EARTH OBSERVATION SIMULATOR (EO-SIM): AN OPEN-SOURCE SOFTWARE FOR OBSERVATION SYSTEMS DESIGN

Vinay Ravindra, Ryan Ketzner, Sreeja Nag

Bay Area Environmental Research Institute, NASA Ames Research Center
vinay.ravindra@nasa.gov

ABSTRACT

This paper presents the Earth Observation Simulator (EO-Sim), a software framework which facilitates the design of novel observation systems. EO-Sim allows exploration of observing strategies by facilitating users to configure and simulate heterogenous satellite constellations. A set of potential observation opportunities and the associated observation metrics during mission-operations can be generated by the simulations. EO-Sim also incorporates an observation simulator to mock the operation of instruments taking into consideration the instrument specifications and observation geometry. A beta version has been made available to the public.

Index Terms— remote-sensing, adaptive sensing, synthetic imagery, open-source

1. INTRODUCTION

The capabilities of Earth Observation (EO) missions have seen steady growth in the past many years, assisted by developments of small-satellite bus (e.g. the 3U, 6U, 12U CubeSat and microsat bus from Blue Canyon [1]), new instruments (e.g. compact Ka band precipitation radar [2]), large inter-connected ground-station networks (e.g. Amazon Web Service Ground Station [3]), intersatellite communications [4], onboard data processing (e.g. SpaceCube [5]) and reduced launch costs (e.g. launch services from SpaceX, Rocketlabs). These technological capabilities further allow development of new observation systems involving intelligent and collaborative constellations [6].

This expansion of technological capabilities has given risen to the need of simulation tools which can consider these technologies and assist in the process of design and analysis of future Earth observation missions. The General Mission Analysis Tool (GMAT) [7] and Systems Tool Kit (STK) [8] support quality orbit propagation and satellite maneuver modeling for EO and deep-space missions. The Tradespace Analysis Tool for Designing Constellations (TAT-C) [9] facilitates Pre-Phase A investigations and optimizes EO constellation designs considering

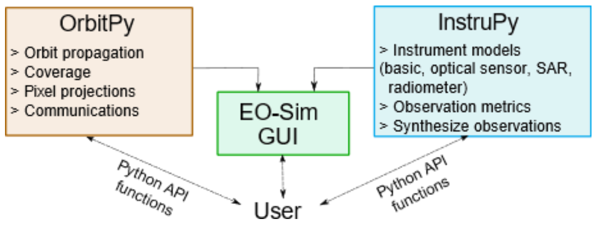

Fig. 1. Structure of EO-Sim consisting of packages OrbitPy, InstruPy and EO-Sim GUI.

performance, cost, and risk for predefined science goals. The Virtual Constellation Engine (VCE) [10] facilitates exploration of different remote sensing constellation and sensor web configurations from an on-board processing perspective, using the cloud. COLLABORATE [11] offers simulation capabilities to developers of future observing system simulation experiments (OSSEs) with collaborative networks. DSHIELD [12] shall offer suite of scalable software tools that helps schedule payload operations of a large constellation, considering constraints imposed by orbital mechanics, (solar) power systems, and attitude control systems and observation value dictated by a simulator of the natural phenomenon of interest.

In this paper we introduce the Earth Observation Simulator (EO-Sim) tool being developed as a part of the DSHIELD project. The beta version EO-Sim [13] which is available under a permissive open-source license adds to the growing list of aids for designing future EO missions. The distinguishing feature offered by EO-Sim over other tools is that it incorporates common EO instrument models which are used to produce synthetic satellite imagery. It offers this feature in a user-friendly, comprehensive setup involving the simulation of components of a remote sensing mission (orbit and coverage, inter-satellite and ground-station communications).

The rest of the paper is organized as follows: Section 2 describes the features and design of the EO-Sim tool. Section 3 describes potential applications and in Section 4 we conclude.

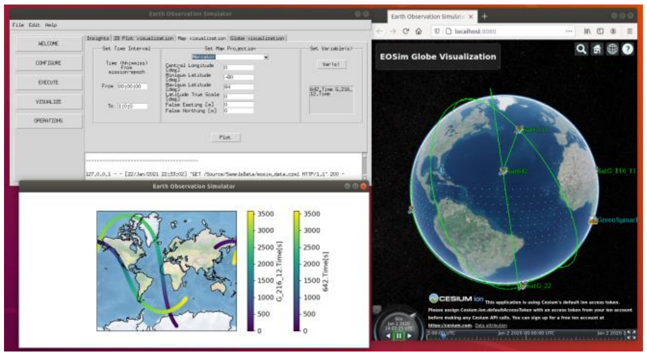

Fig. 2. Screenshot the EO-Sim desktop GUI.

2. FEATURES AND DESIGN

EO-Sim is split into three separate python packages as depicted in Fig.1, the purpose of which is to allow for modular development and integration into user applications. OrbitPy provides various orbit calculation related functionalities, InstruPy contain instrument models to simulate instrument data, and EO-Sim GUI provides users with a friendly graphical-user interface to setup a simulation environment and visualize the results. Alternatively, users may invoke the functionalities of the OrbitPy and InstruPy packages using API calls.

2.1. Desktop Interface

The desktop GUI was developed using TkInter [14]. A screenshot of the desktop application is shown in Fig.2.

2.1.1. Input interface

Standard widgets such as buttons, radio-buttons, checkboxes, user entry fields allow the user to configure a constellation architecture, i.e., the satellites (orbits), instruments (basic-sensor /optical sensor /synthetic aperture radar / radiometer models) with their respective orientations. An heterogenous configuration of satellites and instruments, allowing for definition of multiple, different instruments per satellite is possible. Satellite/instrument maneuverability can also be specified, which constrains the angular space within which the instrument maybe maneuvered. For example, since stripmap SARs are primarily side-looking instruments, a roll-only maneuver can be specified. Ground-stations and regions of interest can be specified, over which contacts/coverage opportunities are calculated. The input configuration can also be exported (and can be re-imported) in the form of a JSON file which provides for human-friendly readability.

2.1.2. Output interface

There are several ways to view the results of the execution (which are described in the later sub-sections). The raw data-files are stored in the user working directory which can be picked up by the user for further processing. Using the Matplotlib python package, X-Y plots can be built on the available datasets where the x-axis and y-axis variables are customizable. The CartoPy python package [15] is used to provide several configurable map-projections (e.g., Mercator, equidistant-conic, Lambert-conformal) on which data (e.g., satellite-positions) can be plotted over a customizable time-interval. The CesiumJs [16] library is used to provide the user with a 3D visualization of the mission (i.e., animation of satellite orbits, observations, communications).

2.2. Orbit Propagator

A basic requirement of a remote-sensing mission development tool is its orbit propagator, which calculates the satellite states (position, velocity) at different times of the mission. An analytical propagator considering the perturbations from the J2 term of the Earth's gravity field is used in EO-Sim, modeled on the framework presented in [17, Chap 9]. Only the secular drift in the Keplerian terms is modeled. The results of the propagation are useful for LEO orbits, and few weeks of analysis and is representative of well-maintained orbits (without modeling of maneuvers). EO-Sim allows to import the results of other high-fidelity orbit-propagators available in GMAT, STK.

2.3. Coverage (Access) Calculator

Coverage opportunities are defined as the times when a ground-station/region can be accessed by a satellite-instrument combination during the mission. EO-Sim provides for three different types of coverage calculations:

2.3.1 Grid based coverage

A region is specified by a set of grid-points (latitude, longitude). The access intervals for each instrument in every satellite of the constellation is calculated over all the grid-points. The calculation involves checks to see if a grid-point lies within the instrument field-of-view. More description of this approach is available in [18].

2.3.2. Pointing-options based coverage

A set of pointing-options is associated with each instrument of a satellite. A pointing-option characterizes the satellite/instrument orientation with respect to the local frame (e.g. LVLH) and is representative of the possible maneuvers by a satellite. The intersection point (latitude, longitude) between the instrument pointing-axis and Earth (assumed spherical shape) is calculated at each propagation time-step.

2.3.3 Hybrid coverage

A hybrid of the above two approaches, involves specifying both set of grid-points and set of pointing-options. The set of grid-points per pointing-option is calculated at each propagation time-step.

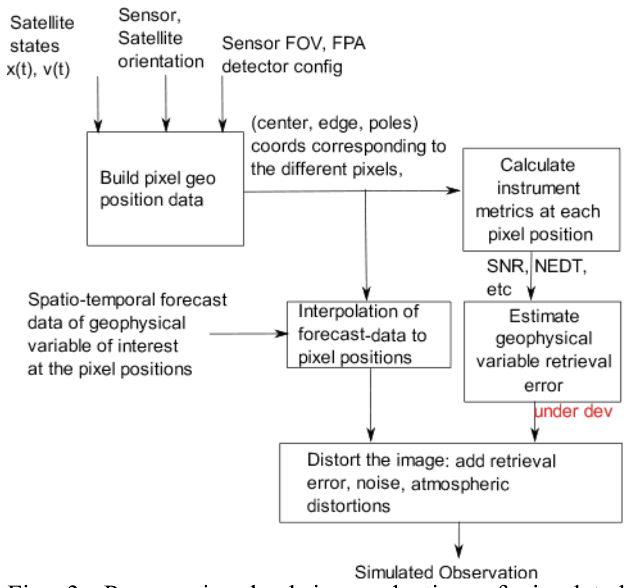
Satellite states x(t), v(t)
Sensor, Satellite orientation
Sensor FOV, FPA detector config
Build pixel geo position data
(center, edge, poles) coords corresponding to the different pixels,
Calculate instrument metrics at each pixel position
SNR, NEDT, etc
Spatio-temporal forecast data of geophysical variable of interest at the pixel positions
Interpolation of forecast-data to pixel positions
Estimate geophysical variable retrieval error
under dev
Distort the image: add retrieval error, noise, atmospheric distortions
Simulated Observation

Fig. 3. Process involved in production of simulated observations for a sensor with Focal Plane Array (FPA).

2.4. Communication Contact Opportunity Calculator

The calculation of communication time-intervals between pairs of entities, where an entity can be a satellite or ground-stations, involves computation of line-of-sight [19, pg. 198] between the entities (with Earth as the possible occluding body). The range and elevation angle (in case of ground-stations) is recorded for each time-step at which line-of-sight exists.

2.5. Observation Metrics Calculator

Observation metrics can be calculated for potential observations made by the satellite-instrument pair during the mission. There are three instrument models available: (a) basic sensor (b) passive optical sensor (which includes stripmap, whiskbroom and matrix imagers), (c) synthetic aperture radar (SAR) and (d) radiometer. The basic sensor allows for calculation of simple metrics such as incidence angle, range, Sun zenith angle, the passive optical sensor model calculates the Signal to Noise Ratio (SNR), Noise-Equivalent Delta Temperature (NEDT) and the SAR model allows for calculation of the noise-equivalent sigma zero. A detailed description of the instrument models is available in [20].

2.6. Simulated satellite imagery

The process for producing simulated satellite imagery for matrix imagers with Focal Plane Array (FPA) is shown in Fig.3. It involves the projection of the (rectangular) FPA detector dimensions onto the surface of Earth (spherical model) based on the framework presented in [17, chapter 8]. The projected ground-pixels are characterized by their center-positions (geo-coordinates), corner-positions and "poles" of the pixel edges. The poles correspond to the center of the small-circle arcs which connect the ground-pixel corners and form the pixel edge.

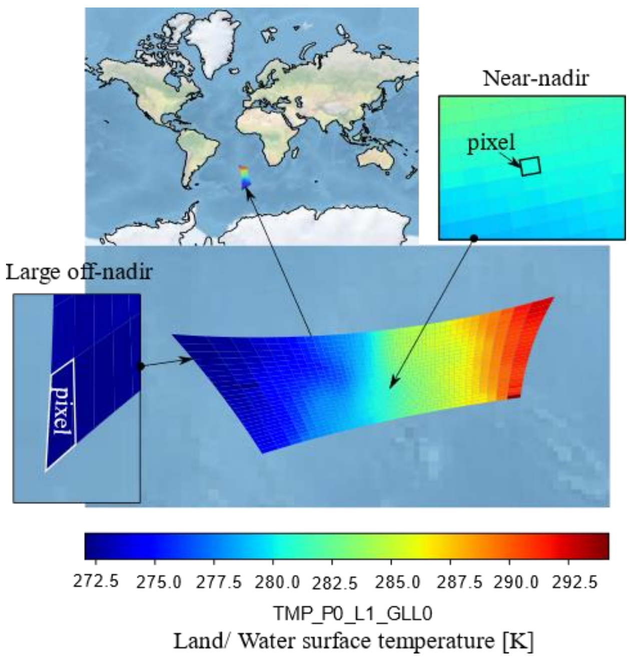

Fig. 4. Example of simulated imagery of instrument observing the surface temperature. The simulation is of a satellite at 500km Sun-synchronous orbit making observation at 10deg off-nadir using an instrument with field of view 60 deg along-track and 120 deg cross-track. The non-homogeneity of the pixel shapes can be seen.

Geophysical forecast data from dynamic weather models such as the Global Forecast System (GFS) can be selected by the user as the source of underlying observation forecast data. An appropriate geophysical variable (e.g., surface temperature, precipitation rate) corresponding to the associated instrument is chosen by the user. This data is projected onto the (non-uniform) grid formed by the ground-pixels by interpolating the source data in spatial dimensions to the pixel center-positions. Several interpolation schemes from MetPy [21] are available for selection by the user. In case of the temporal dimension, the forecast data corresponding to the nearest observation time is chosen (nearest-neighbor interpolation).

In the next step the projected data is distorted to include effects of instrument noise, non-linearities and speckle if applicable (under development). The simulated observation can be said as close to the Level-2 data product from instruments [22]. There is ongoing work to extend the framework to other types of instruments (such as pushbroom sensors, stripmap radars) involving line-scanning methodology.

3. APPLICATIONS

There are two primary target applications for which the results of EO-Sim may be utilized. The first in development

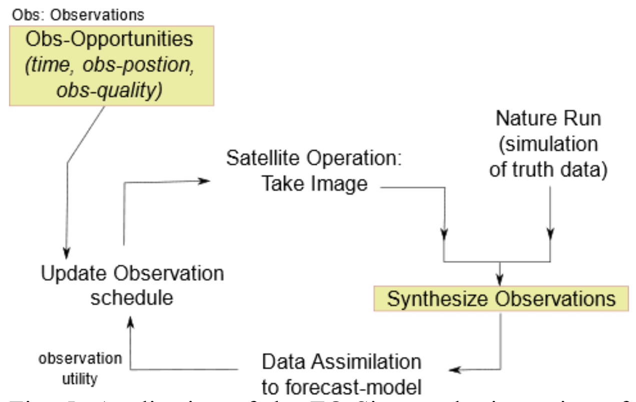
Obs: Observations
Obs-Opportunities (time, obs-postion, obs-quality)
Nature Run (simulation of truth data)
Satellite Operation: Take Image
Update Observation schedule
Synthesize Observations
observation utility
Data Assimilation to forecast-model

Fig. 5. Application of the EO-Sim results in testing of adaptive sensing algorithms.

of adaptive sensing algorithm as portrayed in Fig.5. Adaptive sensing involves the frequent updating of observation schedules (i.e., where and when to make observations) of satellite according to the all the current knowledge of the phenomenon of interest. EO-Sim may be used to provide the set of all available observation opportunities, i.e., the time, position (coverage calculations) and observation quality (observation-metrics). The synthetic observations may be used to feed into the forecast model a simulation of the observation made by a satellite. The scheduler would make an appropriate schedule after consideration of constraints (satellite maneuver, power, etc.) and the utility of making observation.

The second application interest is simulation of instrument data during the development of new instrument concepts. TROPICS [22] and Raincube [2] are examples of recent missions which use dynamic model forecast data for development, evaluation of instrument concepts. EO-Sim offers the additional ability to obtain simulated data on the non-uniform ground-pixel grid, and hence a closer approximation to the instrument output. This is especially required while considering agile-imaging concepts in which observations shall be made over wide range of off-nadir orientations.

4. CONCLUSION

EO-Sim offers a comprehensive development environment for testing out new observing systems and validating the same. It considers various instrument models and can simulate observation quality and observations themselves to mock the satellite remote-sensing operations.

A beta version is made available [3] to the public. Future work shall involve extending the concept of synthetic satellite imagery to instruments which observe using a line-scanning methodology (e.g., pushbroom sensors, stripmap SARs) and the inclusion of retrieval error in synthetic imagery. Additional instrument models such as Doppler-radars and scatterometers shall be added to the instrument suite.

5. REFERENCES

[1] "Blue Canyon Technologies," Web, URL: https://www.bluecanyontech.com/spacecraft
[2] E. Peral, et al, "Raincube: A proposed constellation of precipitation profiling radars in CubeSat," *2015 IEEE IGARSS,* Milan, 2015.
[3] "AWS Ground Station," Web, URL: https://aws.amazon.com/ground-station/
[4] D. N. Amanoret al, "Intersatellite Communication System Based on Visible Light," in *IEEE AES* vol. 54, no. 6, Dec. 2018.
[5] Jim Carr, et. al, "An Innovative SpaceCube Application for Atmospheric Science," *2020 IEEE IGARSS,* 2020.
[6] J. L. Moigne, et al, "New Observing Strategy (NOS) for Future Earth Science Missions," *2019 IEEE IGARSS* Yokohama, Japan, 2019.
[7] "General Mission Analysis Tool", Web, URL: https://sourceforge.net/projects/gmat/
[8] "Systems Tool Kit", Web, URL: https://www.agi.com/products/stk
[9] Nag, Sreeja, et al, "Navigating the Deployment and Downlink Tradespace for Earth Imaging Constellations", *68th International Astronautical Congress (IAC),* Australia, 2017.
[10] A. G. Schmidt, et al, "Constellations in the Cloud: Virtualizing Remote Sensing Systems," *2019 IEEE IGARSS,* Japan, 2019.
[11] R. Linnabary, et al, "Open Source Software for Simulating Collaborative Networks of Autonomous Adaptive Sensors," *2019 IEEE IGARSS*, Yokohama, Japan, 2019
[12] Nag, Sreeja, et al, "D-SHIELD: Distributed Spacecraft with Heuristic Intelligence to Enable Logistical Decisions", *2020 IEEE IGARSS*, 2020.
[13] "Earth Observation Simulator Github repositories", Web, URL: https://github.com/EarthObservationSimulator
[14] "tkinter — Python interface to Tcl/Tk", Web, URL: https://docs.python.org/3/library/tkinter.html
[15] Met Office, *"Cartopy: A Cartographic Python Library with a Matplotlib Interface",* Exeter, Devon. Web, URL: http://scitools.org.uk/cartopy.
[16] "CesiumJs", Web, URL: https://cesium.com/cesiumjs/
[17] Wertz, James R., David F. Everett, and Jeffery J. Puschell. *"Space mission engineering: the new SMAD,"* Microcosm Press, 2011.
[18] V. Ravindra et al., "Fast Methods of Coverage Evaluation for Tradespace Analysis of Constellations," in *IEEE JSTARS,* vol. 13, 2020.
[19] David A. Vallado, *"Fundamental of Astrodynamics and Applications,"* Space Technology Series, Mc-Graw Hill,1997.
[20] V. Ravindra et al., "Instrument Data Metrics Evaluator for Tradespace Analysis of Earth Observing Constellations," *2020 IEEE Aerospace Conference*, Big Sky, MT, USA, 2020.
[21] R. May, et al, "MetPy: A Python Package for Meteorological Data", Unidata. Web, URL: https://www.unidata.ucar.edu/software/metpy/
[22] "Data Processing Levels", Web, URL: https://earthdata.nasa.gov/collaborate/open-data-services-and-software/data-information-policy/data-levels
[23] "Using Synthetic Data to Prepare for the NASA TROPICS Mission", Web, URL: https://nasasport.wordpress.com/2019/08/09/using-synthetic-data-to-prepare-for-the-nasa-tropics-mission/