PRJ2-1
Invited

JAXA Earth API for Python : Open-Source API for Efficient and Effective Satellite Data Distribution

Yoshinobu Sasaki[1], Kohei Kawamura[1], Goh Segami[1], Kei Oyoshi[1]

sasaki.yoshinobu@jaxa.jp
[1]Earth Observation Research Center, Japan Aerospace Exploration Agency, 2-1-1 Sengen, Tsukuba, Ibaraki, Japan
Keywords: Satellite, API, COG, STAC

ABSTRACT

Specifications of earth observation images differ among satellites, sensors, and distribution systems, so the cross-sectional use of the data is still difficult. Therefore, we are developing the prototype of satellite data distribution API, "JAXA Earth API for Python", which enables us to use JAXA's multiple satellite data immediately and intuitively.

1 Introduction

JAXA has launched and operated various earth observation satellites. The obtained Earth observation satellite data and various geophysical products generated based on the data are distributed from distribution sites such as G-Portal [1].

Specifically, JAXA delivers products such as elevation (DSM), precipitation rate (RainRate), normalized difference vegetation index (NDVI), land surface temperature (LST), aerosol optical thickness (AOT), soil moisture content (SMC), land cover classification system (LCCS), short-wave solar radiation (SWR) and sea ice concentration (IC0) [2][3][4][5].

Because of the usefulness of these satellite data, which can be obtained without the need for ground-based observation equipment, each product has been used for different purposes. Specifically, elevation data is used for land use studies, flood and drainage modeling, etc., precipitation rates are used in developing countries where ground observation radars are not sufficiently equipped, and AOT products are used to determine air pollution conditions.

In addition, satellite data applications and technologies that integrate the use of multiple geophysical products have been gaining momentum in recent years. For example, in the fields of agriculture and public health, research is being conducted on the distribution of crop status information and the comprehensive understanding and prediction of ecological distribution using a combination of altitude, vegetation index, soil moisture, and other data.

However, the various geophysical products distributed generally differ in their map projection methods (equirectangular and sinusoidal projections), latitude and longitude ranges of the data they contain, resolution, file naming conventions, file formats (NetCDF, HDF, GeoTIFF, etc.), and data distribution sites, depending on the type of satellite, sensor, geophysical quantity, etc. Therefore, in order to use different geophysical products in a composite manner, one has to have expertise in remote sensing and map projection. In addition, latitude/longitude ranges and physical quantities not required by the user may also be included and distributed, which is inefficient from a data retrieval perspective. Furthermore, since data retrieval is not fully automated, it is time-consuming to manually retrieve data using a browser GUI under the current conditions. Therefore, from the perspective of providing data to users who use it in a cross-sectional manner, this is not necessarily an optimal data delivery method.

Therefore, we have developed and released a prototype of the JAXA Earth service, which integrates the satellite data products distributed by JAXA into two main data formats, COG (Cloud Optimized GeoTIFF) and STAC (Spatio Temporal Asset Catalog), to realize efficient and effective data distribution [6].

In the JAXA Earth services, the API for Python, which enables data acquisition in Python, can realize more efficient and effective data distribution than conventional methods [7]. Specifically, it minimizes the time and effort required for users to manually search for products and enables them to acquire and process only the amount of data optimized for their area of interest and products, without being aware of differences in format or other data differences.

This paper introduces the data, specifications, functions, and use cases available in the API.

2 JAXA Earth API for Python

The overall structure of the JAXA Earth service, which provides data via API, is shown in Figure 1. It consists of a database, API, and web application.

We have developed two versions of the API: a Python language version, which is well known in the field of data science, etc., and a JavaScript language version, which is essential for generating dynamic applications in a browser. Both APIs are in a format that operates on client-side (i.e., using excess computing power on the user side, which is becoming more powerful these days). The API for Python, which was developed and released earlier, is open source so that users can customize the internal programs as needed.

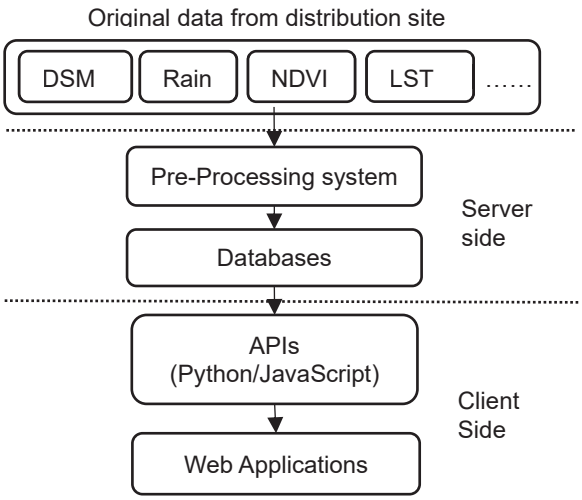

Fig. 1 "JAXA Earth" services data distribution flow

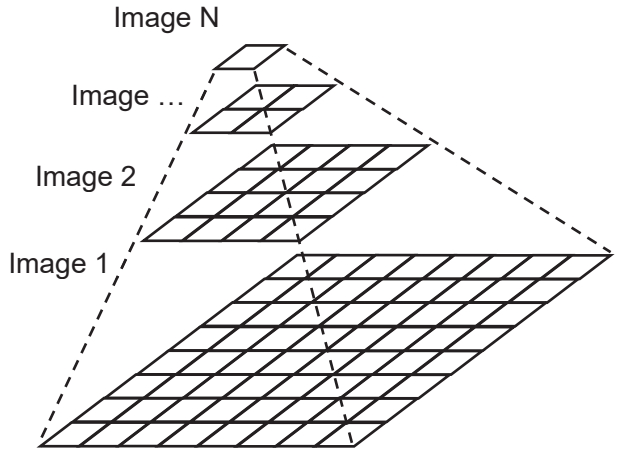

Fig.2 COG Internal file structure (images only)

2.1 Data format used in the service

2.1.1 COG (Cloud Optimized GeoTIFF)

COG is one of the types of the GeoTIFF data format. Normal GeoTIFF data stores a single image of the highest resolution. In contrast, as shown in Figure 2, COG sequentially stores the highest-resolution image plus an image that has been down sampled to twice that resolution. Each image is stored as a set of the fixed tile size (e.g., 256 x 256 pixels). Each tile is data compressed in tile units.

The COG data contains metadata at the beginning of the COG data, called IFD (Image File Directory), which indicates the location where the tiles are stored. Therefore, the user first accesses the COG stored on the server by HTTP range request to obtain the IFD. Then, by comparing the tile storage location information in the IFD with the latitude and longitude range information the user wants to obtain, the user can retrieve only the tiles with the required resolution and latitude and longitude range. Tiles are sent and received in a compressed state, so the amount of data transmission is also optimized.

The internal structure of COG (IFD with tiled multi-resolution data and tile position information) and the data acquisition method (HTTP range request) described above allow users to efficiently acquire data only in the required range. Minimizing the amount of data transmission also reduces the burden of communication volume and facility maintenance on the data distributor side. Therefore, it can be said that data delivery in COG format is beneficial to both users and data distributors.

2.1.2 STAC (Spatio Temporal Asset Catalog)

STAC is a JSON format representation of geospatial data such as satellite data, metadata, and its storage location. By using COG as a set, STAC data are cataloged and data access become easy.

There are two ways to implement STAC: a dynamic implementation that returns STAC that change the data queried by the server, and a static implementation that generates and stores all STAC. For our service, the static implementation method was chosen because it simplifies delivery and maintenance.

Since the STAC specification is separated from the COG specification, it is possible to decentralize the storage location of the COG, which is the main body of data, to avoid access concentration and to demarcate the scope of management. In addition, STAC has the scalability to be updated to a format suitable for search and display at any time, and to add more databases.

2.2 Database for the API

The database includes more than 70 types of data shown in Table-2, including elevation data, rainfall rate data, vegetation index, surface temperature, etc. distributed by JAXA. Since the current service is a prototype, the time range of the product is limited (mainly only data for the year 2021) to allow for flexible specification changes and updates in the future.

As for resolution, it depends on the product, but data are stored from the global level up to the 30m level. Frequency of data ranges from daily observations to annual releases. Since it is inefficient for data management to have all levels from 30m to the global in one COG, the files are divided by keeping a maximum of three image levels in each COG. Therefore, the resolution stored in this database is discrete rather than continuous. It should be noted that satellite images with

a resolution equal to or higher than the resolution of the source data are stored in the database.

The projection method currently supports two types of map projection formats: equirectangular (EPSG:4326) and polar stereo (EPSG:3995).

The data is released as open data that does not require authentication, so anyone can access it without registration for an account.

Table-1 List of products stored in the database

No.	Category	Description
1	DSM	Digital Elevation Model provided by AW3D30
2	Rain Rate	Rain Rate products provided by GSMaP
3	LST	Land Surface Temperature products provided by JASMES, G-Portal, and so on.
4	NDVI	Normalized Vegetation Index products provided by JASMES, G-Portal, and so on.
5	LCCS	Land Cover Class products provided by ESA CCI
6	SMC	Soil Moisture Contents products provided by G-Portal
7	SWR	Short Wave Radiation products provided by JASMES
8	AOT	Aerosol Optical Thickness products provided by G-Portal and JASMES
9	IC0	Sea Ice Concentration product provided by JASMES

2.3 Main function of the API

The API can be used by installing or unzipping the files listed on the reference page.

The API has been designed and produced to perform the basic function of acquiring a desired satellite image as a numpy array by specifying the product ID, date range, latitude/longitude range, resolution, etc. If the product ID and other arguments are omitted, the default settings described inside the API will be used. The minimum code to execute the API and visualize images is shown in Figure 3.

In addition to the basic functions, the API also has the functions shown in Table 1. adjust the color bar and color range of the displayed image, composite using multiple images, masking, statistical processing, and graph display are available. It is also possible to set arbitrary polygons using GeoJSON for the range of interest. Detailed specifications, usage, and examples of use are separately maintained and published as an API reference [7].

An important feature of the API is its ability to interface with QGIS, a major free GIS software program. The API can automatically retrieve the latitude and longitude ranges and resolutions displayed in QGIS, allowing satellite images to be acquired and displayed without the need to manually set these parameters. Therefore, users can acquire satellite images just by setting up areas of interest interactively on the QGIS screen and executing the API. Figure 4 shows an example of the Python API in QGIS.

The data loaded in QGIS is temporarily saved as a layer, so that color bars, color ranges, etc. can be adjusted even within QGIS.

As described above, the API minimizes the time and effort required for users to manually search for products and enables only the optimized amount of data to be delivered to users without being aware of differences in format or other factors.

```
# Load module
from jaxa.earth import je

# Get an image
data = je.ImageCollection(ssl_verify=True)\
          .filter_date()\
          .filter_resolution()\
          .filter_bounds()\
          .select()\
          .get_images()

# Process and show an image
img = je.ImageProcess(data)\
         .show_images()
```

Fig. 3 Minimum Python script to run the API

Table-2 Main function of the API for Python

No.	Function	Description
1	ID Search	Collection ID searching by keywords
2	ROI select	ROI selection by GeoJSON
3	Masking	Masking the image by bit or value range or exact value.
4	Composite Processing	Compositing multiple timeseries image
5	Differential processing	Calculation of the difference image from the reference image
6	Timeseries Processing	Calculation of spacial statistics of multiple timeseries images
7	Visualization	Showing images and timeseries graphs, color bar settings, color range settings
8	Interface	Interface to execute in QGIS

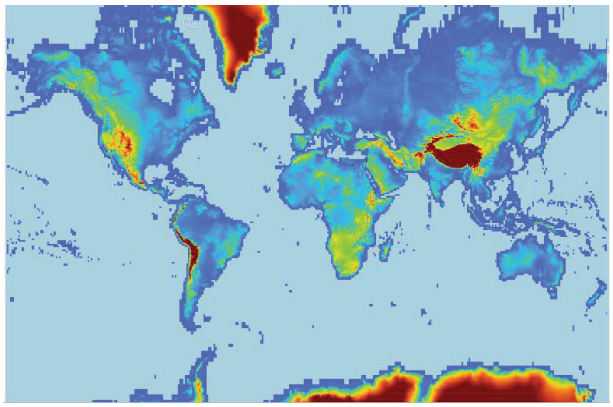

Fig.4 AW3D Data in QGIS visualized by the API

3 Use cases for interdisciplinary studies

The use of this API is expected to increase users in other fields regardless of existing satellite data use areas. Some possible use cases are described below.

3.1 Accelerating conventional research

The use of this API is expected to accelerate research on the cross-sectional application of existing satellite data in agriculture and public health fields, etc. By using the API, researchers do not need to worry about extraneous matters such as format differences between products and can concentrate on the research itself.

3.2 Education for high school students

In 2022, the Courses of Study for high school students in Japan will make geography compulsory, and students will be required to learn how to use GIS (Geospatial Information Systems). Although Web GIS is recommended in education, Python API has an interface function to QGIS and can interactively acquire and visualize satellite data. Therefore, it is considered to be a useful tool for students who are not satisfied with the GIS classes at school, or for students who want to learn programming.

3.3 Various data sciences and services

This API will enable researchers and IT engineers from various fields who have never used satellite data before to acquire and use satellite data.

By creating new users of satellite data, research and service on new indicators using satellite data is expected. In addition, since the API can automatically acquire data, it can also promote the use of IoT devices that can run Python, and satellite data is expected to play an active role as part of the API economy in the future.

4 Conclusions

This API is more efficient (reduced data transmission and manual operation) and effective (necessary processing on the client side, direct interface to QGIS) JAXA satellite data distribution system.

In the future, in addition to continuous updates of the API for Python and the database, we plan to develop and release the API for JavaScript and additional web applications.

References

[1] JAXA G-Portal (https://gportal.jaxa.jp/gpr/?lang=en)

[2] ALOS Global Digital Surface Model "ALOS World 3 D - 30m (AW3D30)" (https://www.eorc.jaxa.jp/ALOS/en/aw3d30/index.htm)

[3] Global Satellite Mapping of Precipitation : GSMaP (https://sharaku.eorc.jaxa.jp/GSMaP/index.htm)

[4] JAXA Satellite Monitoring for Environmental Studies :JASMES (https://kuroshio.eorc.jaxa.jp/JASMES/index.html)

[5] ESA CCI Land cover website (https://www.esa-landcover-cci.org/)

[6] JAXA Earth Portal (https://data.earth.jaxa.jp/en/)

[7] JAXA Earth API for Python API Reference (https://data.earth.jaxa.jp/api/python/)

[8] Y. Sasaki et al, “JAXA Earth : Prototype of Earth Observation Data Distribution System and Application for Interdisciplinary Studies”, International Display Workshop 2021 (https://doi.org/10.36463/idw.2021.0622)