ASAP-Stereo, Ames Stereo Automated Pipeline. A. M. Annex[1], K. W. Lewis[1]. [1]Department of Earth and Planetary Sciences, Johns Hopkins University, Baltimore, MD 21218, USA (annex@jhu.edu).

Introduction: ASAP-Stereo (hereafter ASAP) is an open-source (BSD-3) high-level workflow wrapper for the NASA Ames Stereo Pipeline (ASP) written in Python [1]. ASAP currently implements an enhanced version of the University of Chicago "ASP Scripts" workflow to make co-registered CTX and HiRISE Digital Elevation Models (DEMs) [2]. ASAP provides a Command Line Interface (CLI) and Python API with templated workflows using Jupyter Notebooks, which embeds logs, metadata, and preview images in a shareable format to enable reproducible processing of stereo products with ASP [3]. ASAP provides a framework and utility "glue" for implementing workflows using ASP. ASAP is available at https://github.com/AndrewAnnex/asap_stereo/, and the documentation is available at https://asap-stereo.readthedocs.io.

Motivation: The initial inspiration for ASAP was to automate and simplify the existing University of Chicago "ASP Scripts" methodology to produce paired and co-registered CTX and HiRISE DEMs, hereafter "ASP Scripts" [2]. For future reference: a step is an individual unit of work within a collection of steps called a workflow, and multiple collections of steps (meta-steps) can define a workflow when run in series. Although functional, the "ASP Scripts" workflows could be further simplified to require fewer meta-steps to simplify user interaction. Some steps within the "ASP Scripts" workflows are slightly error-prone, so access to individual steps was also desirable for efficient trial-and-error [2]. Ultimately, the "ASP Scripts" workflow was broken up into dozens of separate steps to reimplement enhanced versions of the "ASP Scripts" workflows. These enhancements will be described in a subsequent section.

Intent. ASAP is not a replacement for ASP and deep familiarity with ASP capabilities and documentation is still a requirement. ASAP will be most helpful to users with moderate knowledge of ASP and users working with the "ASP Scripts". Novice ASP users are best served following existing tutorials and documentation to use ASAP properly.

Workflow: Making a CTX and HiRISE DEM requires four meta-steps within the ASAP workflow: **1** generate the CTX DEM, **2** align the CTX DEM to MOLA, **3** develop the HiRISE DEM, and **4** align the HiRISE DEM to the aligned CTX DEM from step **2**. Note that each of the first DEM meta-steps can run in parallel and that it is possible to align HiRISE to MOLA without a CTX DEM. Further enhancements to ASAP could simplify this further to a single meta-step. The alignment steps require a maximum disparity parameter the user estimates that can only be determined after the first DEMs are constructed [2]. However, it can be estimated using the ASP geodiff program or using feature matching techniques partially enabled for the HiRISE workflow in ASAP.

Architecture: ASAP is architected as a conventional Python package that provides a CLI to the user. A single Python file contains the majority of the code. Steps for workflows are implemented as functions in four core classes: ***CommonSteps***, ***CTX***, ***HiRISE***, and ***ASAP***. The first class includes generally useful utilities or abstracted from the data-specific workflow classes ***CTX*** and ***HiRISE*** to reduce repeated code. The ***ASAP*** class provides access to workflows that call many individual processing steps as a higher-level interface to make DEMs with a single and simple command. Steps are implemented as functions on the classes and are current named by step order rather than functionality intent as certain steps are re-run in workflows with different parameters.

Installation. ASAP is intended to be installed using pip within a dedicated conda environment with ASP and ISIS installed. ASAP provides an example conda environment YAML file in the repository and documentation tutorials to create a working installation.

Core Dependencies: ASAP-Stereo is built on a small number of foundational dependencies that enable the project's flexibility. ***Moody***, also written by the first author, provides programmatic and CLI access to a subset of the PDS Geosciences Node REST API and Granular Data Search services to download CTX/HiRISE/MOLA data and metadata based on queries and known or partially known image ids [4]. ***Python-fire*** automatically generates CLIs for ASAP, enabling a single codebase to implement both the API and CLI. The ***sh*** python library provides an automatic subprocess interface between ASAP in Python and programs on the path, such as ASP, ISIS, and other CLI programs like GDAL utilities. ASAP uses ***Jupyter Notebooks*** to implement parameterized workflows that use ASAP using the ***Papermill*** project to execute them.

Logging: ASAP provides rich logging in multiple log files for individual stages or steps and an accumulated total log containing all executed subprocess commands and arguments as well as start and end times, enabling more accessible debugging of individual processing steps.

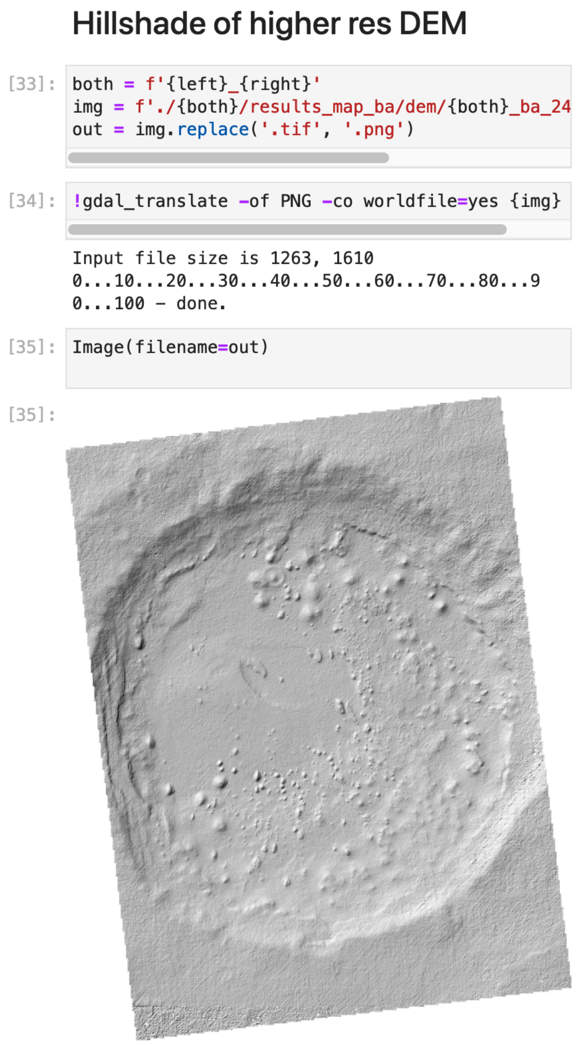
Development: Testing ASAP using Continuous Integration (CI) services is complex due to the need for local kernels and base data in ISIS and the computational requirements of ASP. However, after the SPICE and base data were curated, a CTX processing workflow was implemented to test ASAP within a GitHub Action Workflow.

Enhancements: ASAP provides several improvements to the "ASP Scripts" workflow that simplifies usage and improves stereo product generation. ASAP (using Moody) can download the image files from the PDS and MOLA PEDR data for the user. ASAP augments the "hillshade-align" feature of ASP pc-align by performing the hill shade using GDAL CLI to determine an initial transform for registering HiRISE point clouds (PCs) to CTX PCs. ASAP uses image metadata information to determine conservative ground sample distances (GSDs) for stereo products (four times the worst image GSD, rounded up to nearest integer) and the optimal number of threads and processes to use for parallelizable steps. ASAP also runs bundle adjustment for both CTX and HiRISE workflows. ASAP includes a stereo pair quality assessment tool adapted from https://github.com/rbeyer/scriptorium following criteria established by Becker et al. 2015 [5].

Reproducibility: One of the key challenges to using ASP is the number of steps and parameters that frustrate reproducing a stereo product. Workflows with ASP take many command line steps, requiring dozens of stereo parameters to obtain optimal or higher quality results. ASAP creates reproducible workflows using Jupyter Notebook templates that run ordered processing steps from ASAP in notebook cells to implement the workflows. Notebooks capture logs, visualizations, and stereo configuration file parameters into a single shareable file [3]. Jupyter Notebooks were selected because of the broad popularity of Jupyter and the ability to mix Python, Bash, and markdown in a single file in a literate computing paradigm [6]. Notebook workflows in ASAP contain additional diagnostic visualizations, including good pixel map previews and hill-shaded previews (Fig 1) to enable quick visual inspection of processing results over remote connections. Notebooks also allow rapid iteration of individual steps with updated parameters or whole new workflow components implemented by the user. Output notebook files from ASAP can be shared as supplementary materials or data files for publications or shared with other users and re-executed to reproduce results.

Future Work: Future directions for ASAP include additional workflows for LRO NAC and Rover (MSL/M2020) stereo processing. Other features such as automatic ground control point creation for improved bundle adjustment could also be integrated into ASAP. User-contributed enhancements and workflows are also welcome.

Fig 1: Example DEM hill shade preview of a CTX DEM of Sera Crater as displayed within a Jupyter Notebook produced by ASAP, illustrating the mix of Python and Bash to show the progress of an ASAP processing job.



Hillshade of higher res DEM

```
[33]: both = f'{left}_{right}'
      img = f'./{both}/results_map_ba/dem/{both}_ba_24
      out = img.replace('.tif', '.png')

[34]: !gdal_translate -of PNG -co worldfile=yes {img}

      Input file size is 1263, 1610
      0...10...20...30...40...50...60...70...80...9
      0...100 - done.

[35]: Image(filename=out)

[35]:
```

Acknowledgments: CTX data used to produce Fig 1. was obtained from the PDS Geosciences node http://pds-geosciences.wustl.edu/. This work was funded by Mars Data Analysis Program (MDAP) NASA grant (80NSSC17K0672). We thank D. P. Mayer for advice while developing ASAP.

References: [1] Beyer, R. A., Alexandrov, O. and McMichael, S. (2018) *ESS*, 10.1029/2018EA000409. [2] Mayer, D. P. and Kite, E. S. (2016) LPSC 47, Abstract #1241. [3] Kluyver, T. et al. (2016) *ELPUB*. [4] Wang, J., et al. (2017) 3rd *PDW* #1986. [5] Becker et al. (2015) *LPSC* 46, Abstract #2703. [6] Kery, M. B., et al. (2018) *CHI,* 10.1145/3173574.3173748.