Automatic Time Series Generation Using HyP3 RTC Products

ASF Engineering

Abstract

This document is a guide to users of the ASF RTC stack processing software. It includes installation and operation instructions as well as examples of code usage.

Installing the Software

Installing Prerequisite Packages

In order to run the ASF RTC stack processing software, the following prerequisite packages must first be installed: numpy, gdal 2.0, scipy, pip, boto3, and imagemagick. On a Ubuntu 16 base installation, these can be installed as follows:

```
$  sudo apt-get install python-numpy
$  sudo add-apt-repository -y ppa:ubuntugis/ubuntugis-unstable
$  sudo apt update
$  sudo apt upgrade
$  sudo apt install gdal-bin python-gdal
$  sudo apt-get install python-scipy
$  sudo apt-get install python-pip
$  sudo apt-get install python-boto3
$  sudo apt install imagemagick-6.q16
```

Installing ASF Software

Next is the installation of the ASF stacking software and the HyP3 API. This can be performed by executing these commands:

```
$  cd /usr/local
$  sudo git clone https://github.com/asfadmin/hyp3-lib.git
$  sudo git clone https://github.com/asfadmin/hyp3-giant.git
$  sudo pip install asf-hyp3
```

Setting Environment Variables

Set the necessary environment variables by adding these lines to your ~/.bashrc file:

```
#HYP3
export PYTHONPATH=$PYTHONPATH:/usr/local/hyp3-lib/src
export PATH=/usr/local/hyp3-giant/src:$PATH
export PATH=/usr/local/hyp3-lib/src:$PATH
```

Accessing ASF HyP3 API

In order to access your HyP3 subscriptions you must have a NASA Earthdata account. If you do not have an Earthdata Login, you can create one at https://urs.earthdata.nasa.gov/users/new.

Once you have an Earthdata login, in order to use the HyP3 API, you must create a .netrc file in your home directory. The file has this form:

```
machine urs.earthdata.nasa.gov
    login username
    password password
```

Where, *username* and *password* are your Earthdata login information.

Finally, you must approve a new application through Earthdata. The application “Alaska Satellite Facility HyP3 API” provides access to your HyP3 subscriptions via the ASF HyP3 API:

1) Login to https://urs.earthdata.nasa.gov/home
2) Go to "Applications" → “Authorized Apps”
3) Go to "Approve More Applications" at the bottom of the page
4) Approve the application Alaska Satellite Facility HyP3 API

Having gone through these steps, you’ll be able to automatically download HyP3 subscriptions using the stack processing software.

Operating Instructions

In order to create time series of specific regions of interest using ASF HyP3 RTC products, ASF provides the python command line tool procS1StackRTC.py.

```
usage: procS1StackRTC.py [-h] [-b BLACK] [-d upper lower] [-f] [-g] [-i]
                         [-k {a,d}] [-l] [-m MAGNIFY] [-o OUTFILE] [-p PATH]
                         [-r RES] [-t {dB,sigma-byte,dB-byte,amp,power}] [-z]
                         [-c ULE ULN LRE LRN | -s shapefile | -v]
                         [-a AWS | -n NAME]
                         [infile [infile ...]]
```

Input Files

The tool operates from a HyP3 subscription name, a directory of ASF HyP3 zip files, a directory of directories of already unzipped ASF HyP3 zip files, an AWS bucket, or a list of geotiffs given on the command line. That is, if input files are given on the command line, they are assumed to be custom geotiffs. If no input files are given on the command line, then the code will

1. Download zip files from ASF and unzip them (*--name* option), or
2. Unzip user specified files (*--zip* option) from either the current directory or that specified by the --path option, or
3. Look for directories of unzipped HyP3 zip files (*default*) in either the current directory or the directory specified by the --path option, or
4. Use cloud optimized geotiff files stored in the AWS bucket specified by the --aws option

Preprocessing Options

Once the initial collection of input files is established, they may be grouped by time (*--group*) or else processed en masse. The list may also be culled by orbital direction, either ascending or descending (*--keep*). Next, the code will optionally subset, filter (*--filter*), and resample (*--res*) the stack. Prior to subsetting, all files are reprojected to a common UTM zone (this does not work with AWS - see AWS note below). If the clipping option is selected (*--clip*), then the file with the most overlap with the user supplied bounding box will be used as the base image and all other images will be reprojected to match that one. During clipping, images will be examined for zero content and discarded if the content falls less than a given threshold (*--black*) after clipping is performed. Alternately, data can be cut to a shapefile (*--shape*), or the stack overlap can be calculated and used to subset the images (*--overlap*). Note that the three different clipping options are mutually exclusive - only one my be selected at a time.

Notes Regarding Using AWS Buckets

When AWS buckets are used, rather than downloading the entire file and then cutting out the area of interest, images are clipped directly from the bucket. This can be a valuable time saver when dealing with geotiffs already stored in AWS. However, there are a few caveats. First, the geotiffs must all be cloud optimized geotiffs. HyP3 subscription outputs do not work directly. Secondly, because the files are assumed to be custom geotiffs, no orbit direction information is included. Thus, culling by orbit direction is not supported. Finally, because the files are pulled from AWS as they are cut, they must all be in UTM coordinates from the same UTM projection zone prior to running the stack processing code.

Post processing Options

After the data pre-preprocessing steps are accomplished, the results are scaled to dB, scaled to byte values using a set dB range (*--dBscale*), and converted to png files. If the input files are from the ASF Hyp3 website, then the png files will be annotated (*--magnify*) with the date of the input file. The resulting stack of png files are then converted into an animated GIF file. In addition, all of the preprocessed geotiff frames will be converted to the requested output type (*--type*).

Output Files

The outputs from the tool are placed in a directory named PRODUCT_{*outfile*}, where *outfile* is specified on the command line via the *--outfile* option, or simply PRODUCT if no outfile is specified. The product directory will contain the output frames, the animated GIF, and the log file. If a HyP3 subscription name was passed to the code there will also be a directory named “hyp3-products” containing the zip files downloaded from the HyP3 API and another named “hyp3-products-unzipped” containing the contents of the zip files. At run completion the working directory, TEMP, will be deleted unless the leave flag is set (*--leave*).

List of Command Line Options

Option		Type	Description
--help	-h	boolean	Gives long list of help
--inamp	-i	boolean	Specifies that input files are amplitude and not power. ASF HyP3 RTC files are power.
--black	-b	float	Fraction of black required to remove an image from the stack. After images are clipped to the same area of interest (see --*clip*), the total black in the image is counted and divided by the total pixels to give the fraction of black for each clipped image. If this fraction is less than the black threshold, the image is removed from the stack. If you are missing partial images from your final animation, you may decrease this number. If you have too many partial images, you may increase this number. The default is 0.4.
--dBscale	-d	float x 2	Specifies the upper and lower bounds for dB to byte scaling. The default is 0 to -40 for VV polarizations images.
--filter	-f	boolean	Apply enhanced lee filter to image stack
--keep	-k	char	Flag to keep either only ascending (a) or only descending (d) images. The default is to keep all images in the stack regardless of orbit directions
--leave	-l	boolean	Leave the TEMP directory with all intermediate files in place. The default action is to delete the TEMP directory at the end of each run.
--magnify	-m	integer	Magnify (set) the annotation font size. For large animations, increase the font size to maintain the visibility of the date annotations. The default is 24 point.
--outfile	-o	string	Specify the output GIF animation name. The default is animation.gif
--path	-p	string	Specifies the path to the input zip files, directories, or custom geotiffs. Can be a relative or absolute path. The default is to assume that input files or directories are in the current working directory.
--res	-r	float	Set the output resolution of the frames. The default is to create output files at the resolution as the input files. Use this option if your custom geotiffs have different pixel spacings in order to create equally spaced output frames. Also, useful for downsizing very large animations.
--type	-t	string	One of *dB, sigma-byte, dB-byte, amp,* or *power*. This controls the output format of the frames that are generated. Since ASF RTC images are power scale, choosing the option *power* leaves the data as is. Choosing the option *amp* creates amplitude output images by taking the square root of the input data. The option *dB* converts the input data to dB scale by applying 10*log() function to the input data. *dB-byte* scales the dB images to byte using the upper and lower bounds specified by the option *--dBscale*.

			Default is -40 to 0. Finally, *sigma-byte* creates an amplitude image that is scaled to byte using a 2-sigma range. Note that regardless of the option chosen for type, the output animation will always use dB-byte since it provides a consistent scaling across all imagery. Thus, the default output type is *dB-byte*.
--zip	-z	boolean	Start from HyP3 zip files instead of directories. Since it is assumed that multiple animations will be created per image stack, it makes sense to only unzip the Hyp3 zip files once. Thus, the default is to assume that the user has already unzipped these files. However, if the --zip option is chosen, then the code will start by unzipping Hyp3 zip files prior to other processing.
Mutually Exclusive Subsetting Options			
--clip	-c	float x 4	Specifies the upper left and lower right corner coordinate for clipping the output frames and animation. One of three options for subsetting the data.
--shape	-s	string	Specifies the shapefile to be used to create the output image frames and animation. The second of the subsetting options.
--overlap	-v	boolean	The final subsetting option specifies that outputs will be clipped using the common overlap of all input scenes.
Mutually Exclusive Input Options			
--aws	-a	string	Name of the AWS S3 bucket to use for the input files. If this option is selected, then all geotiff files in the given bucket will be used as input to the stack processing procedure.
--name	-n	string	Name of the HyP3 subscription to use for the input files. If this option is selected, then all files in the HyP3 subscription named will be downloaded through the HyP3 API.

Test Case #1 -- Delta Junction, Alaska

Part A -- Setting up the HyP3 subscription

Follow instructions found at hyp3.asf.alaska.edu

Part B -- Downloading Files and Creating a First Time Series Animation

Once the subscription jobs have run, time series animations are only a single command away:

```
$ procS1StackRTC.py --group --name "Delta Gamma Time Series" -o dg_ts -r 150
```

This command will first download the HyP3 zip files for the subscription named “Delta Gamma Time Series” and store them in a directory named “hyp3-products”. Next, it will group the zip files by time, sorting them into separate directories (with symbolic links so no data movement takes place). Then, a time series will be created for each time class found (as long as the time class contains at least three images). Checking the log file for this run, we find that three separate animations will be created:

```
Class 0 : 030327 contains
    S1B_IW_GRDH_1SDV_20180225T030327_20180225T030352_009774_011A82_2FB0-RESORB-30m-rtc-gamma
    S1B_IW_GRDH_1SDV_20180108T030328_20180108T030353_009074_010389_0EC9-POEORB-30m-rtc-gamma
    S1B_IW_GRDH_1SDV_20180213T030327_20180213T030352_009599_0114C3_F364-POEORB-30m-rtc-gamma
    S1B_IW_GRDH_1SDV_20180120T030328_20180120T030353_009249_01093F_0D92-POEORB-30m-rtc-gamma
Class 1 : 031947 contains
    S1B_IW_GRDH_1SDV_20180118T031947_20180118T032012_009220_01084D_97C9-POEORB-30m-rtc-gamma
    S1B_IW_GRDH_1SDV_20180211T031946_20180211T032011_009570_0113CE_4A20-POEORB-30m-rtc-gamma
    S1B_IW_GRDH_1SDV_20180223T031946_20180223T032011_009745_011990_A655-RESORB-30m-rtc-gamma
    S1B_IW_GRDH_1SDV_20180307T031946_20180307T032011_009920_011F6A_0441-RESORB-30m-rtc-gamma
    S1B_IW_GRDH_1SDV_20180106T031947_20180106T032012_009045_01029C_5F58-POEORB-30m-rtc-gamma
Class 2 : 161201 contains
    S1A_IW_GRDH_1SDV_20180128T161201_20180128T161226_020357_022C68_F5F3-POEORB-30m-rtc-gamma
    S1A_IW_GRDH_1SDV_20180116T161202_20180116T161227_020182_0226E1_D5EF-POEORB-30m-rtc-gamma
    S1A_IW_GRDH_1SDV_20180209T161201_20180209T161226_020532_023203_D3B6-POEORB-30m-rtc-gamma
    S1A_IW_GRDH_1SDV_20180305T161201_20180305T161226_020882_023D1D_85DA-RESORB-30m-rtc-gamma
    S1A_IW_GRDH_1SDV_20180221T161201_20180221T161226_020707_023798_E53D-RESORB-30m-rtc-gamma
    S1A_IW_GRDH_1SDV_20180104T161202_20180104T161227_020007_022150_C2FA-POEORB-30m-rtc-gamma
```

Each animation will contain a stack cut to common overlap, which general is nearly the entire image when using group sorting. During creation of the stack, clipping will take place first. Images will be resampled to 150 meters. When the run is done, the directory where the run occurred contains the following:

```
dg_ts_run_stats.txt
hyp3-products
hyp3-products-unzipped
PRODUCT_dg_ts_030327
PRODUCT_dg_ts_031947
PRODUCT_dg_ts_161201
```

Here, *dg_ts_run_stat.txt* is the log file for the run. It contains informational messages as well as warnings and errors. It provides useful information about the classes that were created, files linked, and processing

steps performed. The directory *hyp3-products* contains the original zip files that were download for the subscription “Delta Gamma Time Series”. While, the directory named *hyp3-products-unzipped* contains the unzipped versions of these same files. Finally, the PRODUCT directories contain the individual animations as well as the animation frames stored as geotiffs with the values of dB-byte. In this case:

```
PRODUCT_dg_ts_030327:
dg_ts_030327.gif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180108T030328_20180108T030353_009074_010389_0EC9_cut_150m_dB
-40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180120T030328_20180120T030353_009249_01093F_0D92_cut_150m_dB
-40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180213T030327_20180213T030352_009599_0114C3_F364_cut_150m_dB
-40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180225T030327_20180225T030352_009774_011A82_2FB0_cut_150m_dB
-40_0.tif

PRODUCT_dg_ts_031947:
dg_ts_031947.gif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180106T031947_20180106T032012_009045_01029C_5F58_cut_150m_dB
-40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180118T031947_20180118T032012_009220_01084D_97C9_cut_150m_dB
-40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180211T031946_20180211T032011_009570_0113CE_4A20_cut_150m_dB
-40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180223T031946_20180223T032011_009745_011990_A655_cut_150m_dB
-40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180307T031946_20180307T032011_009920_011F6A_0441_cut_150m_dB
-40_0.tif

PRODUCT_dg_ts_161201:
dg_ts_161201.gif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180104T161202_20180104T161227_020007_022150_C2FA_cut_150m_dB
-40_0.tif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180116T161202_20180116T161227_020182_0226E1_D5EF_cut_150m_dB
-40_0.tif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180128T161201_20180128T161226_020357_022C68_F5F3_cut_150m_dB
-40_0.tif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180209T161201_20180209T161226_020532_023203_D3B6_cut_150m_dB
-40_0.tif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180221T161201_20180221T161226_020707_023798_E53D_cut_150m_dB
-40_0.tif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180305T161201_20180305T161226_020882_023D1D_85DA_cut_150m_dB
-40_0.tif
```

Part C -- Clipping to an Area of Interest Bounding Box

Suppose that we now want to look at a specific area of interest - that being the city of Delta Junction, AK. We can use the following command to take advantage of the already downloaded and unzipped HyP3 files from part B:

```
$ procS1StackRTC.py --path hyp3-products-unzipped -o dg_ts_zoom1 -c 554535
7127205 578535 7103205
```

In this case, we have supplied the path *hyp3-products-unzipped*, the directory that was created in the first example. So, we have all of the same input files as the first run. This time however, we've selected the clipping option, pulling out a box from an upper left corner of (554535, 7127205) to a lower right of corner of (578535, 7103205). Instead of segregating the files into groups as before, all files will be used in this run.

The first step is to intersect each of the files with the bounding box to find the granule that best fits the requested AOI. To do this, geometric objects are formed from the requested bounding box and the corner coordinates of each of the input files. The intersection of the bounding box with each image is then calculated and the first image with the best match will be selected as the base image for the next operation, reprojection, wherein all files are projected to the same UTM zone as the base image.

Once files are reprojected (if needed), they will all be clipped to the bounding box. During this process, any images that fall outside of the bonding box will be culled from the list. For the Delta Junction subscription, four files have less than the default overlap of 40% (in fact, they have no overlap of real pixel values). The are all discarded before proceeding to animation creation.

For this run, only a single animation is created. After this run, the working directory looks like this:

```
dg_ts_run_stats.txt
dg_ts_zoom1_run_stats.txt
hyp3-products
hyp3-products-unzipped
PRODUCT_dg_ts_030328
PRODUCT_dg_ts_031946
PRODUCT_dg_ts_161201
PRODUCT_dg_ts_zoom1
```

The new time series animated GIF is in the directory PRODUCT_dg_ts_zoom1, which contains the gif as well as 11 geotiff frames:

```
dg_ts_zoom1.gif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180104T161202_20180104T161227_020007_022150_C2FA_clipped_dB-
40_0.tif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180116T161202_20180116T161227_020182_0226E1_D5EF_clipped_dB-
40_0.tif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180128T161201_20180128T161226_020357_022C68_F5F3_clipped_dB-
40_0.tif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180209T161201_20180209T161226_020532_023203_D3B6_clipped_dB-
40_0.tif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180221T161201_20180221T161226_020707_023798_E53D_clipped_dB-
40_0.tif
s1a-iw-rtcm-vv-S1A_IW_GRDH_1SDV_20180305T161201_20180305T161226_020882_023D1D_85DA_clipped_dB-
40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180106T031947_20180106T032012_009045_01029C_5F58_clipped_dB-
40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180118T031947_20180118T032012_009220_01084D_97C9_clipped_dB-
40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180211T031946_20180211T032011_009570_0113CE_4A20_clipped_dB-
40_0.tif
s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180223T031946_20180223T032011_009745_011990_A655_clipped_dB-
40_0.tif
```

s1b-iw-rtcm-vv-S1B_IW_GRDH_1SDV_20180307T031946_20180307T032011_009920_011F6A_0441_clipped_dB-40_0.tif

Test Case #2 -- Denali, Alaska

Part A -- Unzipping Files and Creating Glacier Animation

Starting from a directory of HyP3 zip files, this test case will create several different animations of glacial motion over the course of a year.

```
$ procS1StackRTC.py --path hyp3-products --zip -o dne_5 -c 539955.0 6954015.0
570675.0 6923295.0
```

This command will go through the 55 zip files found in hyp3-products, unzip them into a directory named hyp3-products-unzipped, intersect all files with the bounding box, and create an animation named dne_5.gif.

Part B -- Choosing Files Based On Orbital Direction

In examining the resulting animation from part A, one can readily see the differences between ascending and descending passes. Although the RTC process corrects for look angle as much as possible, the mountains in the animation still appear to jump around. In fact, the images are well co-registered, the apparent motion between images is caused by combining both ascending and descending tracks into a single animation. To avoid this apparent “jumping” problem, one can choose to keep only ascending or only descending passes from a stack of input files:

```
$ procS1StackRTC.py --path hyp3-products -o dne_5_asc -c 539955.0 6954015.0
570675.0 6923295.0 -k a
```

The above command creates an ascending animation, while the one below creates a descending animation:

```
$ procS1StackRTC.py --path hyp3-products -o dne_5_desc -c 539955.0 6954015.0
570675.0 6923295.0 -k d
```

Notice how much smoother the ascending and descending animations are compared to the original with all of the data.