SCADI - Satellite Cross-correlation for Accurate Determination of Ice velocity

Authors: Andrew Sole, Ben Davison

Institution: University of Sheffield

Table of Contents

With great power comes great responsibility

I have done my best to include several checks throughout SCADI to prevent catastrophic data loss or irresponsible use of computing resources. BUT, as a user of SCADI and member of grio1, you have a responsibility to make sure you don't delete everyone's data. Whatever pairs you choose, SCADI will start them from scratch. So if you submit all pairs over Greenland, it will process all of the pairs over Greenland - even the pairs that we've already done. So, read the guidelines below very carefully and prepare your jobs with care. If in doubt, don't do it and ask Ben or Andrew!

Prerequisites

Before using SCADI, you should make sure that you are familiar with the Sheffield HPC clusters. We recommend reading the Sheffield HPC documentation here: Using the HPC Systems — Sheffield HPC Documentation.

- You should be able to connect to them as described here: Connecting to a cluster using SSH — Sheffield HPC Documentation.
- You should be familiar with the Stanage filestores, described here: Filestores — Sheffield HPC Documentation.
- You should be able to navigate around the Stanage filestores using your choice of SSH Client such as MobaXterm or WinSCP.
- You should also be able to open a bash terminal, for which you will probably needt PuTTY: Download PuTTY - a free SSH and telnet client for Windows.
- You should be familiar with running MATLAB on Stanage: MATLAB — Sheffield HPC Documentation

You also need an account in Andrew's Shared storage space **grio1**.

Once you have an account and are ready to run SCADI, you need to do the following things in a Bash terminal, after logging into Stanage:

1. You will always run SCADI from your home directory. I recommend making a folder in your home directory. i.e. `mkdir /users/$USER/SCADI`. If you're going to have multiple instances of SCADI (e.g. processing different areas), then you should use folders to keep them organised.

2. Make a test folder to get started. i.e. `mkdir /users/$USER/SCADI/test_run`

3. Copy the SCADI template to your test folder. `cp /mnt/autofs/grio1/Shared/SCADI/code/SCADI.m /users/$USER/SCADI/test_run`

4. Setup a scrontab to keep your home space tidy.

a. Copy the housekeeping script to your home SCADI directory: `cp /mnt/autofs/grio1/Shared/SCADI/code/housekeeping.sh /users/$USER/SCADI`

b. Start an editing session in your scrontab: `scrontab -e`

`c.` <make note of which login node you are on - scrontabs are specific to each>

d. <enter "i" (without quotations) to enter insert mode>

e. <copy and paste the following lines into your scrontab>:

```
# Tidy up home space

58 23 * * * cd /users/$USER/SCADI/code && /users/$USER/SCADI/code/housekeeping.sh > /
users/$USER/SCADI/code/housekeeping.out 2>&1
```

f. <press "Esc" to exit insert mode>

g. <enter ":wq" (without quotations) to save and quit vim>

h. <check your scrontab by typing "scrontab -l">

# It's probably worth making a copy of your scrontab in a text file on your pc, just so you have a record of what's going on

5. Add login2 to your list of known hosts:

```
echo "login2 ecdsa-sha2-
nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBFwlziLMaI58xdLxlll+3LR/
xXDQt/LpExBCHgj88rEuiZuE617fqPqihJRDRX2IZpZphOSlO2To4y/oeopqQC" >> ~/.ssh/known_hosts
```

Overview

SCADI is a set of functions to automate the identification and processing of image pairs for ice velocity estimation. It was developed initially for Antarctica and Greenland with Sentinel-1 imagery and the aim is to add more sensors/image types and regions over time. It is designed to allow 'operational' processing, in which new image pairs over a region are processed routinely, but it should also be flexible enough to permit ad-hoc processing of any area and any time period.

All users can have their own versions of SCADI.m (the master script, described below) - you can even have multiple versions of this and name them whatever you like - and all users can run their own versions independently of everyone else. All of the processing is done in Ben's fastdata/parscratch directory and all of the output will typically be stored in the grio1 Shared space. Monitoring of processing progress and file transfer between parscratch and grio1 will happen automatically.

Definitions

Region: I have pre-defined a set of regions, based on the ITS-LIVE regions. At the moment, these include only Greenland and Antarctica. The aim is to expand these to also include Svalbard, Patagonia, Alaska, the Alps and High Mountain Asia (and maybe others!) Although I have defined these fixed regions, you can process smaller areas within them (but the output will be saved within the parent region).

Type: This refers to the satellite mission or image type. At the moment, this only include Sentinel-1. Eventually, I will add Sentinel-2, the various Landsats and hopefully UAV. Each image type will be dealt with separately and will likely have different routines, but the output and metadata will be stored in a similar architecture across image types.

Directory structure overview

The following directory structure exists in grio1 and will be mirrored where necessary in Ben's fastdata/ parscratch directory. Any new directories will be created by SCADI at runtime. The directory structure looks like this. You should familiarise yourself with this structure, so that you can find the velocity output after doing any processing:

from /shared/grio1/Shared/SCADI

SCADI.m =================== this is a template file. Do not modify. Copy it to your own directory and modify it there.

SCADI.sh =================== this is a template file. Do not modify. Copy it to your own directory and modify it there.

- code

-------- PIVsuite_v.0.83 ========= All the code for the feature tracking

-------- utils ========= A set of helper functions used by everything

-------- Sentinel1

----------------- GMTSAR ======= Contains GMTSAR image files, modified code, config file and code to make DEMs

----------------- IV ======= Currently not used. In future, we might have image-type specific PIV code, rather than having it all in PIVsuite.

----------------- scheduler ======= A set of functions for finding and submitting image pairs to the HPC job scheduler

----------------- tools ======= A set of helper functions used by the scheduler

- output

--------- Sentinel1

------------------ orbits ======= Contains orbit files used by GMTSAR to align the sentinel1 images

------------------ Antarctica

------------------------- DEM ======= DEMs for every unique image footprint, in .grd format (for GMTSAR), at certain resolutions.

------------------------- metadata ======= .mat files containing information about every image pair over the region

------------------------- mosaic ======= Will eventually hold mosaics at certain spatial and temporal resolutions over the region (or parts of the region)

------------------------ image_pairs ======= holds the output of the feature tracking (and where the processing happens). Organised into unique footprints and date pairs (and swaths)

Image pairs are organised according to unique footprints and date-pairs. For Sentinel1 it looks like

- image_pairs

- PPP -- the 3-digit 'path' identifier. There are lots of unique paths.

- FFFF --- the 4-digit 'frame' identifier. There can be multiple frames per path.

- YYYYMMDD1_YYYYMMDD2 -------- the dates of the first and second image comprising the pair. There can be multiple date pairs for every PPP_FFFF (called a pathframe). There is one directory for each swath under this, called F1, F2 and F3.

Workflow

SCADI.m - This is the master script. It calls 'scheduler' functions for that image type. For Sentinel 1, these are:

1. **run_Sentinel1.m -** this function reads your user options from SCADI.m, finds all of the corresponding image pairs, makes the necessary directories, keeps a log of those pairs in the metadata (described below) and submits the GMTSAR and velocity jobs. Two jobs or tasks are submitted for every swath that is found. Each image pair is comprised of three swaths. That means if you find 50 image pairs, 300 jobs or tasks will be submitted. This function runs interactively in the terminal.
2. **process_Sentinel1_GMTSAR.m** - this function downloads the Sentinel1 SLCs, gets the orbit files, gets or makes the DEM and runs GMTSAR to align and prepare the images for feature tracking. This function runs as a batch job and the processing happens in the PPP_FFFF/YYYYMMDD1_YYYYMMDD2/Fn directory for that swath.
3. **process_Sentinel1_velocity.m** - this function reads the output of process_Sentinel1_GMTSAR.m, does the feature tracking and copies the output to grio1.

Eventually, we'll add other image types and satellite missions, and they'll each have their own set of scheduler functions.

The image pair metadata

SCADI is extremely dependent on having prior information about the number, location and processing stage of each image pair. These are stored in metadata files that are saved in grio1 (and mirrored in my parscratch). When you task SCADI to process a region, SCADI loads the metadata to find suitable pairs and keeps track of those pairs that are submitted. When those pairs finish, or when they fail, the metadata is updated (not necessarily instantly). When new images become available, the metadata is updated. So, please **take care when working with the metadata -** if at all possible, only interact with it using subset_sentinel1_pairs and plot_sentinel1_pairs (described below). If you absolutely must work with it in another way, please be careful not to overwrite it. If you think you need to overwrite it, please check with me and Andrew first.

If disaster strikes and you overwrite or delete the metadata, please let us know as soon as possible. I keep daily snapshots of the metadata for the last 45 days, which can be reinstated if needed.

Simplest use case

To run SCADI, you just need to set your search parameters in SCADI**.m** then run SCADI**.sh** in the command line. SCADI**.sh** just opens a MATLAB window and runs SCADI**.m.**

1. Make a SCADI directory in your home directory. This could be something like: /users/gg1bjd/SCADI/code/test
2. Copy SCADI.m and SCADI.sh to your SCADI directory.
3. Set your processing options in your copy of SCADI.m (as a minimum, you must set the region and image type. See below for details.)
4. Open a bash terminal on Stanage and connect to a worker node by copying this to the command line: `srun --pty bash -i`
5. In the terminal, navigate to your SCADI folder, which should be something like: `cd` /users/gg1bjd/SCADI/code/test
6. Enter: `./SCADI.sh`

In the simplest example, we can find all pairs over Antarctica ever collected, and process them. To do that, you would just need to make the top part of your SCADI.m file look like this:

```
% SCADI

%% GLOBAL/SUBMISSION OPTIONS
% e.g. job resources

% About you
email = 'b.j.davison@sheffield.ac.uk'; % change this to your email please

%% USER SEARCH OPTIONS
% See SCADI_defaults for all options
% As a minimum, you need to provide an image type and region.
% If that's all you provide, this will search for (and submit!) all image
% pairs for that image type and region

% Image type/satellite
type = 'Sentinel1'; % Currently only 'Sentinel1'. Eventually we will add
'Sentinel2', 'Landsat8',9,7,4,5, 'Aster, 'Hillshade' etc. Case sensitive.

% Region
region = 'Antarctica'; % Currently only: 'Antarctica', or 'Greenland'. Eventually
will add 'Svalbard','Canada','High Mountain Asia','Patagonia','Iceland','Alaska'.
Case sensitive.

% Since we're testing, let's make sure we don't submit the pairs
display_only = 1;
```

Once you've set up your SCADI.m file, just connect to the worker node and run SCADI.sh as described above.

Search smart: check how many pairs you will get with subset_sentinel1_pairs

UPDATE: I've added an option to SCADI to allow testing and visulisation of footprint search results prior to submission. If you want to test SCADI without submitting any jobs, just set `display_only = 1;` in your SCADI.m file - this will make some footprint plots and print some information to the terminal about which jobs would have been submitted. Or, if you want to make plots and continue with the submissions, just set `plot_footprints = 1;` in your SCADI.m file (and leave `display_only = 0`, which is the default).

It's normally a good idea to check how many pairs you're going to end up with **before** you run SCADI. I've made a script to help with this called subset_sentinel1_pairs.m. To use this script on Stanage, open a bash terminal on a login node, then do:

`srun --pty bash -i`   <-- this will connect you to a worker node

`module load MATLAB/2022a`

`matlab`

<wait for matlab to load>

Then, using the search settings above, do this in the MATLAB window:

```
% Add the tools to the MATLAB path
if ispc % i'm just adding this in here so I can make this demo on my pc...
    addpath(genpath('D:/work/Fellowship_Leeds/SCADI/code'));
    addpath(genpath('D:/work/Fellowship_Leeds/SCADI/output/Sentinel1/Antarctica/
metadata'))
```

```
    addpath(genpath('D:/work/Fellowship_Leeds/SCADI/output/Sentinel1/Greenland/
metadata'))
else
    addpath(genpath('/mnt/parscratch/users/gg1bjd/SCADI/code')); % I've made this
code accessible to all grio1 users
    addpath(genpath('/mnt/parscratch/users/gg1bjd/SCADI/output/Sentinel1/Antarctica/
metadata'));
    addpath(genpath('/mnt/parscratch/users/gg1bjd/SCADI/output/Sentinel1/Greenland/
metadata'));
end

% Search for pairs
I = subset_sentinel1_pairs(region);
```

That will create a MATLAB structure variable, '`I`', containing metadata for all of the image pairs over Antarctica during the specified time period. Each row of the structure corresponds to an image pair and contains all sorts of useful info about that pair:

```
% Take a look at the metadata for the first image pair
I(1)
```

```
ans = struct with fields:
               Geometry: 'Polygon'
               im1_name: 'S1A_IW_SLC__1SSH_20160929T043621_20160929T043656_013262_0151F6_2C06'
               im2_name: 'S1B_IW_SLC__1SSH_20161005T043539_20161005T043613_002366_004001_D300'
               im1_link: 'https://datapool.asf.alaska.edu/SLC/SA/S1A_IW_SLC__1SSH_20160929T043621_20160929T043656_01
               im2_link: 'https://datapool.asf.alaska.edu/SLC/SB/S1B_IW_SLC__1SSH_20161005T043539_20161005T043613_00
           im1_acquired: '2016-09-29T04:36:56.000000'
           im2_acquired: '2016-10-05T04:36:13.000000'
        time_separation: 6
                    Lon: [6×1 double]
                    Lat: [6×1 double]
            BoundingBox: [2×2 double]
                S1_path: '065'
               S1_frame: '0919'
           S1_pathframe: '065_0919'
             im1_orbit: '13262'
             im2_orbit: '2366'
             direction: 'ASCENDING'
        directory_stub: "065/0919/20160929_20161005"
    F1_processing_stage: 0
    F2_processing_stage: 0
    F3_processing_stage: 0
```

```
% How many pairs did we find?
disp(['This search found ' num2str(numel(I)) ' image pairs']);
```

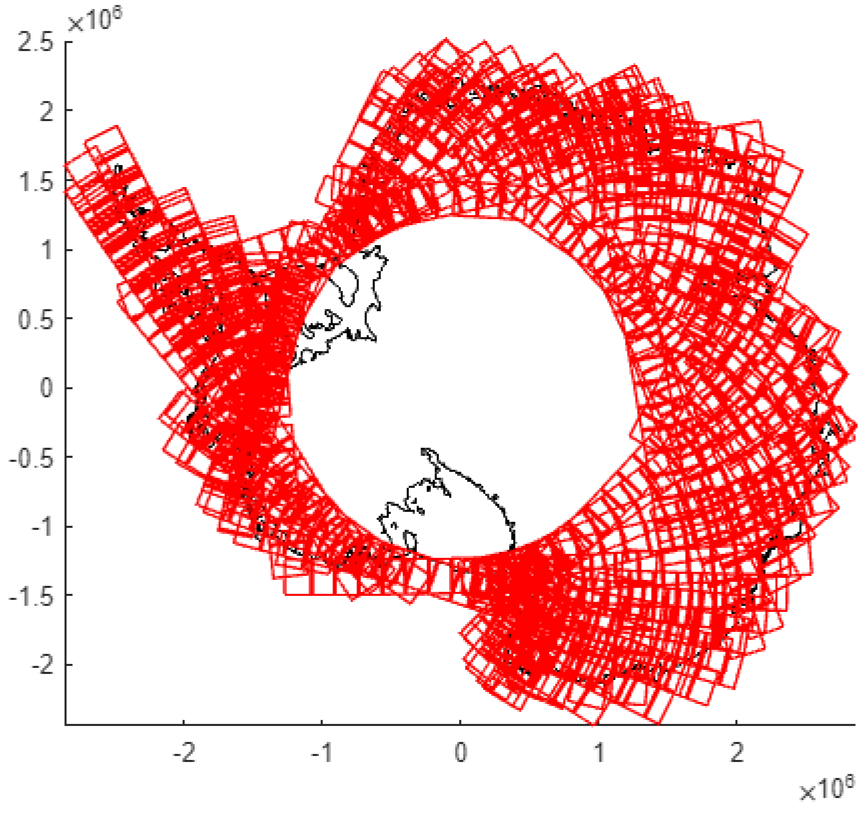
```
This search found 230723 image pairs
```

That's a lot of pairs!

Visualising the footprints

Knowing the number of footprints is helpful, but sometimes it also helps to see where the footprints are. You can do that easily using plot_sentinel1_pairs:

```
plot_sentinel1_pairs(I,'outlines')
```



Since Sentinel-1's orbit is repeated regularly, each footprint is revisited many times, resulting in several date-pairs for a given footprint. We can see how many image pairs are available at each footprint with:

```
%plot_sentinel1_pairs(I,'counts'); % this takes a bit longer to execute,
%especially if you have lots of pairs. Commenting this out because MATLAB
%live script can't handle it!
```

Subsetting by time

We probably don't want to find and submit every single image pair over the region, so we should be more targeted in our search. One way to reduce the number of pairs is to subset your search by time. You just need to specify the start date and the end date in MATLAB's datenum or datetime formats. For example:

```
% Image type/satellite
type = 'Sentinel1'; % Currently only 'Sentinel1'. Eventually we will add
'Sentinel2', 'Landsat8',9,7,4,5, 'Aster, 'Hillshade' etc. Case sensitive.

% Region
```

```
region = 'Antarctica'; % Currently only: 'Antarctica', or 'Greenland'. Eventually
will add 'Svalbard','Canada','High Mountain Asia','Patagonia','Iceland','Alaska'.
Case sensitive.

% Time
time = [datenum(2020,1,1) datenum(2020,12,31)];
```

Will find all pairs over Antarctica in 2020.

```
% Let's see how many pairs we have now:
I = subset_sentinel1_pairs(region,time);
disp(['This search found ' num2str(numel(I)) ' image pairs']);
```

```
This search found 42022 image pairs
```

That's still quite a lot of pairs. Let's subset that even further, but we can speed up the search by providing our structure, `I`, as an input argument to subset_sentinel_pairs:

```
time = [datenum(2020,7,8) datenum(2020,7,14)];
I = subset_sentinel1_pairs(I,time); % provide the structure instead of 'region' as
the first input argument
disp(['This search found ' num2str(numel(I)) ' image pairs. The first pair looks
like this:']);
```

```
This search found 29 image pairs. The first pair looks like this:
```

```
I(1)
```

```
ans = struct with fields:
               Geometry: 'Polygon'
               im1_name: 'S1A_IW_SLC__1SSH_20200708T113034_20200708T113100_033362_03DD8A_7F2F'
               im2_name: 'S1B_IW_SLC__1SSH_20200714T112953_20200714T113019_022466_02AA3D_461E'
               im1_link: 'https://datapool.asf.alaska.edu/SLC/SA/S1A_IW_SLC__1SSH_20200708T113034_20200708T113100_03
               im2_link: 'https://datapool.asf.alaska.edu/SLC/SB/S1B_IW_SLC__1SSH_20200714T112953_20200714T113019_02
           im1_acquired: '2020-07-08T11:31:00.000000'
           im2_acquired: '2020-07-14T11:30:19.000000'
        time_separation: 6
                    Lon: [6×1 double]
                    Lat: [6×1 double]
            BoundingBox: [2×2 double]
                S1_path: '040'
               S1_frame: '0948'
           S1_pathframe: '040_0948'
              im1_orbit: '33362'
              im2_orbit: '22466'
              direction: 'ASCENDING'
         directory_stub: "040/0948/20200708_20200714"
    F1_processing_stage: 0
    F2_processing_stage: 0
    F3_processing_stage: 0
```

Subsetting spatially - coordinates

You can also subset your search spatially by providing at least two coordinates in polar stereographic x/y or lat/lon format. subset_sentinel1_pairs will fit a bounding box around whatever points you provide to do the subsetting. Let's say we wanted all of the pairs over Pine Island Glacier in West Antarctica:

```
[xq,yq] = basin_data('imbie refined','Pine Island','xy'); % get the coordinates of
the PIG basin
time = [datenum(2019,7,1) datenum(2019,7,31)]; % set time to search in July 2019
I = subset_sentinel1_pairs(region,xq,yq,time); % provide the structure instead of
'region' as the first input argument
disp(['This search found ' num2str(numel(I)) ' image pairs']);
```

```
This search found 221 image pairs
```

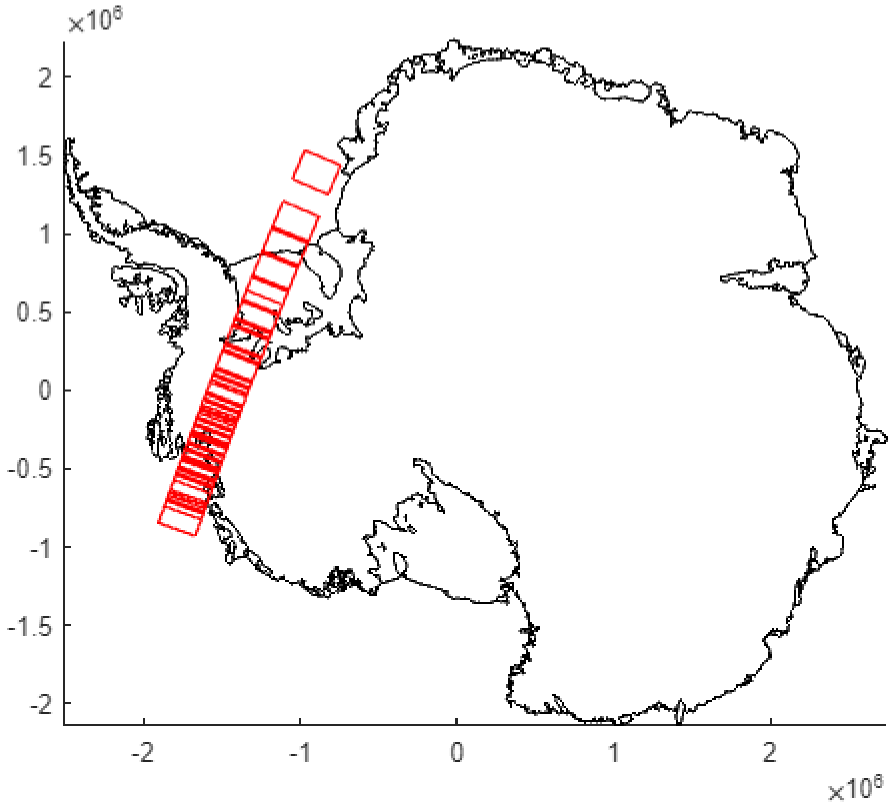
In this example, we used Chad Greene's basin_data function to get the coordinates of the Pine Island drainage basin in south polar stereographic format. You can get the coordinates however you like - e.g. from a shapefile, or just by manually entering them.

Other subsetting options

Subsetting by path and frame

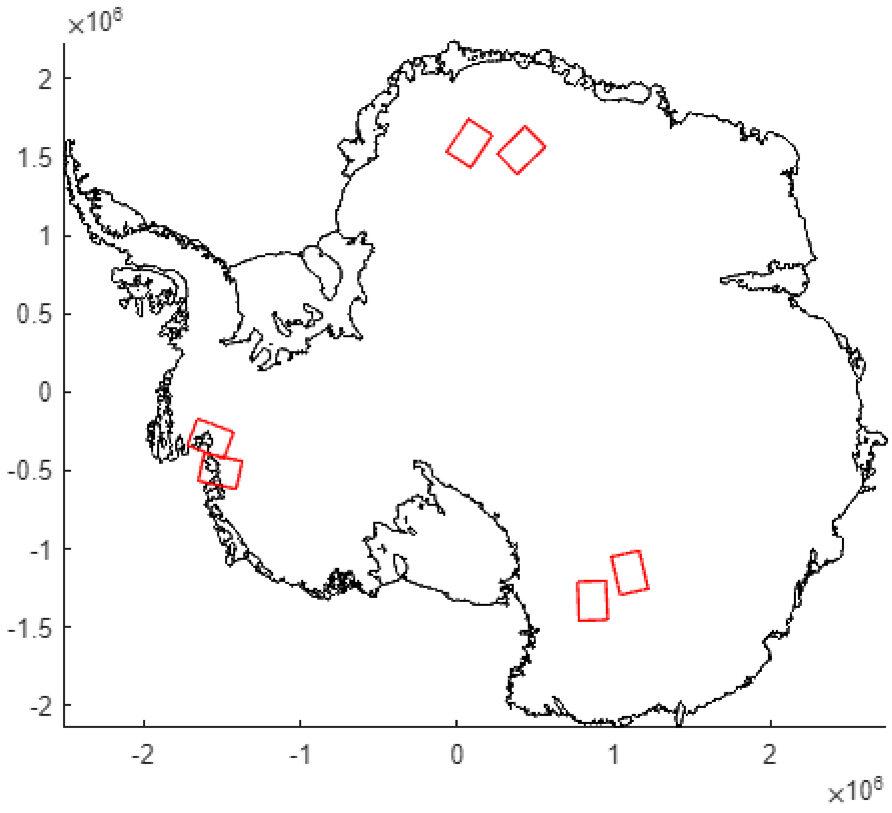
The Sentinel-1 footprints are organised by path and frame - as defined by the Alaska Satellite Facility. ASF's 'Vertex' web browser is also a useful place to identify which path and frame to focus on. We can filter by either or both path and frame using subset_sentinel1_pairs. Let's say we want to view path 65:

```
S1_path = 65; % this can be numeric, char, a string array, or a cell array of
character vectors
I = subset_sentinel1_pairs(region,'path',S1_path);
plot_sentinel1_pairs(I,'outlines');
set(gca,'XLim',[-2.5081    2.7454]*1e6,'YLim',[-2.1439    2.2415]*1e6);
```



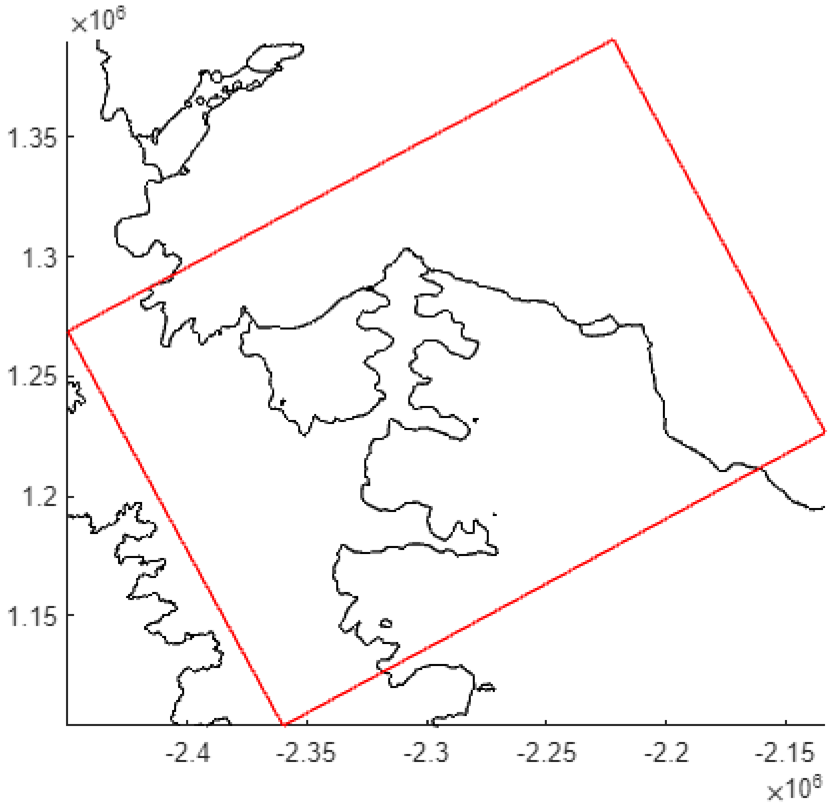
Or we can filter just by frame:

```
S1_frame = 910; % this can be numeric, char, a string array, or a cell array of
character vectors
I = subset_sentinel1_pairs(region,'frame',S1_frame);
plot_sentinel1_pairs(I,'outlines');
set(gca,'XLim',[-2.5081    2.7454]*1e6,'YLim',[-2.1439    2.2415]*1e6);
```



Or we can filter using both path and frame. These can be provided as separate input arguments, or as a single input as follows:

```
S1_pathframe = '038_0818';
I = subset_sentinel1_pairs(region,'pathframe',S1_pathframe);
plot_sentinel1_pairs(I,'outlines')
```



Subsetting by time separation between images in a pair

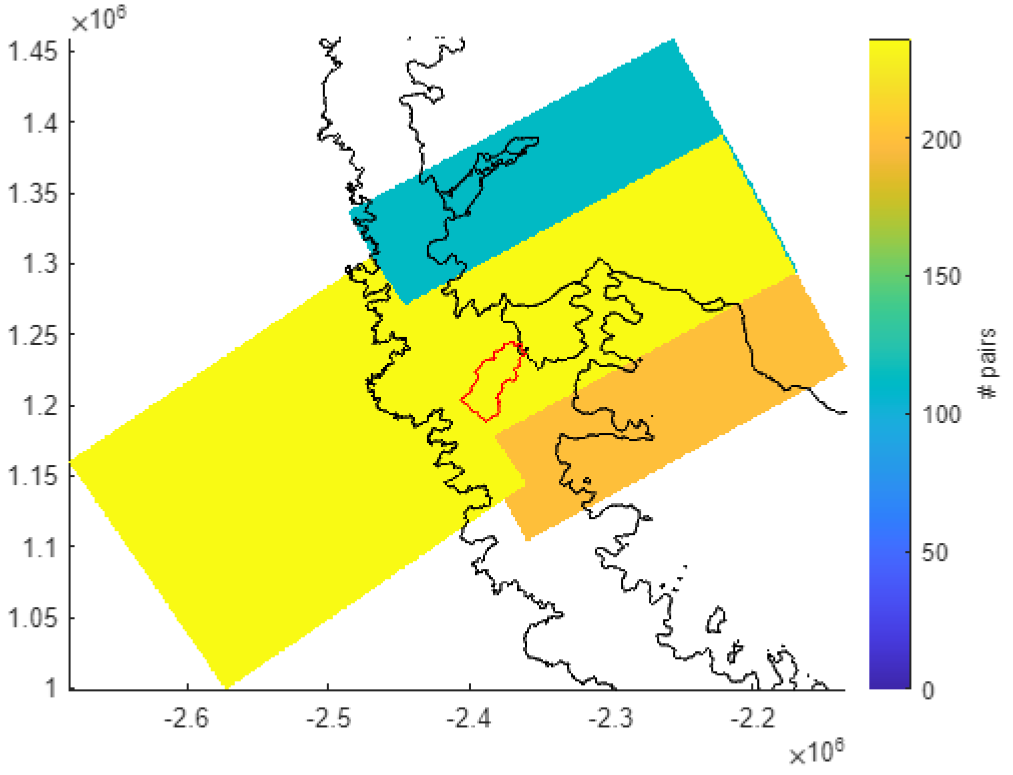
Sentinel-1 has a revisit time of 12 days. In tandem, Sentinel-1a and Sentinel-1b have a revisit of 6-days. With subset_sentinel1_pairs, you can set the time separation target to be 6, 12, 18 or 24 days (or any combination of those). By default, subset_sentinel1_pairs searchs for all of them. Of course, any multiple of 6 will usually exist as a potential pair, but we're not usually interested in longer time separations for feature tracking fast-flowingice, so I haven't saved them in the metadata (but they could be created if necessary). To filter by by time separation, just do:

```
% Let's find just the 6-day pairs over Flask Glacier
S = shaperead('NEB_Basins_reproject.shp');
all_names = string({S.name})'; % get all of the basin names in an easy-to-use
string array
idx = strcmp(all_names,'Flask Glacier'); % find the index of flask glacier
xq = S(idx).X(1:end-1);
yq = S(idx).Y(1:end-1);

% Set the time separation target
time_separation_target = 6;

% Search for pairs
I =
subset_sentinel1_pairs('Antarctica',xq,yq,'time_separation',time_separation_target);
```

```
% Plot them
h = plot_sentinel1_pairs(I,'counts'); % you can save the figure handle if you want
plot(xq,yq,'r'); % add the basin
```



```

```

Subsetting by processing stage

In the image metadata, you'll see some fields called F1_processing_stage, F2_processing_stage and F3_processing stage. These indicate how far through the processing each swath is.

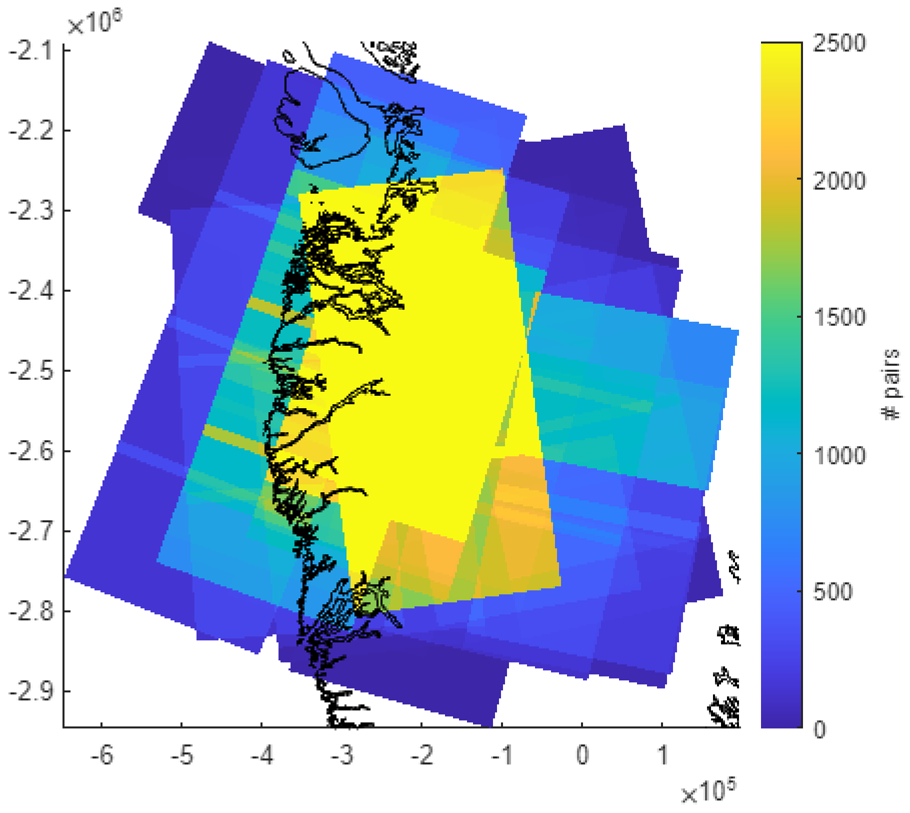
- 0 means this pair hasn't been processed at all
- 1 means GMTSAR has finished successfully
- 2 means that the pair is fully processed and there should be some velocity output for that swath in its output directory
- Any other value should mean that job has been submitted by a user of SCADI but has not yet finished (it might have finished, but the metadata doesn't immediately update).

By default, subset_sentinel1_pairs searchs for all levels of processing completeness (i.e. 0, 1 or 2) except pairs that are currently processing. Usually, if we're submitting pairs for PIV processing, we only want pairs that have not finished, so you should filter by processing stage. Let's say we wanted to find unfinished pairs over the Leverett region of west Greenland:

```
% Set up and do the search
```

```
load('Lev_example_coordinates.mat'); % load some coordinates for Leverett I made
earlier
processing_stage_target = [0,1,2,3]; % search for all pairs
time = [datenum(2014,10,1) datenum(2024,6,15)]; % search whole period
I =
subset_sentinel1_pairs('Greenland',xq,yq,time,'processing_stage',processing_stage_ta
rget); % even though we're providing coordinates over Greenland, we still need to
tell it that we want 'Greenland'

% Visualise the output
[h1,num_pairs_total] = plot_sentinel1_pairs(I,'counts'); % when doing this type of
plot, you can save the figure handle and a grid showing the number of pairs
clim([0 2500])
```



If you want to resubmit pairs that have previously been submitted (which should have a processing stage of something other than 0, 1 or 2), then just include 3 as a processing stage target:

```
I = subset_sentinel1_pairs('Greenland',xq,yq,time,'processing_stage',[0,3]); %
search for pairs that have not been processed or that have been submitted
disp(['We found: ' num2str(numel(I)) ' image pairs']);
```

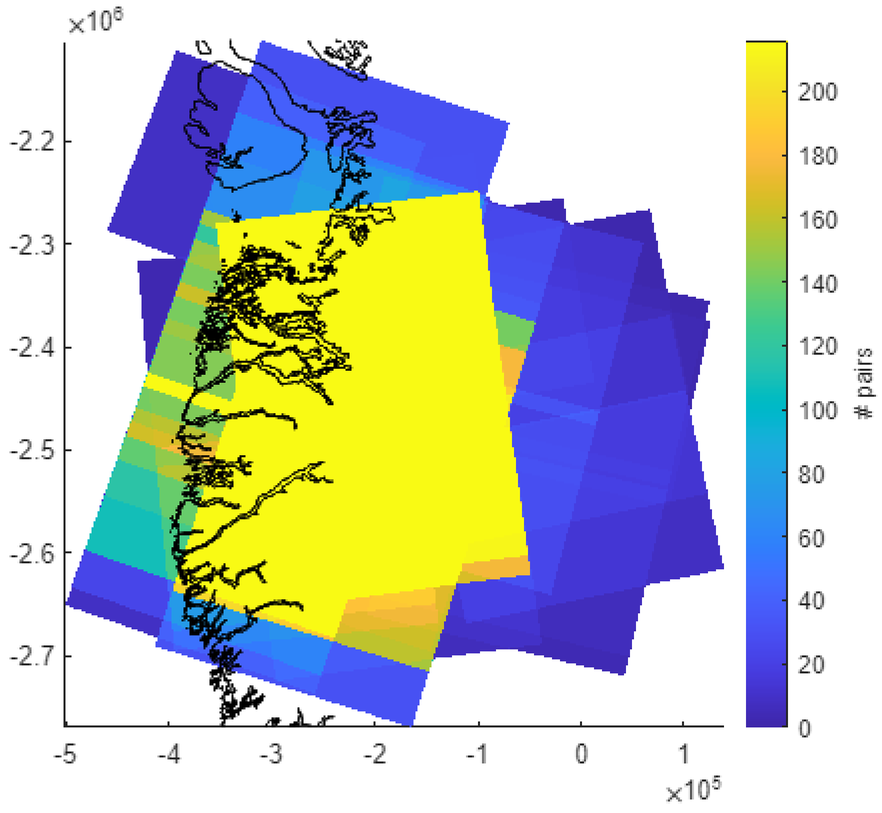
```
We found: 11694 image pairs
```

Take care when doing this: if another user is submitting pairs in the same region and time period, you could end up submitting the same pairs as them. At best, that would waste resources, but it could cause other problems when tracking submissions.

If you're interested in using the output - for example, to make a time-series over a region - then you will usually want to find only pairs that have finished successfully:

```
% This will be nice once the Lev output is moved to /Shared/SCADI
% do the search
I = subset_sentinel1_pairs('Greenland',xq,yq,time,'processing_stage',2);

% visualise the results
[h2,num_pairs_processed] = plot_sentinel1_pairs(I,'counts');
```

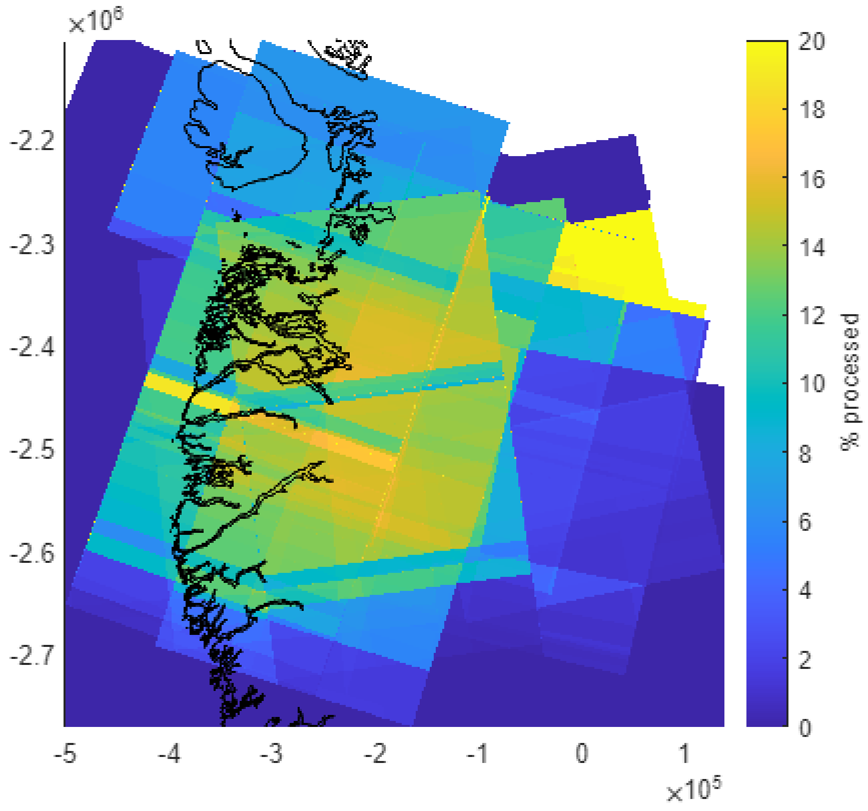


```
plot_xlim = get(gca,'XLim');
plot_ylim = get(gca,'YLim');
```

That information let's us do all sorts of useful things. For example, we can visualise how far through the processing we are:

```
% Now that we have grids of the total pairs and the number of pairs
% processed, we can easily visualise how far through the processing we are
% for any period and location
x_grid = double(-650000:500:849500); % it's always this for Greenland plots
y_grid = double(-650000:-500:-3349500); % it's always this for Greenland plots
figure('color','w')
S = shaperead('Greenland_coast.shp');
greenland('color','none'); hold on
```

```
imagescn(x_grid,y_grid,num_pairs_processed./num_pairs_total.*100); % plot the
percentage of pairs processed
plot([S(:).X]',[S(:).Y]','k'); % plot the coast so it's on top
set(gca,'XLim',plot_xlim,'YLim',plot_ylim);
cb = colorbar;
ylabel(cb,'% processed');
clim([0 20]);
```



Or we can create a list of directories to search to find the image pair output:

```
% % get a string array of directory stubs
% dir_stub = string({I.directory_stub})'; % PPP/FFFF/YYYYMMDD1_YYYYMMDD2
%
% % Set some other variables
% region = 'Greenland';
% type = 'Sentinel1';
%
% % Then loop through each one and get the info you need:
% for i = 1:numel(dir_stub)
%
%     % Set the search path
%     fullPath = ['/shared/grio1/Shared/SCADI/output/' type '/' region '/
image_pairs/' convertStringsToChars(dir_stub(i)) ];
%
%     % Get a list of all speed geotiffs on this path
%     file_list = dir([full_path '/**/*Sp*filt.tif']);
%
```

```
%     % How many did we find
%     disp(['Found ' num2str(numel(file_list)) ' speed geotiffs']);
%
% end
```

Subset to find specific pairs

Sometimes, we know exactly what pair or pairs we want to process and it's easier just to tell SCADI that rather than to construct a set of components that will find that pair. You can do that easily enough with subset_sentinel1_pairs:

```
% Create a list of pairs to search
pair_list = {'038/0818/20210709_20210721','/mnt/autofs/grio1/Shared/SCADI/output/
Sentinel1/Antarctica/image_pairs/038/0818/20210709_20210715'};

% Search for those pairs
I = subset_sentinel1_pairs('Antarctica','pair_list',pair_list);
```

Just note that `'I'` will contain the image pair metadata in the order it finds the pairs, which might not be in the order you provide. subset_sentinel1_pairs always searchs for 6-day pairs first by default, so in the example above the second target pair is found first.

Outputs

At the end of each successful run, SCADI copies the output for each swath to

/shared/grio1/Shared/SCADI/output/ 'type' / 'region' /image_pairs/ 'PPP' / 'FFFF' / YYYYMMDD1_YYYYMMDD2/Fn

The output files include:

1. vector_vel_transp_YYYYMMDD1_YYYYMMDD2.png
2. amp_Sp_ll-S1AYYYYMMDD1_YYYYMMDD2_filt_3.tif
3. amp_Sp_ll-S1AYYYYMMDD1_YYYYMMDD2_orig_3.tif
4. amp_Sp_ll-S1AYYYYMMDD1_YYYYMMDD2_orth_3.tif
5. amp_Sp_err_ll-S1AYYYYMMDD1_YYYYMMDD2.tif
6. amp_U_ll-S1AYYYYMMDD1_YYYYMMDD2_filt_3.tif
7. amp_U_ll-S1AYYYYMMDD1_YYYYMMDD2_orig_3.tif
8. amp_U_ll-S1AYYYYMMDD1_YYYYMMDD2_orth_3.tif
9. amp_U_err_ll-S1AYYYYMMDD1_YYYYMMDD2.tif
10. amp_V_ll-S1AYYYYMMDD1_YYYYMMDD2_filt_3.tif

11. amp_V_ll-S1AYYYYMMDD1_YYYYMMDD2_orig_3.tif
12. amp_V_ll-S1AYYYYMMDD1_YYYYMMDD2_orth_3.tif
13. amp_V_err_ll-S1AYYYYMMDD1_YYYYMMDD2.tif
14. amp_SNR_ll-S1AYYYYMMDD1_YYYYMMDD2.tif
15. amp-S1AYYYYMMDD1_YYYYMMDD2_Xm_BM3D_db.tif
16. amp-S1AYYYYMMDD1_YYYYMMDD2_Xm_db.tif
17. SCADI_PPP_FFFF_YYYYMMDD1_YYYYMMDD2_Fn.txt
18. metadata_ID.mat
19. config_ID.mat
20. manifest_im1.safe
21. manifest_im2.safe
22. s1a-iwn-slc-*-YYYYMMDD1*.xml
23. s1a-iwn-slc-*-YYYYMMDD2*.xml
24. noise-s1a-iwn-slc-*-YYYYMMDD1*.xml
25. noise-s1a-iwn-slc-*-YYYYMMDD2*.xml
26. calibration-s1a-iwn-slc-*-YYYYMMDD1*.xml
27. calibration-s1a-iwn-slc-*-YYYYMMDD2*.xml

What happens in the background?

On submission:

1. A unique ID is written into the 'master' metadata files for those pairs (the master metadata files are stored in e.g. /shared/grio1/Shared/SCADI/output/Sentinel1/Antarctica/metadata)
2. Metadata for each instance of SCADI is saved to /mnt/parscratch/users/gg1bjd/SCADI/output/Sentinel1/Antarctica/metadata/$USER
3. Progress for each task is monitored in /mnt/parscratch/users/gg1bjd/SCADI/output/Sentinel1/Antarctica/logs/$USER - there will be .out and .err files for each GMTSAR and velocity task.

Each night:

1. New orbit files are downloaded and saved to /mnt/parscratch/users/gg1bjd/SCADI/output/Sentinel1/orbits
2. Any new image pairs are identified and appended to the master metadata
3. The metadata is updated to account for newly processed pairs: this just loads the existing metadata and looks for the unique IDs: if velocity output exists for those pairs, it assumes the processing finished successfully. If it cannot find velocity output in grio1, it checks if the job is still running. If the job is not still running, it assumes the processing failed, so it sets the processing stage in the metadata back to zero and keeps a log of the failed jobs. These are stored in e.g. /mnt/parscratch/users/gg1bjd/SCADI/

output/Sentinel1/logs/failed_jobs/SCADI_failed_Antarctica_6day_jobs_YYYY-MM-DD_hhmmss.txt (which should be at about 2 am every day, when the metadata updates)

Limitations

There are a few things that SCADI can't currently do, but that it would be useful to do:

1. You can't easily submit individual swaths. When you search for new pairs, it will find pairs in which any of the swaths match your requirements, then no further filtering is done. This is true even if you specify an individual swath using the list method. I hope to improve this in future.
2. It's not very good at picking up where it left off. If a pair fails part way through, it will always start again from scratch upon resubmission of that pair. This will waste some resources, but is hopefully more likely to lead to successful job completion.
3. Probably lots of other things!

All configuration options

All the default options are stored in e.g. Sentinel1_defaults.m. You can override these by setting your own options in SCADI.m.

General options

```
% Machine options
config.GMTSAR_rmem='4G'; % if you expect to make a DEM, then you will need more
like 20G
config.GMTSAR_time='04:00:00';
config.velocity_rmem='68G';
config.velocity_time='03:00:00'; % note: this is more than enough for PIV
processing with a single pass, no IA shift and with a 25% overlap between pairs.
With 12.5% overlap between pairs, it should just be enough. If doing multiple
passes and or shifting the IAs, you will need to request more time
config.velocity_cpus_per_task='16';
config.amplitude_rmem='64G';
config.amplitude_time='02:00:00';
config.amplitude_cpus_per_task='16';
config.maxNumConcurrentTasks='25';
% if new machines become available that use SGE, more options will need to be added
here

% SCADI general options
```

```
config.display_only = 0; % if 1, just check number of pairs, make directories and
plot, but don't submit or track. Plots are saved in the same directory as SCADI.m
config.test_mode = 0; % if 1, submit the jobs, but save the output to
test_image_pairs
config.test_directory_stub = ''; % if you're testing particular parameter values,
then it's helpful to name the folder something meaningful to reflect those values.
This will append some text to the swath folder name, so you can test various
options without overwriting data
config.plot_footprints = 0; % plot footprints, whether or not you want to test.
Plots are saved in the same directory as SCADI.m
config.track_submissions = 1;
config.skip_GMTSAR = 0; % do not submit GMTSAR
config.submit_amplitude_and_velocity_separately = 1; % submit amplitude image
creation and feature tracking as separate task arrays
config.submit_amplitude_only = 0; % do not submit GMTSAR or feature tracking, only
submit amplitude image creation task array
config.submit_velocity_only = 0; % do not submit GMTSAR or amplitude creation
config.maxNumSubmissions = 333*3; % swaths. If testing, you may want to set this to
1 (in your SCADI.m of course, NOT here)
```

Image search options

```
%% USER SEARCH OPTIONS
config.region = 'Antarctica'; % The region
can be: 'Antarctica','Greenland','Svalbard','Canada','High Mountain
Asia','Patagonia','Iceland','Alaska'. Case sensitive!
config.xq = [];
config.yq = [];
config.latq = [];
config.lonq = [];
config.time = [datenum(2014,10,1) datenum(2050,1,1)];
config.S1_pathframe = [];
config.S1_path = [];
config.S1_frame = [];
config.time_separation = [6,12]; % [6, 12, 18, 24] possible with Sentinel-1
config.processing_stage = [0]; % default is only to find unprocessed pairs. Set to
[0,1,2] to overwrite.
config.listq = [];
config.target_swaths = [1,2,3];
config.use_dual_polarized_images = 0; % If == 1, dual polarized images will be used
to create amplitude images whenever available.
```

GMTSAR options

```
%% USER GMTSAR options
config.GMTSAR_version='master'; % OPTIONS: '6.0', '6.1', '6.2', '6.3', '6.4', or
'master'; master recommended. 6.4 in beta as of 10/05/2024.
config.alignment_method=1; % 1 = standard; 2 = 6par; 3 = ESD; 4 = amp_only
config.DEM_buffer_km = 10;
config.DEM_resolution_m = 100; % for Antarctica, options are 100 or 200; for
Greenland, options are 90 (30 in prep).
```

PIV options

```
%% First order processing options (defaults in brackets)
% decide at what stage to start processing an image pair (depending on how far you
have already got previously).
% 1 = split .grd files and save as individual parts;
% 2 = calculate velocities for individual parts;
% 3 = once the parts have all been processed, join them together and filter them.
config.processing_start_point = 1;

% Decide above what abs(latitude) to use polar stereographic coordinate systems
config.pst_lat_thresh = 55; % (55)


%% Compute options (defaults in brackets)
% GPU
config.saveRAM = 1; % decide whether to read parts of expanded images from file (a
bit slower but with significant memory savings)

% Decide whether to print extra information to the command line (helps with
diagnosing RAM limit issues).
config.debugging = 1;

%% Interrogation area and interrogation step options (defaults in brackets)
config.y_dir_interp_factor = 2; % (2; MUST BE AN INTEGER) oversampling amount in y
direction.
%config.y_multiplier = (config.azimuth_pixel_size/config.range_pixel_size)/
config.y_dir_interp_factor; % ratio between velocity postings in x- and y-
directions (used for plotting individual parts at correct aspect ratio)
% relevant powers of 2: 2, 4, 8, 16, 32, 64, 128, 256, 512
% Xsize and Ysize should be ~4 x max expected motion:
%   - 360 m/yr = ~1m/d = 12 m between 12d repeat images. Minimum Xsize and Ysize at
least 48 m = 21 pixels
%   - 1080 m/yr = ~3m/d = 36 m between 12d repeat images. Minimum Xsize and Ysize
at least 114 m = 50 pixels
%   - 2160 m/yr = ~6m/d = 72 m between 12d repeat images. Minimum Xsize and Ysize
at least 228 m = 100 pixels
%   - 4320 m/yr = ~12m/d = 114 m between 12d repeat images. Minimum Xsize and Ysize
at least 456 m = 198 pixels
%   - 8640 m/yr = ~24m/d = 228 m between 12d repeat images. Minimum Xsize and Ysize
at least 912 m = 396 pixels
% X direction (Nagler 2015 uses 144 and 40)
config.Xsize = 512; % actual size: Xsize*~2.3m (if config.y_dir_interp_factor =
default). Make bigger for inland? Should be powers of 2?
config.Xstep = 128; % actual step: Xstep*~2.3m (if config.y_dir_interp_factor =
default). Should be powers of 2?
```

```
config.Ysize = 128; % if you set y_dir_interp_factor to be other than 2, you may
wish to modify these
config.Ystep = 32;  % ~25 percent step
% and IAStep are similar to those in the X direction).
% - range pixel size: ~2.4 m (but varies for each image pair)
% - azimuth pixel size: ~14.0 m (but varies for each image pair)
% if config.y_dir_interp_factor == 6 % (i.e. oversampled y pixels ALMOST same size
as x pixels)
%     config.Ysize = [256]; % (256)
%     config.Ystep = [64]; % (64)
% elseif config.y_dir_interp_factor == 5
%     config.Ysize = [224]; % (224)
%     config.Ystep = [56]; % (56)
% elseif config.y_dir_interp_factor == 4
%     config.Ysize = [192]; % (192)
%     config.Ystep = [48]; % (48)
% elseif config.y_dir_interp_factor == 3
%     config.Ysize = [128]; % (128)
%     config.Ystep = [32]; % (32)
% elseif config.y_dir_interp_factor == 2 % DEFAULT
%     config.Ysize = [96]; % (96)
%     config.Ystep = [24]; % (24)
% elseif config.y_dir_interp_factor == 1
%     config.Ysize = [48]; % (48)
%     config.Ystep = [12]; % (12)
% else
%    disp(['Oversampling factor ' num2str(config.y_dir_interp_factor) ' not yet
coded; code is  stopping']);
%    return
% end
% Reduce IA Step size to compare outputs these 'fine resolution' with the standard
values
config.fine_res = 0;

%% Ice velocity plotting options (defaults in brackets)
config.nan_boundary = 400; % get rid of data in azimuth direction 'frayed edges'
config.vel_scale = 200; % (200 for lev). Velocities in plots are scaled so that
this is the maximum for quiver arrows.
config.vel_plot_caxis_lim = 200; % (200 for lev) upper limit for combined velocity
pcolor plot
%config.max_allowed_speed = 0.8*(sqrt((config.range_pixel_size*(config.Xsize/
2)).^2 + (config.azimuth_pixel_size*(config.Ysize/config.y_dir_interp_factor/
2)).^2)/config.time_diff*365.25); % 80% of max detectable velocity based on size
of IAs


%% Pre-processing options (defaults in brackets)
% Radiometric calibration of GMTSAR amplitude.grd.
% - The beta nought convention is usually preferable for use as the 'native' radar
brightness estimate inthe initial processing of SAR imagery [11], as it gives the
best unencumbered estimate of what the radar actually measured.
% - Helmut Rott uses sigma0. So for purposes of comparison that is the one
%   to go for.
% Also see: https://pro.arcgis.com/en/pro-app/3.0/help/analysis/image-analyst/
analysis-ready-sentinel-1-grd-data-generation.htm
% https://hyp3-docs.asf.alaska.edu/guides/rtc_product_guide/
config.rad_cal = 'sigma'; % Options are: 'sigma' (USE), 'beta', 'gamma',
'dn', 'none'. See: https://sentinel.esa.int/documents/247904/685163/S1-Radiometric-
Calibration-V1.0.pdf
config.do_rad_terrain_correction = 0; % Do radiometric terrain correction. Terrain
correction not recommended for ice surfaces!
config.noise_lut = 1; % (?) apply noise reduction look up table to sentinel 1 SLC
data
config.check_input_images = 0; % (1) make a plot of input images to check for gaps/
stripes etc.
config.make_amplitude_geotiffs = 1; % (1) make amplitude geotiffs (may take a
while... but useful for analysis of surface lake drainages etc). 0=none, 1=master
only, 2=master and slave.
config.amp_res = 15; % (15 or 10) m: resolution
of amplitude geotiffs. Must not be less than
either (config.sub_sample_y*azimuth_pixel_size*config.y_dir_interp_factor) or
(config.sub_sample_x*range_pixel_size) or there will be lots of holes in the
geocoded outputs
config.reduce_amp_speckle = 1; % (3) 1: produce raw and unspeckled geotiffs,
2:produce raw geotiffs only, 3:produce unspeckled geotiffs only
config.amp_only = 0; % (0) only make amplitude geotiffs and then stop processing
(i.e. don't continue to full velocity processing). If 'adding' amplitude images to
existing velocity pairs, you must first delete the SCADI completion text files from
the relevant velocity folders, or the run will never start.
config.retain_single_precision = 1; % (1) if =1 keep the input images as single
precision (32 bit), if =0 cast to 16 bit integers (32 bit will obviously use more
RAM)
config.sub_sample_y = 1; % (1) subsample Y range and azimuth grids
config.sub_sample_x = 3; % (3) subsample X range and azimuth grids
config.fill_nans = 1; % (1) fill nans in images before doing velocity processing
(shouldn't be any??)
config.buffer_mult = floor(config.Ysize./config.Ystep)+1; % Ensure overlap between
input image parts. Actual buffer width = config.buffer_mult*config.Ystep*~2.3m
config.interp_method = 2; % (1) 1=bilinear, 2=bicubic (see interpolation method for
oversampling input images in e.g. prep_amplitude.m)

% Filtering options (some more of these are set in run_GMTSAR_ice_velocity_code)
config.pre_proc = 1; % (1) apply high pass filter, or minus low pass filter (2) to
the input images to remove longer wavelength variations (0 = use raw images) df = 1
config.filtSigma = 0.5; % (2) low pass filter sigma (if config.pre_proc = 2)
config.butter_ord = 1; % (1) order of butterworth high pass filter (if
config.pre_proc = 1)
config.CLAHE = 1; % (0) choose whether to equalise the image brightness prior to
applying the high pass or butterworth filter (applied in prep_amplitude.m)
config.min_max_filt = 0; % (0) use min max filter on input images
config.low_pass_filt_inputs = 0; % (0)  if config.AJS_lev_summer_sparse = 1 this
will be overwritten between June 1st and October 1st
config.low_pass_filt_method = 2; % (2) will have no effect
config.low_pass_filt_inputs = 0; % (0)

%% Cross-correlation options (defaults in brackets)

% Set the cross-correlation method:
%  - 'fft' (cross-correlation computed using fast Fourrier transform: very fast and
works with complex data
%  - 'fta' alternative fft code: fast and works with complex data
%  - 'dcn' cross-correlation evaluated using convolution
%  - 'dft' see Manuel Guizar-Sicairos, Samuel T. Thurman, and James R.
%          Fienup, "Efficient subpixel image (usesnormxcorr2_general to calculate
%          cross-correlation values)
%          registration algorithms," Opt. Lett. 33, 156-158 (2008). I THINK THIS IS
BEST: fast, can do complex
%          data and has anvanced sub-pixel identification of cc peaks.
%  - 'nxc' normalized cross correlation (Matlab function which uses either
%          fft or conv depending on image size): much slower, can do complex data
%          (normxcorr2_general can't)
%  - 'cnv' cross correlation: much slower but can deal with complex data.
%  - 'dcn' cross correlation using convolution: THE SLOWEST and cannot deal with
complex data
% Default setting is |'fft'| for first iterations, and |'dcn'| for final iterations
config.ccMethod = {'dft'  'dft'  'dft'  'dft'}; % ('dft' for all iterations)
config.usfac = 50; % (50) 1/config.usfac = precision for finding sub-pixel peaks if
config.ccMethod = 'dft'
config.validation_passes = 0; % if>0 do PIVsuite internal median filtering this
number of times; if=0 don't do any internal median fioltering (there is lots of
filtering doe outside PIVSuite anyway)


% for config.ccWindow (below): if IA size is >4 x expected velocity, weighting the
% cross-correlation matrix reduces the chances of getting an anomalous
% result with velocities close to the edge of the cross-correlation matrix
% (i.e. velocities 2 x as big as they should be)
config.ccWindow = 'Uniform'; % ('Uniform') 'Gauss'; 'Uniform'; 'Parzen'; 'Hanning';
'Nogueira'.


% iaMethod ... way, how interrogation area are created. Possible values are
%    'basic' ... interrogation areas are regularly distribute rectangles
%    'offset' ... interrogation areas are shifted by the estimated displacement
%    'deflinear' ... deformable interrogation areas with linear deformation
%    'defspline' ... deformable interrogation areas with spline deformation
%    - Note: if Uest and Vest contains only zeros or if they are empty/unspecified,
'basic' method is
%    always invoked regardless .iaMethod setting
% iaImageToDeform ... defines, which image should deform. It is taken in account
if .iaMethod is
%    'deflinear' or 'defspline', or if it is 'offset' (then it defines, in which
image IAs are
%    shifted). Possible values are
%    'image1', 'image2' ... either im1 or im2 is deformed correspondingly to Uest
and Vest
%    'both' ... deformation both images are deformed by Uest/2 and Vest/2. Sligthly
more CPU time is
%    required.
% iaImageInterpolationMethod ... way, how the images are interpolated when
deformable IAs are used
%    (for .iaMethod == 'deflinear' or 'defspline'. Possible values are:
%    'linear', 'spline' ... interpolation is carried out using interp2 function
with option either
%    '*linear' or '*spline'
% These are overwritten if it determines we're using a GPU later
config.iaMethod = 'deflinear';     % ('basic', 'deflinear', 'defspline') 'offset'
creates issues with rounding of velocities in pixels units (i.e. many velocities <
1 pixel?) - commented out by BJD for testing
config.iaImageInterpolationMethod = 'linear'; % ('spline') choose 'linear' to be
same as GPU but 'spline' also works;
config.iaImageToDeform = 'image2'; % slave is deformed if required


% Decide whether to pair slightly shifted interrogation areas: increases
% signal to noise ratio resulting in much better coverage (with a
% significant speed penalty).
% - 0 = do not shift IAs - results in much lower signal to noise
%   ratios and hence less good data coverage (because of SNR limit for good
%   data (config.SNR))
% - 1 = faster but not quite correct (?)
%   way (all IAs cut on one side and padded with values grabbed from adjacent
%   IAs on the opposite sides), but minimal RAM
%   usage and works well;
% - 2 = proper way to shift IAs: separate shifted
%   expanded images created in pivInterrogate_GPU. Slower and uses lots
%   of RAM.
config.shift_IA = 0;
config.num_IA_shift_iters = 5; % 5 or 10; 5 only moves the IA diagonally, by up to
2 pixels; 10 also moves the IA vertically and horizontally.

% decide whether to iteratively modify the IAs with noise and repeat the
% cross-correlation. If shifting IAs as well, num_cc_noise_iters will equal
% num_IA_shift_iters
config.iterate_cross_correlation_with_noise = 0;
config.num_cc_noise_iters = 0;
```

```
% Decide whether to segment filter and fill (clean) between runs
config.clean_between_runs = 0; % decide whether to remove outliers and fill the
data a bit between cross-correlation passes


%% Velocity calculation options
config.surf_par_flow = 0; % (0) decide whether to just use displacement in line of
site scaled based on DEM slope direction (DOES NOT YET WORK)
config.vel3D = 0; % (0) decide whether to incorporate vertical velocities (i.e.
distance between start and end positions projected onto the DEM)


%% Post-processing options for each part of the SLC arrays (defaults in brackets)
% General PIVSuite options
config.remove_noise_floor = 0; % (0) decide whether to set to zero velocities
smaller than the precision of the code and satellite data. Generally set to zero,
but can be useful for delimiting ice margin sometimes.
config.rem_unphys_speeds = 1; % (1) remove speeds above a specified physical limit
(max_allowed_speed)
config.inpaint_nans_method = 1; % (1, 4 also quite good) filling method for
inpaint_nans. See inpaint_nans documentation for more details
config.off_ice_vel_thresh = 22; % (22 m/yr) 'noise threshold' (1/10 pixel
displacement over 12 days) threshold velocity for ignoring from median direction
and median magnitude filtering (avoids removing good on-ice data close to ice
margins): ~sqrt((0.23/12*365).^2 + (0.7/12*365).^2)

% Image segmentation filter (replaces pixel difference filter)
config.seg_filt = 1; % (1) removes small regions of pixels with similar values
but leaves large blocks of smoothly varying pixels. See: Luttig et al 2017 A
Combined Approach for Filtering Ice SurfaceVelocity Fields Derived from Remote
Sensing Methods
config.seg_filt_diff_thresh = 400; % (400 for lev (land-terminating with df
IAStep), 2000 for kns (marine-terminating)) set a maximum threshold difference
between adjacent pixels (m/yr). OR: config.Xstep.*7
config.seg_filt_pix_num_thresh = 10; % (20) set a minimum threshold number of
pixels to be kept as a segment

% VSNR filtering options
config.VSNR = 0; % (1) decide whether to use VSNR filter on V velocities (more
noisy and lower resolution and displaying ionospheric 'streaking'). Generally
useful, but not so effective if preproc = 1 or 2.

% Gaussian noise filtering options
config.weiner_filt = 1; % (1) Decide whether to filter the V velocities to reduce
gaussian noise
config.weiner_filt_size = 3; % (3) diameter of weiner filter (units is velocity
postings: config.Xstep and config.Ystep)

% Flow direction filter options
```

```
config.flow_dir_filt = 1; % (1) filter flow direction for final combined data:
difference between each cell and median within kernel
config.flow_dir_angle_thresh = 45; % (45) maximum allowed difference between a
pixel and the median of its surroundings
config.dir_filt = 'median'; % ('median') method for filtering direction: 'median'
or 'mean'

% Median filter for U and V options
config.med_filt = 1; % (0) median filter data of individual parts of the SAR image.
NaN if difference between individual cell and median within kernel > multiple of
std.
config.med_filt_size = 7; % (3) size of kernel for median filter
config.med_filt_iters = 4; % (4) Number of times to run the median filter

% Gaussian filter for U and V
% - Individual parts
config.gauss_filt_parts = 0; % (1)
config.gauss_filt_parts_size = 3; % (3)
config.gauss_filt_parts_sigma = 0.5; % (0.5)
% - Combined swath
config.gauss_filt_swath = 1; % (1)
config.gauss_filt_swath_size = 3; % (3)
config.gauss_filt_swath_sigma = 0.5; % (0.5)

% Combined swath binning method
config.swath_bin_method = 1; % (1). 1=nanmedian, 2=nanmin

% Outlier filter options
% - Grubbs
config.outlier_remove_grubbs = 0; % (0) decide whether to remove outliers using
block processing and an iterative Grubbs test.
config.num_blocks_outlier = 400; % (400) decide on number of bins (i.e.
linspace(min(U):max(U),config.num_bins)
config.outlier_removal_significance = 0.02; % (0.02) significance level to remove
outliers. larger numbers remove more outliers, but beware also removing good data.
config.grubbs_iterations = 1; % (1) number of times to do the Grubbs outlier
removal at each block size
% - Gridfit
config.outlier_remove_gridfit = 0; % (0) decide whether to remove outliers by
removing misfits with a smoothed surface representation of the data.
config.outlier_removal_gridfit_smoothing = 2; % (2) gridfit smoothing factor for
surface to which raw data are compared
config.outlier_removal_gridfit_stdev_thresh = 3; % (3) raw data with a difference
config.outlier_removal_gridfit_stdev_thresh x stdev greater than mean difference
between raw and smoothed will be removed
config.outlier_removal_gridfit_niter = 3; % (2) number of times to iteratively run
gridfit outlier removal

% Cross-correlation quality filter options
```

```
config.filter_data_quality = 1; % (0 or 1 (and 3 below) if using 'dft' as cross-
correlation method) if=1 remove spurious data as defined by pivsuite, otherwise
if=0 don't bother. Best to set as 0 if using dft for cross-correlation as this
calculates sub-pixel peak separate to normal PIVSuite methods.
config.quality_filter = 3; % (3) parameter for config.filter_data_quality. 1=median
test fail or cross-correlation fail are assigned NaN; 2=median test fail or cross-
correlation fail or peak detect fail are assigned NaN; 3=peak_detect_fail==1 |
interpolated==1

% SNR filter options
config.snr_filt = 1; % (1) use signal to noise ratio filter
config.SNR = 4; % (5.8 generally, although 4  sometimes better to include more
data). Signal to noise ratio threshold (see de Lange et al 2007 Fig. 5). Vectors
with ratios < SNR will be converted to NaNs and may be in-filled.

% Sparse data filter options
config.remove_sparse_data = 0; % (0) remove data based on point density of U, V in
a scatter plot (see Adrain and Westerweel)
config.sparse_data_removal_method = 1; % 1 = kde2d
method (see: https://uk.mathworks.com/matlabcentral/fileexchange/17204-kernel-
density-estimation?focused=5829342&tab=function); 2 = my own method
config.AJS_lev_summer_sparse = 0; % if 1, then stricter options will be used
between June 1st and October 1st

% Spatially isolated cells filter options
config.remove_isolated_cells = 1; % (1) Decide whether to remove spatially isolated
cells
config.remove_isolated_cells_thresh = 30; % (30) Remove cells with fewer connected
pixels

% BW morph filter
config.BW_morph_filt = 1; % (1) Tidy up the edges of good data regions and remove
small spatially isolated good data pixels


%% Output plots
config.vel_overlay = 1; % (1) Make a figure of transparent velocity magnitude and
vectors over the master amplitude image if it exists, or otherwise a hillshade of
the input DEM.
config.vel_overlay_fast = 1; % (1) Quickly, make a lower quality figure of
transparent velocity magnitude and vectors over the master amplitude image if it
exists, or otherwise a hillshade of the input DEM.

%% Radar data splitting options (defaults in brackets)
% Split input radar amplitude data into this many parts along the long axis (i.e.
oversampled azimuth)
% The below splits should make the processing possible on a machine with
% 64Gb of RAM
```

```
config.num_parts = 'default'; % allow number of parts to be calculated on the fly.
If you think you know best, then go ahead and set it to some integer, probably
between 4 and 32.

% Set satellite name (this will be overwritten when the metadata xml files
% are parsed in prep_amplitude_v2.m)
config.sat = 'S1A';
% Decide on whether to join the individual parts back together to form a
% single geotiff
config.join_parts = 1; % (1)


%% Housekeeping (defaults in brackets)
config.del_images = 0; % (0) decide whether to delete amplitude tiffs/mat files
after velcoities have been calculated. Set as 0 if debugging, 1 otherwise?
```

Making a new region

If you want to start a new region - talk to Ben!

New metadata

You'll need to create the metadata for that region:

1. Make a new output directory for that region
2. Download some kmls for that region (see download_kml.txt in /shared/grio1/Shared/SCADI/code/Sentinel1/tools). You'll need to get the coordinates that surround your region manually.
3. Run build_sentinel1_metadata for that region (see /shared/grio1/Shared/SCADI/code/Sentinel1/tools)

Note that some care will need to be taken if those regions are in places not suitable for using south or north polar stereographic coordiantes

You should then:

1. Add options for that region in update_sentinel1_metadata
2. Add that region to run_update_sentinel1_metadata.sh, so that it updates daily

The image pair subsetting and plotting has region specific options which will need updating. Again, take care with any conversions from lat/lon to x/y.

New DEM

1. Find a DEM covering the whole region, download it and put it in users/gg1bjd/Data and /mnt/parscratch/gg1bjd/SCADI/output/Sentinel1/<region>/DEM. It might be wise to buffer the DEM with some distance around the region, so that you can use footprints that only partially overlap the original DEM.
2. Update SCADI_get_reference_DEM_name with the new region
3. Update make_all_sentinel1_DEMs_task_array.m, make_single_GMTSAR_DEM.m and crop_framewise_DEM.m to work with the new region
4. Run make_all_sentinel1_DEMs_task_array.m interactively to make all of the new DEMs for that footprint.
5. in run_sentinel1 - add a check for that DEM resolution when setting the default options
6. Update scrontab to keep these DEMs alive

Update scheduler code

There are region specific options throughout the code. These will need to be modified accordingly (it would be nice to determine the region from the coordiantes or something).
