Hall, G. B. and Leahy, M. G. (2008) *Open Source Approaches in Spatial Data Handling*, Springer.

Bradski, G. OpenCV Library. (2000) *Dr. Dobb's Journal of Software Tools*.

QGIS Development Team. (2010) QGIS Geographic Information System. Open Source Geospatial Foundation Project. <http://qgis.osgeo.org>.

Zhao Y. (2013) Towards Open Source Remote Sensing Software – A Survey. In: Bian F., Xie Y., Cui X., Zeng Y. (eds) *Geo-Informatics in Resource Management and Sustainable Ecosystem. Communications in Computer and Information Science*, vol 398. Springer, Berlin, Heidelberg

B.H.P. Maathuis, V. Retsios (2006). Installation, setup and use of a low cost C-band Meteosat-8 ground receiving station in Rwanda. AARSE 2006: Proceeding of the 6th AARSE International Conference on Earth Observation and Geoinformation Sciences in Support of Africa's Development, 30 October–2 November 2006, Cairo: The National Authority for Remote Sensing and Space Science (NARSS), Cairo, Egypt

Ascher, D., Dubois,P.F., Hinsen,K., Hugunin, J., Oliphant, Tim T. (2001) “Numerical Python.” <http://www.numpy.org/> (July 30, 2017).

Dahl, Ryan. 2009. “Node.js.” <https://nodejs.org/en/> (January 15, 2016). Development Seed. 2015. “Libra.” <http://libra.developmentseed.org/> (July 30, 2017).

Kim, Kyung Hee et al. 2011. “Ground Station Design for STSAT-3.” *International Journal of Aeronautical and Space Sciences* 12(3): 283–87.

Leptoukh, Gregory G. (2005) “NASA Remote Sensing Data in Earth Sciences: Processing, Archiving, Distribution, Applications at the GES DISC.” Manoochehri, Michael. 2013. *30 Data Just Right: Introduction to Large-Scale Data & Analytics*. Addison-Wesley. https://books.google.com/books?id=kmlCagAAQBAJ&p_gis=1 (August 10, 2017).

Mapbox. (2013). “Rasterio.” <https://www.mapbox.com/blog/rasterio-announce/> (August 12, 2017).

Rau, Jiann-Yeou et al. 2003. “AN OPERATIONAL MULTISENSOR GEOCODED PRODUCTION SYSTEM FOR EARTH RESOURCES SATELLITE IMAGES.” In *Proceedings of the 5th Int. Symposium on the Reducing Cost of Spacecraft and Ground Systems and Operations*, Pasadena, USA.

Rossum, Guido van. 1991. “A High-Level Interpreted Language Combining Ideas from ABC, C, Modula-3, Icon, Etc. Intended for Prototyping or as an Extension Language for C Applications. Modules, Classes, User-Defined exceptions. “Linking a Stub Generator (AIL) to a Prototyping Language.” <https://www.python.org/> (July 22, 2017).