SoftwareX 7 (2018) 37–40

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

AutoCNet: A Python library for sparse multi-image correspondence identification for planetary data

Jason Laura *, Kelvin Rodriguez, Adam C. Paquette, Evin Dunn

U.S. Geological Survey, Astrogeology Science Center, 2255 N. Gemini Drive, Flagstaff Arizona, 86001, United States

ARTICLE INFO

Article history:
Received 11 October 2017
Received in revised form 2 February 2018
Accepted 6 February 2018

Keywords:
Computer vision
Planetary science
Photogrammetry
Image mosaics

ABSTRACT

In this work we describe the **AutoCNet** library, written in Python, to support the application of computer vision techniques for *n*-image correspondence identification in remotely sensed planetary images and subsequent bundle adjustment. The library is designed to support exploratory data analysis, algorithm and processing pipeline development, and application at scale in High Performance Computing (HPC) environments for processing large data sets and generating foundational data products. We also present a brief case study illustrating high level usage for the Apollo 15 Metric camera.

Code metadata

Current code version	v0.2.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-17-00073
Legal Code License	UnLicense (Public Domain)
Code versioning system used	git
Software code languages, tools, and services used	python, CUDA
Compilation requirements, operating environments & dependencies	Linux, OSX & Python Scientific Computing Stack
If available Link to developer documentation/manual	For example: https://usgs-astrogeology.github.io/autocnet/
Support email for questions	jlaura@usgs.gov

1. Motivation and significance

The accurate spatial location of remotely sensed planetary image data is a critical precursor to many planetary science applications including topical science studies, landing site selection, and Digital Terrain Model (DTM) generation. To rigorously determine the location of an image accurate sensor position and attitude are required [1]. Traditionally, the identification of identical features or correspondences (e.g., crater rims, boulders) has been largely manual due to issues of viewing geometry [2], image homogeneity, sensor heterogeneity, and a lack of robust, broadly applicable Computer Vision (CV) algorithms. Automated methods for correspondence identification for bundle adjustment [3] are required due to exponential increases in data volumes, sensor spatial and spectral resolutions, and cross domain science studies [4]. Manual correspondence identification is not scalable to the problem domain.

We are aware of other efforts to provide automated *n*-image correspondence identification for planetary images or generic solutions that are applicable to planetary images. These include the U.S. Geological Survey Science Center's Integrated Software for Imagers and Spectrometers (ISIS3) [5] *findfeatures* application, the NASA Ames Stereo Pipeline [6] application for dense 3D model reconstruction, the Automatic Coregistration and Orthorectification (ACRO) Mars tools [7], and standard off the shelf image processing packages such as ESRI ArcGIS and Adobe Photoshop. The AutoCNet library differs from these applications in six ways. First, AutoCNet natively supports 32-bit images without rescaling to 8-bit as we do not depend upon OpenCV [8] for feature extraction. Second, AutoCNet focuses on dense feature extraction and sparse feature matching to support bundle adjustment and not dense 3D reconstruction. Third, AutoCNet supports feature extraction using tiling schemes,

* Corresponding author.
E-mail address: jlaura@usgs.gov (J. Laura).

https://doi.org/10.1016/j.softx.2018.02.001
2352-7110/

user definable downsampling, and a priori matching information (described below) to support planetary data sets of arbitrary size. Fourth, AutoCnet is designed as a functionally programmed library with high extensibility and not a set of high level applications or processing pipelines. Fifth, while AutoCNet has implemented both the coupled-decomposition [9] and ring matching [7] approaches from the ACRO Mars project, we have focuses on the creation of a scalable and distributable graph representation to support processing large volumes of data. Finally, other software packages, such as ArcGIS Pro or Adobe Photohop support the creation of mosaic data sets that apply Computer Vision (CV) techniques to warp (as opposed to applying rigorous transformations) images into aesthetically pleasing mosaics. The resultant products are not photogrammetrically controlled resulting in unknown errors that can make these products unusable for some science applications.

We have developed the **AutoCNet** library with the goals of (1) providing a computational research environment for the development and application of CV and photogrammetric techniques for planetary images, and (2) creating pipelines for bulk data processing across heterogeneous sensors. The computational environments for these goals differ. For use in developing new algorithms or beginning work with new data sets, the Jupyter notebook [10] interactive computing environment is ideally suited for interactive computation and visualization. For bulk data processing, the AutoCNet data structures and functional method application are well suited to High Performance Computing (HPC) environment. The AutoCNet library is currently being used to process data collected by the Mars Odyssey Thermal Emission Imaging System (THEMIS) [11] to support subpixel accurate pairwise image registration for automated change detection [12] using the same processing work flow as described below in Section 3. Automated methods to support correspondence identification and subsequent bundle adjustment are crucial computational tools to support the development of foundational data products [13] that fulfill an infrastructural role across many planetary science applications.

2. Software description

The AutoCNet library is designed to support *n*-image correspondence identification using CV approaches for remotely sensed planetary data. Written in Python and CUDA, AutoCNet is intentionally designed as a library of functionality with a high level API for use by research scientists and a reference implementation to demonstrate chaining the high level API into a processing pipeline.

2.1. Software architecture

The AutoCNet project is modularized into three components focusing on (1) general book keeping of overlapping images and the associated correspondences, (2) functional analysis algorithms, and (3) general utility and visualization functionality.

2.1.1. autocnet.graph and autocnet.control

Central to the autocnet library is the **CandidateGraph** that represents images as nodes and potential overlap as edges. We utilize an undirected NetworkX [14] graph as the foundation for the CandidateGraph thereby gaining access to all of the graph analysis algorithms contained within NetworkX. The **Node** and **Edge** objects are custom developed to extend NetworkX functionality. Each **Node** represents a remotely sensed image with associated spatial information such as image footprint and geotransformation, as well as convenience functions to support interest point extraction. Each **Edge** represents the overlap between two images and stores pairwise correspondence information. A key design decision was to maintain all correspondences throughout the matching and outlier detection process. Therefore, AutoCNet utilizes a Pandas [15] DataFrame to store correspondence information and a like indexed boolean DataFrame of masks.

While the **CandidateGraph** stores pairwise image overlap information, the **control** module supports aggregation of pairwise information into neighborhood information. The identification and storage of correspondences identified in a high number of images is critical for the photogrammetric bundle adjustment process. Therefore, we utilize a Pandas DataFrame, stored within a **Control** object to track individual pairwise correspondences or measures and aggregate these into point objects.

2.1.2. Algorithms

The **camera**, **matcher**, and **transformation** modules contains the bulk of the algorithmic functionality. These modules are designed to contain generic functions that are mapped to the **CandidateGraph**, **Node**, or **Edge** objects as appropriate. This is an intentional, functional design decision to support scalability and extensibility. The **camera** module contains standard, idealized camera representations and multi-image triangulation functionality that is commonly used in multi-view geometry applications for estimation of image relationships [16]. The **matcher** module contains both CPU [8,17] and GPU (CUDA) [18] implementations for feature extraction and matching. Finally, the **transformation** module encapsulates algorithms for pairwise and multi-image relationship computation in the form of homography, fundamental, and tri-focal tensor matrices [8,16].

2.1.3. Support modules

Finally, the library provides **utils**, **io**, **vis**, and **cg** modules that house utility functions, specialized serialization routines and data Input/Output (IO), visualization convenience functions, and computational geometry capabilities, respectively. For serialization we use JSON metadata and the built-in NumPy [19] array input/output capability and the externally managed Planetary IO module provides input data reads in all Geospatial Data Abstraction Library (GDAL) [20] supported formats, (e.g., ISIS3 Cube, GeoTiff, JPEG2000). The MatPlotLib [21] library provides the foundation for visualization. Finally the **cg** module contains general computational geometry algorithms such as a point-in-polygon check for determining if an identified correspondence could exist within the overlap of two images.

2.2. Extensibility

As noted above, AutoCNet utilizes a functional design by which algorithms with generic interfaces can be applied to the general graph structure. In practice this means that custom algorithms can be developed, tested, and used in pipeline processing as long as they conform to the API. For example, the feature extraction functionality requires an input array of observed data and any appropriate arguments and keyword arguments. The underlying extraction algorithm can take any form. For an example of this see (http://bit.ly/2ENTIfw) where a custom tiled extractor is patched to the CandidateGraph to support large Apollo 15 Panoramic images. This same approach is taken for matching, outlier detection algorithms, and keypoint masking algorithms.

3. Illustrative example of software functionality

To demonstrate the functionality of the AutoCNet library, we process eight Apollo 15 Mapping (Metric) Camera images [22] over the Hadley-Apennine region of the Moon. The spacecraft mounted, metric framing camera images were collected during orbits 16, 22, and 27 with approximately 80% overlap between images and variable lighting conditions [23]. Broadly, processing this subset of the Apollo 15 data set requires eight steps: (1) identification of