10TH ADVANCED TRAINING COURSE ON LAND REMOTE SENSING

Natural disturbances of forests : exercise

Oleg Antropov

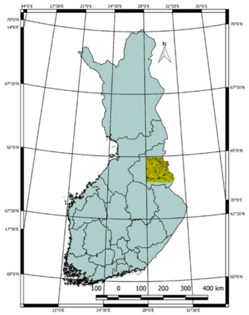
Introduction

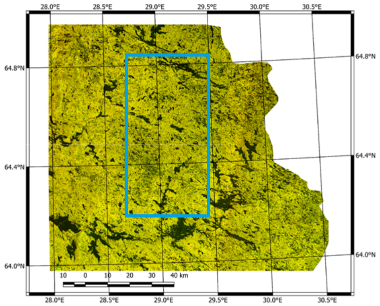
- Practical session consists of a hands-on exercise on mapping snow-induced forest damage using multitemporal Sentinel-1 data over boreal forest
- This is one of least visible and difficult use-cases in mapping natural disturbances of forest area (compared to mapping burned forest areas or windstorm damage).
- Sentinel-1 data pre-processing is done using the ESA SNAP software
- Mapping snow-damaged forest is implemented as supervised binary classification using machine learning (support vector machines) and Python scikit-learn library

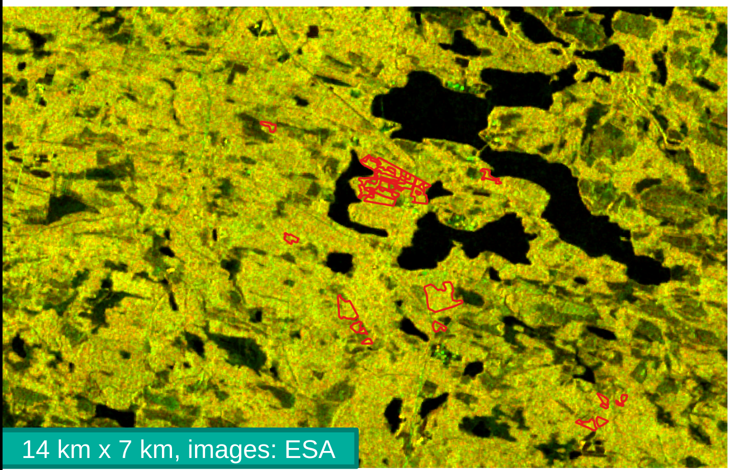
More information:

1. O. Antropov, Natural disturbance of forests : lecture, ESA Land Training 2021, Ljubljana, Slovenia, September 21, 2021
2. E. Tomppo, O. Antropov, J. Praks. Boreal forest snow damage mapping using multi-temporal Sentinel-1 data. *Remote Sensing*. 2019; 11(4):384.
3. M. Fitrzyk, SNAP S1 exercise: Forest monitoring, ESA Land Training 2021, Ljubljana, Slovenia, September 20, 2021

Study area, SAR and reference data

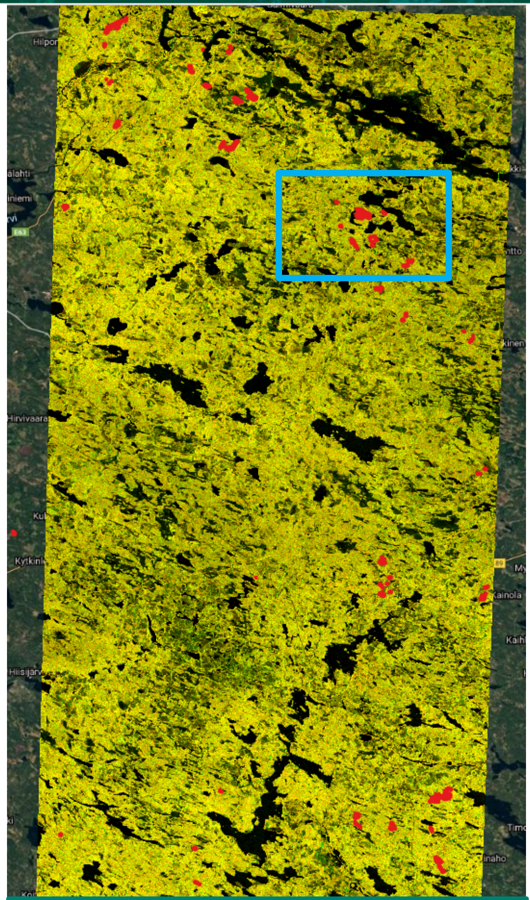






14 km x 7 km, images: ESA

Multitemporal composite of Sentinel-1 images (green-VH, red-VV). Red polygons denote sanitary cutting reports from Forest Centre (Metsäkeskus,2018).



40 km x 80 km, images: ESA, Google Earth

List of Sentinel-1 scenes

Image	Date	Mode	Polarization
1	12 November 2017	IW	VV, VH
2	24 November 2017	IW	VV, VH
3	6 December 2017	IW	VV, VH
4	18 December 2017	IW	VV, VH
5	30 December 2017	IW	VV, VH
6	11 January 2018	IW	VV, VH
7	23 January 2018	IW	VV, VH
8	4 February 2018	IW	VV, VH
9	16 February 2018	IW	VV, VH
10	28 February 2018	IW	VV, VH
11	12 March 2018	IW	VV, VH
12	24 March 2018	IW	VV, VH

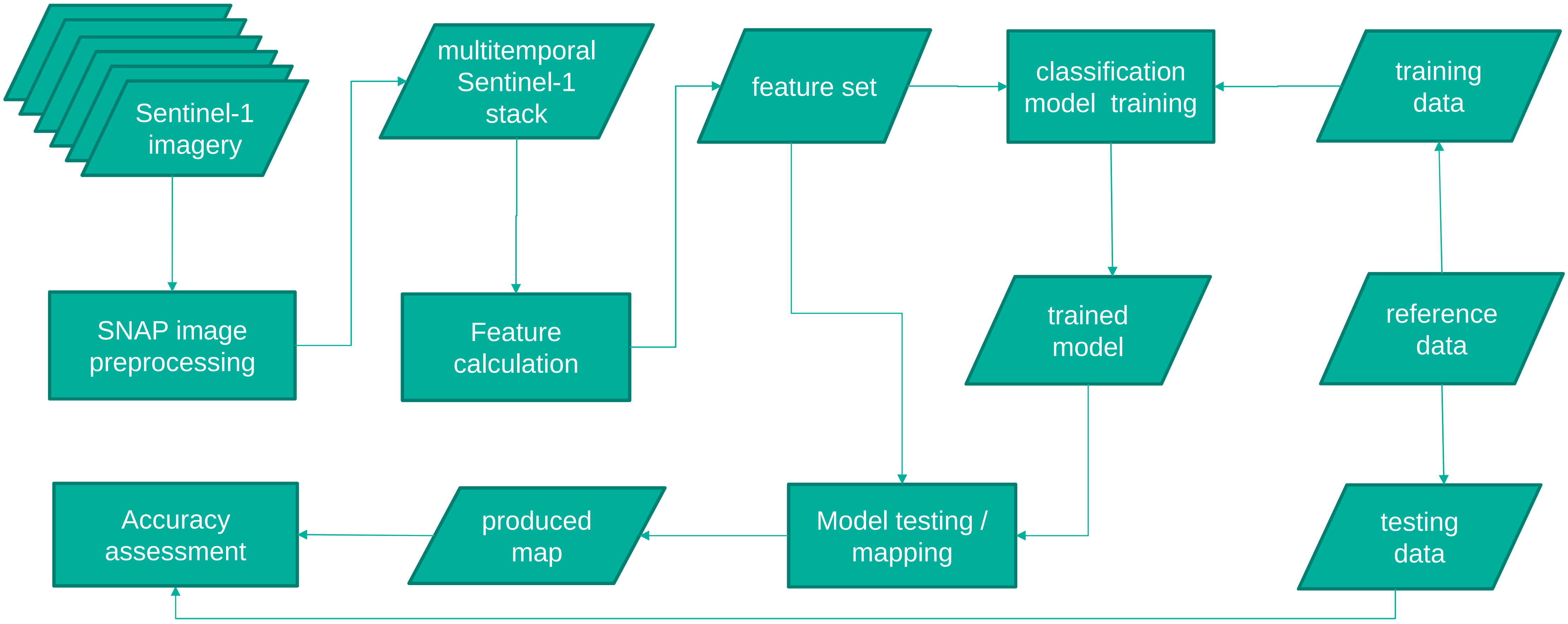
Reference data

Sanitary cutting reports were available from the Finnish Forest Centre (Metsäkeskus, 2018), along with MS-NFI data from Natural Resources Institute Finland (Luke, 2018 for sampling non-damaged stands.

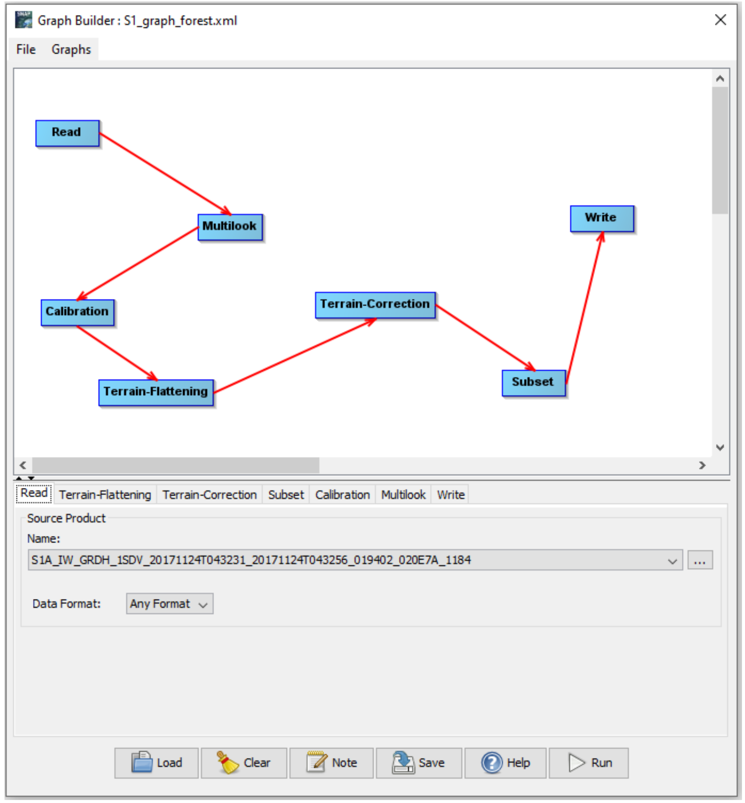
Training dataset: a random sample of 100 damaged forest stands and 100 intact forest stands

Accuracy assessment (testing) dataset: independent random sample of 100 damaged forest stands, and 100 intact forest stands

Snow-damaged forest mapping: overall approach



SNAP pre-processing (1)



SNAP->Tools->GraphBuilder

Sentinel-1 image orthorectification:

- Multilooking
- Calibration
- Terrain-flattening
  - external/local DEM used here
- Terrain-correction
  - external/local DEM used here
- AOI subset retrieval
- Reprojection
- Projection

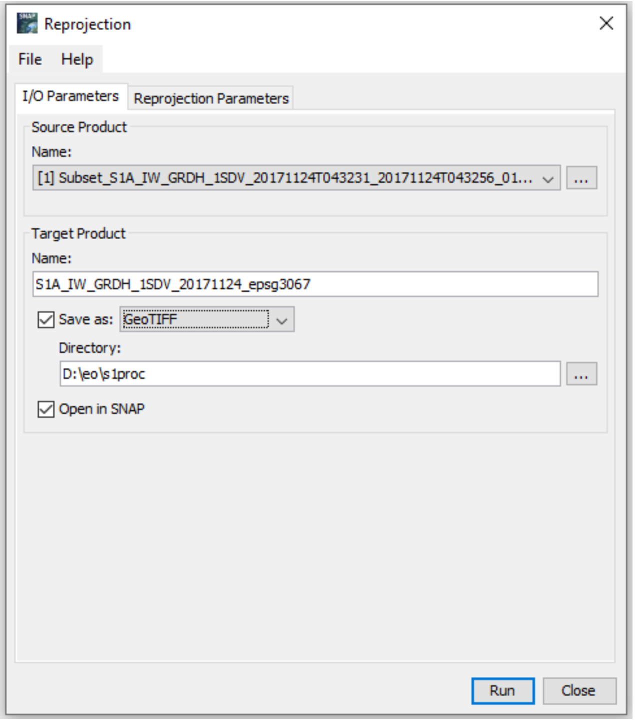
*thermal noise removal, speckle filtering can be included

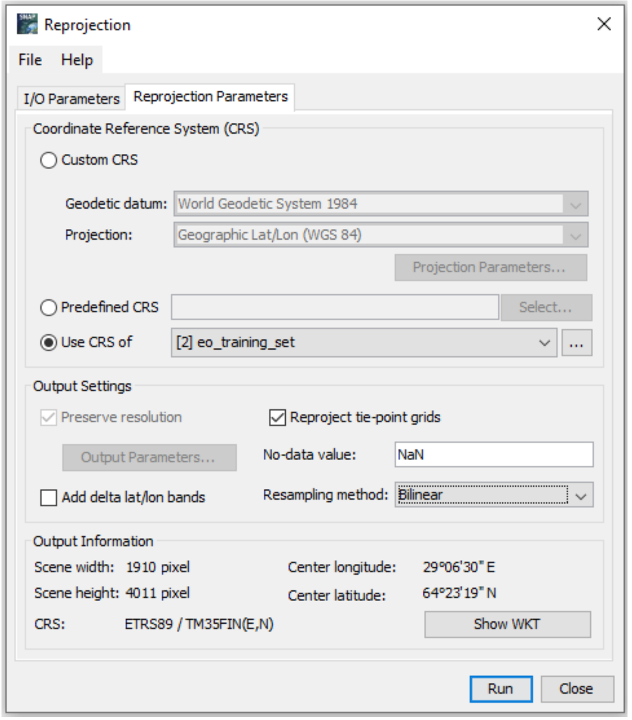
Action points:

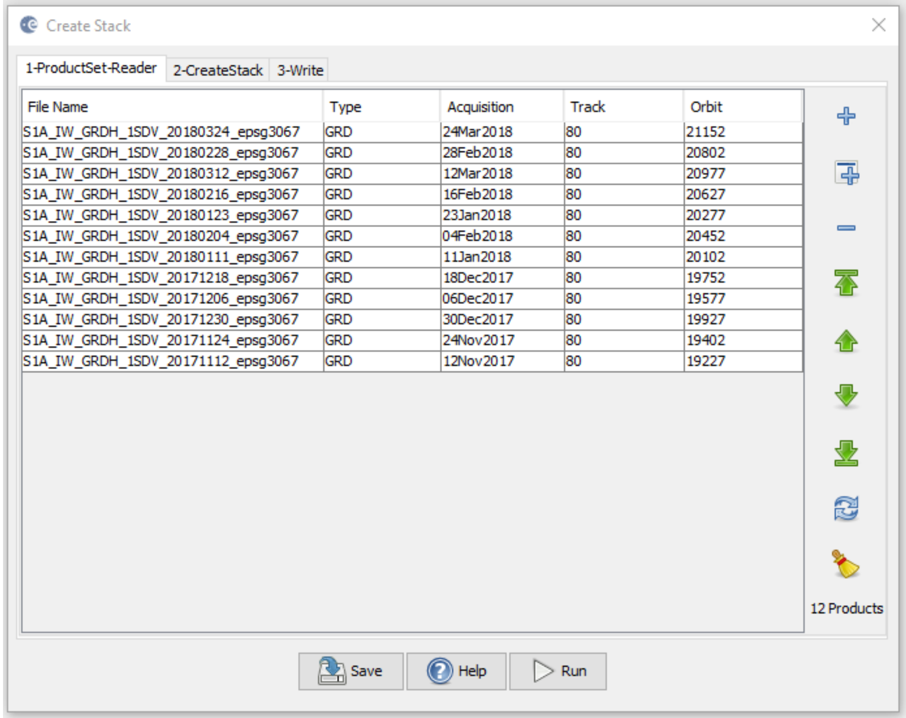
- Load prepared graph
- Investigate parameters
- Run the graph in SNAP using chosen Sentinel-1 image

*graph can be run command-line using gpt-command;
*there, operating parameters/variables can be passed as variables
*Python module “snappy” can be used for customizing EO data processing chains within Python scripts

SNAP pre-processing (2)







SNAP->Raster->Geometric->Reprojection

SNAP->Radar->Coregistration->...
-> Stack Tools->Create Stack

Python processing (1)

Launch Jupyter Notebook and open file ...\lts2021_snow_damage_forest.ipynb. Run cells as necessary

[1]

```
#reading in SNAP-preprocessed stack of 12 Sentinel-1 dual-pol images
src=rasterio.open(os.path.join(data_path, '\\eo_lts2021\\Sentinel_1_stack.tif'),'r')
s1data = src.read()

#reading reference data
src=rasterio.open(os.path.join(data_path, '\\eo_lts2021\\eo_training_set.tif'),'r')
train = src.read()
src=rasterio.open(os.path.join(data_path, '\\eo_lts2021\\eo_testing_set.tif'),'r')
test = src.read()
```

[2]

```
#calculating data-table for further processing (can be skipped if saved data available)
#training data - first 100 stands belong to snow-damaged areas, further 100 stands represent non-damaged forest
dtrain = np.zeros((200,24))

for col in range(24):
     s1=s1data[col,:,:]
     for row in range(200):
         ind=(train==row+1)
         dtrain[row,col]=np.nansum(np.nansum(s1*ind))/np.sum(np.sum(ind))
np.savetxt(os.path.join(data_path,'\\eo_lts2021\\train.out'), dtrain, delimiter='\t')
```

[3]

```
#calculating data-table for further processing (can be skipped if saved data available)
#testing(accuracy assessment) data - first 100 stands belong to snow-damaged areas,
#extra 100 stands represent non-damaged forest

dtest = np.zeros((200,24))

for col in range(24):
    s1=s1data[col,:,:]
    for row in range(200):
        ind=(test==row+1)
        dtest[row,col]=np.nansum(np.nansum(s1*ind))/np.sum(np.sum(ind))
np.savetxt(os.path.join(data_path,'\\eo_lts2021\\test.out'), dtest, delimiter='\t')
```

[4]

```
#load pre-calculated data
dtrain=np.loadtxt(os.path.join(data_path,'\\eo_lts2021\\train.out'));
dtest=np.loadtxt(os.path.join(data_path,'\\eo_lts2021\\test.out'));
```

[4] loads precalculated data from [2] and [3]

Python processing (2)

Calculating stand-level features (stand-average intensity in this exercise) and preparing class-labels

```
#calculate backscatter in dB from stand-level averaged intensity
X_train=10*np.log10(dtrain[:,:24])
X_test=10*np.log10(dtest[:,:24])

#class labels for training (100 damaged (ones), and 100 nondamaged (zeros))
y_train=np.ones((200,1),dtype=int)
y_train[100:,:]=0

#class labels for testing (as above)
y_test=np.ones((200,1),dtype=int)
y_test[100:,:]=0
```

using more features generally improves classification accuracy, (Tomppo et al., 2019)

Creating processing pipeline. Consider adding and removing PCA and evaluate change in accuracy.

```
clf = make_pipeline(StandardScaler(), PCA(n_components=2), SVC(gamma='auto'))
clf.fit(X_train, np.ravel(y_train))
pred = clf.predict(X_test)
```

```
disp = plot_confusion_matrix(clf, X_test, np.ravel(y_test), cmap=plt.cm.Blues, normalize=None)
```

```
acc=accuracy_score(pred,y_test)
kappa=cohen_kappa_score(pred,y_test)
print('\nPrediction accuracy for the normal test dataset with PCA: {:.2%}'.format(acc))
print('\nKappa equals: {:.2%}'.format(kappa))
```

```
Prediction accuracy for the normal test dataset with PCA: 63.00%

Kappa equals: 26.00%
```