November 2011

Msc thesis in geomatics

IDENTIFICATION AND ANALYSIS OF POINT SCATTERERS IN TIME SERIES INSAR

Piers Titus van der Torren

TU Delft Delft University of Technology

Challenge the future

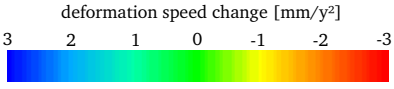
Piers Titus van der Torren: *Identification and analysis of point scatterers in time series InSAR*, Master of Science thesis in Geomatics, © November 2011

SUPERVISORS:
Ramon Hanssen
Sami Samiei Esfahany

Delft University of Technology

COVER IMAGE:
Deformation speed change rate over Delft, as measured with Persistent Scatterer InSAR (PSI) using TerraSAR-X radar images from April 2009 till November 2011. Featured in red is the construction of a train tunnel, started in 2010. See section 6.2 for more details.

Legend:



ABSTRACT

In this study methods are developed for improved analysis and processing of PSI data. PSI, or radar interferometry, makes it possible to use satellite radar images to measure deformation of the Earth's surface and objects on it with millimetre accuracy. However, interpretation of the measurements and identifying the actually measured objects is still a common problem.

There are no dedicated tools available for validation, for finding both falsely detected and falsely rejected points, or for deeper analysis of PSI results. Existing algorithms for automatic coherent scatterer selection need a lot of acquisitions to obtain reliable results, which makes it necessary to collect data for many months or years before processing can be done.

In this study a suite of tools is developed that facilitate detailed analysis of results and versatile processing of radar data. This suite consists of a visual inspection tool, and a toolbox that handles metadata and can do versatile processing of radar data.

Furthermore a method is developed for reliable point scatterer selection, that works for a small number of acquisitions, among other improvements.

CONTENTS

ACRONYMS

IRF impulse response function

SAR synthetic aperture radar

INSAR Interferometric SAR

TSINSAR Time Series InSAR

PSI Persistent Scatterer InSAR

SCR signal to clutter ratio

SSCR signal to signal+clutter ratio

GUI graphical user interface

ISLR integrated side lobe ratio

SLC single look complex

MRM mean reflectivity map

AHN2 Actueel Hoogtebestand Nederland versie 2

1 INTRODUCTION

On a lot of places on Earth the surface or objects on it move due to natural phenomena or human activities, such as earthquakes, volcano eruptions, mining, and water extraction. Often these movements are very subtle, in the order of millimetres. Even so they may be precursors to bigger events, or lead in accumulation to failure of infrastructure or houses. Measuring them is important for better understanding of the earth and can give early warnings for more serious events.

The traditional way of measuring these movements is using geodetic surveying techniques such as levelling. While those techniques are very accurate and reliable, it is very labour intensive per measured point. Because of this resolution in both space and time is usually low.

In the last decades a new technology has been researched which overcomes these problems, while allowing similar accuracy. This is achieved using satellite radar. The type of radar that is used is an imaging side looking radar, that can achieve a very high resolution using Synthetic Aperture Radar (SAR).

Satellite radar collects complex scattering signals from the Earth's surface and objects on it, which results in a radar image. The phase difference between two of such images is related to topography, deformation and atmosphere. By processing time series those parameters can be estimated. This technique is called radar interferometry (InSAR).

Radar interferometry makes it possible to use satellite radar images to measure deformation of the Earth's surface and objects on it with millimetre accuracy [Hanssen, 2001]. Since new images are continuously acquired all over the world, and have been acquired already for almost two decades, the amount of available data gives a lot of opportunities.

It has been successfully applied for measuring deformations caused by earthquakes, volcano eruptions, gas and salt mining, and for dike monitoring [Hooper et al., 2004, Ketelaar and Hanssen, 2006, Hanssen and van Leijen, 2008].

However, interpretation of the measurements and identifying the actually measured objects is still a common problem. This is caused by the nature of radar scatterers, which are the 'objects' that are measured by the radar. Opposed to geodetic surveying techniques, where a measurement is done on a known point, with imaging radar the geometry and material properties of nearby objects define the measurement. The geometry of such a scatterer can be quite complex, and can include parts which are not directly expected. For example the wall of a building and the street in front of it can together form a scatterer, of which the movement can be related to either one. This is because all radar signal that travels the same distance arrives at the same time, and hence is in the same measurement. Since this way of observing is so different from human observation interpretation errors are very probable.

Furthermore the amount of data is very large, and is ambiguous, multi-interpretable, relative, and multivariate.[1]. Also only parts of the image are usable for deformation measurement, because for the parts where the composition of a scatterer has changed one cannot speak of a single movement. When nothing in the composition of a scatterer has changed it is called coherent.

The goal of this study is to develop a method to identify coherent scatterers and analyse them in detail. For this analysis properties of the scatterers can be used, such as precise location, deformation, amplitude behaviour, and other characteristics derived from the radar signal. Those properties have to be estimated and interpreted either automatically or manually. Automatic processing is very useful for reducing the amount of data and for focusing on interesting parts. But for validation, for finding both falsely detected and falsely rejected points, or for deeper analysis, manual interpretation is necessary.

There are no dedicated tools available for this detailed analysis, making it a complicated and error prone task. Existing algorithms for automatic coherent scatterer selection need a lot of acquisitions to obtain reliable results, which makes it necessary to collect data for many months or years before processing can be done.

The primary focus will lie on making it possible to query and visualise all steps of the processing interactively and in detail. This way the raw data can be analysed in detail, just as all steps of the processing algorithms. Auxiliary data can be shown along to show possible relations. An interactive way of handling data allows getting better understanding of the problems that are accompanying time series InSAR, and analysing the origin of peculiar results.

For this a graphical user interface (GUI) is developed, featuring an overview plot and detailed information and plots of a selected point with respect to a selected reference point. From the GUI parts of the processing can be executed. To be able to show all point and radar properties and run processing steps on the fly an object-oriented toolbox will be developed, including existing and newly developed or improved algorithms.

The secondary focus is on algorithmic parts: developing improved methods for the selection of coherent point scatterers and accurate geocoding. A method is developed for reliable point scatterer selection that will work for a small number of acquisitions, among other improvements.

1 The phase of the radar signal is the part containing the deformation signal. It is ambiguous because as it is only the phase it is impossible to distinguish signals one wavelength apart. The phase is influenced by elevation, atmosphere and deformation, making multiple interpretations possible. A phase value only has meaning if it can be related to the same point at another observation, ánd another point in these observations, hence to another phase value in space and time. At last multiple properties of each point can be extracted, besides the position in the image also properties such as height, size, orientation, and behaviour in time, resulting in a multivariate dataset.

The structure of this thesis is as follows: Chapter 2 gives an overview of the theory of interferometry, radar and signal processing. Chapter 3 describes the developed tools and GUI and their possibilities. Chapter 4 introduces a new scatterer selection method and reports the results of a comparative study. Chapter 5 explains improvements in geocoding, improving accuracy, computation time, and ease of use. Chapter 6 describes some cases which show the capabilities of the tools and algorithms. The last chapter contains conclusions and recommendations.

2 RADAR SIGNAL THEORY

2.1 INTRODUCTION

This chapter is meant to give a background on the satellite radar image signal. It tries to give a deeper understanding of radar images and how to get the most information out of it. It is most useful as an introduction to chapter 4, the chapter about the selection of point scatterers, but also interesting for those who just want to understand radar images better.

For those interested in the history, synthetic aperture radar (SAR) image formation, and broader information the book on this topic by Hanssen [2001] and references in it are recommended. Ketelaar [2009] goes deeper into using time series of radar images using PSI.

This chapter is built up as follows. In section 2.2 interpreting general imaging radar is explained, section 2.3 handles the elements from which the radar image is built: scatterers. Section 2.4 explains the exact shape of those scatterers, and how that differs per satellite or processing chain. In section 2.5 the step to the phase observations and the interesting opportunities of interferometry is made, after which section 2.6 goes more in detail about describing the quality of those interferometric measurements.

2.2 RADAR

Radar observes in quite a different way than we are used to as humans. In this chapter the focus will directly be on imaging radar, the type of radar that is used for satellite based earth observation. Imaging radar produces a 2D image, at the first glance not so different from optical systems. But besides using a different wavelength which makes it look through clouds, and using its own illumination which makes it independent of the sun, there is a more structural difference. Instead of observing angles, as an eye does, it observes range. For each pixel in range the radar observes the amount of signal that is reflected back to the satellite at that range. This causes the projection to be quite different from what we are used to, and also shadows work the other way around.

Because all reflections at the same range end up in the same pixel some part are overlapping, while other parts are shadowed out, there just void is observed, see Figure 1. It is also possible for the radar signal to reflect multiple times. In that case the signal takes a longer way and will end up in the image further away in range than the reflecting objects are in reality. Imagine, on figure 1, where a ray bouncing on the top of the high building and then on the small building in front will end up in the radar image. In figure 2 a bigger part of a radar image is shown, with some interesting features, and a high amount of detail.

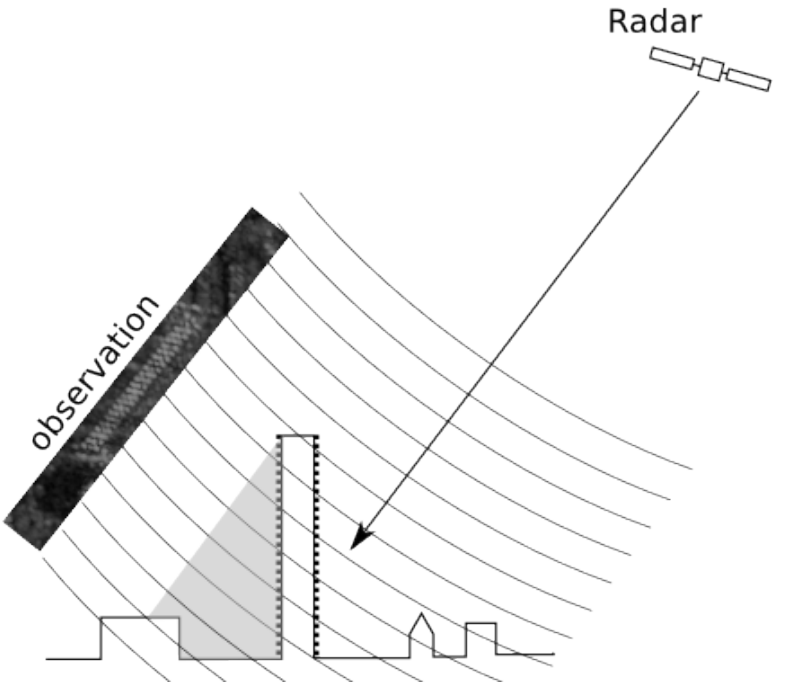


Figure 1: The way radar observes the earth. High points get overlayed on low points, and behind buildings a shadow is observed.

Figure 2: Radar image showing part of Delft University of Technology. The radar looks from the top, so the bottom of the buildings is lower in the image than the top, as you would expect in a optical picture. Shadows can be clearly seen below a number of buildings. The most prominent building, the 22 story EWI building in the centre of the image, is overlayed on top of the Mekelpark and the civil engineering building and its shadow. Note that you're looking at the 'back' side of the buildings, the side that is pointing away or up, as the radar observes from the top. Especially interesting in this view is the lack of shadow of the EWI building. While it is by far the highest building there is no shadow at all. This is because the building acts as a mirror, and the terrain and building at the radar side of the building are seen through this mirror. The signal has to travel a longer way, and as such it ends up further away in range in the image. This image is created by averaging 72 images taken between 2009 and 2011 by the TerraSAR-X satellite. It is taken with a wavelength of 16 mm, a ground resolution of 3x3 m, and an incidence angle of 24°.

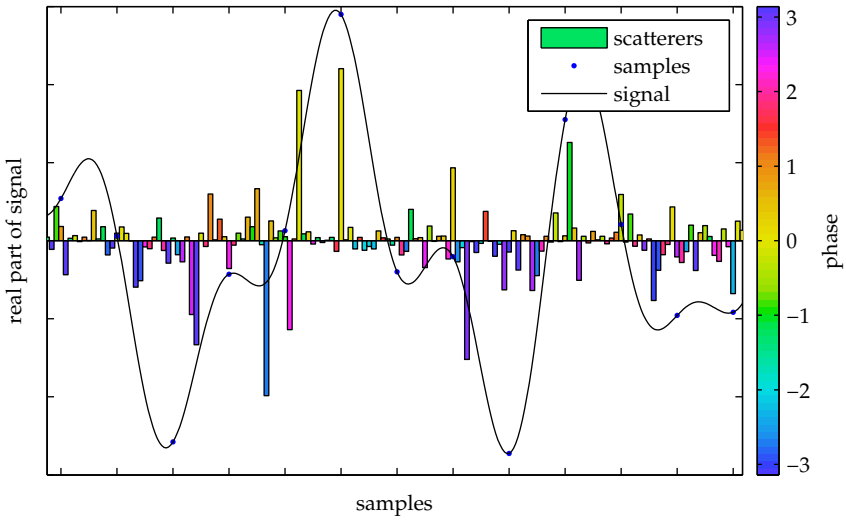


Figure 3: Scatterers, sampled signal, and fully reconstructed signal. This gives an idea of how a scattering surface could look and what signal that would give.

2.3 SCATTERERS

An object that reflects radar signal back to the satellite is called a scatterer. What can be seen as one scatterer can have a size of anything between a couple of centimetres and something very large, like a skyscraper (when acting as double bounce corner reflector).

Essentially everything that causes the radar signal to return at the radar at the same time can be part of the same scatterer.

The influence of a scatterer in a radar image extends over a resolution cell, which has a size and shape dependent on the bandwidth and spectrum of the radar signal. The shape of this influence area is the impulse response. In other words: the radar signal is a convolution of the scatterers with the impulse response.

A radar image is a sampled 2 dimensional signal, and since it is sampled above the Nyquist frequency, it is a discrete, non-aliased, signal. The signal can be perfectly reconstructed from the sampled signal. An example of such a reconstructed signal is shown in Figure 3.

Every point of the signal is built up as the sum of all reflections in a resolution cell.

When the energy is concentrated in a small area the signal represents a point scatterer. This type of scatterer is the easiest to deal with since it has defined properties: one location with one radar cross section.

This also means that a point scatterer isn't influenced by geometric decorrelation, which means that small incidence angle variations do not influence the phase.

Point scatterers generally are corners or poles, which are triple respectively double bounce scatterers. A single bounce pointscatterer will be less strong, so can only be detected when there's little signal in the surroundings.

If the energy is more distributed over the resolution cell the signal represents distributed scattering. This type of scattering is undergoing

geometric decorrelation, and also is more prone to temporal decorrelation since the signal changes as soon as only a part of the reflecting objects is changed.

The distribution of distributed scattering can be divided in three classes. It can be a 1, 2, or 3 dimensional distribution. If it is a 1D or 2D distribution the signal of observations from slightly different looking directions can be filtered to match each other. If it is a 3D distribution it can only be coherent if the looking direction is the same (it is very prone to geometric decorrelation). A point scatterer in this sense is a 0D distribution, hence a point.

2.4 THE IMPULSE RESPONSE

An impulse response is the response of a dynamic system to a very short input signal, a dirac pulse. For a point scatterer all reflected energy is concentrated at one point, hence an infinitely short signal. So the shape of the radar signal around a point scatterer equals the impulse response function of the signal. The impulse response, and hence the shape of a point scatterer, can be easily calculated if the spectrum of the signal is known, by taking the Fourier transform of the spectrum.

The spectrum of the radar image is defined by the system and processing characteristics. The most simple spectrum is a rectangular spectrum. This spectrum gives a sinc as impulse response, which has very strong sidelobes. To reduce the intensity of the sidelobes a window can be applied during processing. For SAR commonly applied windows are the raised cosine (RC) or generalized Hamming window, and the Kaiser window, which has generally better characteristics [Cumming and Wong, 2005]. The sidelobe patterns of these windows can be seen in figure 4

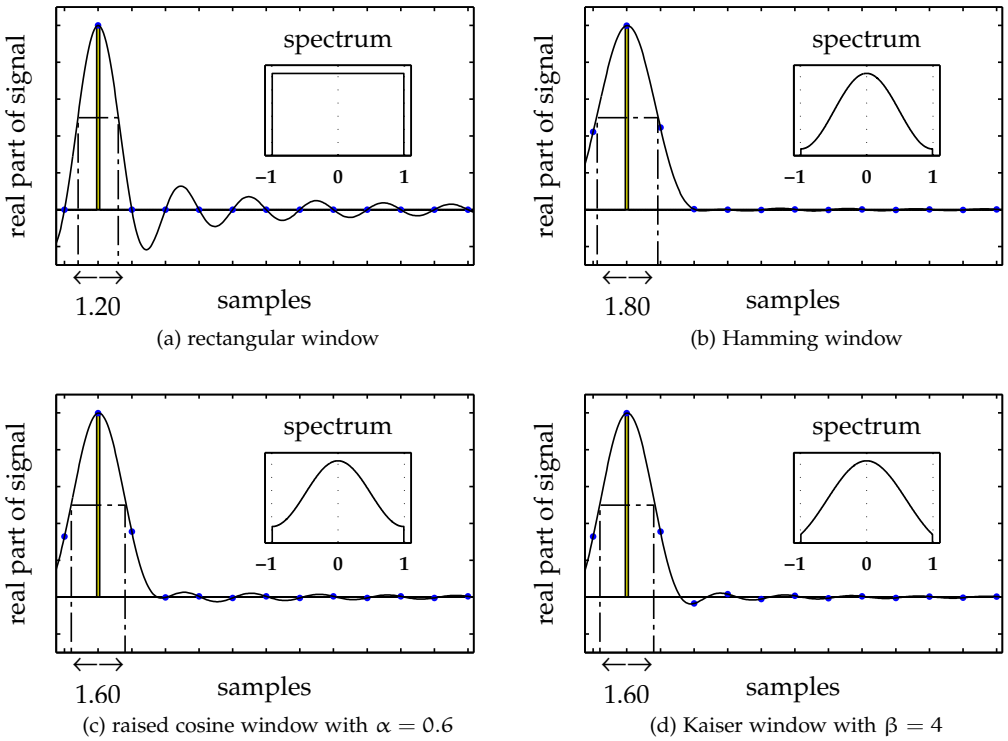
The spectrum of SAR systems is often described in the metadata. Since a SAR image is a 2D signal it has two independent spectra, one in range and one in azimuth. The spectrum in range is equal to the spectrum of the chirp, and is mostly rectangular, and in azimuth it depends on the antenna pattern. For some systems the antenna pattern is removed during processing, such as TerraSAR-X and RADARSAT, otherwise it is necessary to take it into account. The antenna pattern is a sinc^2 function [Swart, 2000, Geudtner, 1995] , which is dependent on the size of the antenna and the wavelength.

The spectra of several SAR systems can be found in table 1. These are the spectra that come from the default SAR processing for those satellites. The SAR images can be refiltered to a more optimised window as the Kaiser window [Cumming and Wong, 2005], as that will either reduce sidelobes or increase resolution, depending on the setting. This window is designed to have most energy in the main lobe for a given resolution related parameter. This parameter can be chosen to balance resolution and sidelobe suppression. A Kaiser window with $\beta = 3$ very much resembles a window of Envisat/ERS in azimuth direction. With $\beta = 3.55$ the sidelobe ratio of Terrasar-X is matched, but with slightly higher resolution.

This spectrum shape defines the actual resolution. There is no fixed definition of resolution, but a commonly used measure is the -3dB main-

Satellite	spectrum function	window resolution	integrated sidelobe ratio ISLR dB
ASAR/ERS - range	RC $\alpha = 0.75$	1.02	-15.9
ASAR/ERS - azimuth	RC $\alpha = 0.75$ * antenna pattern	1.11	-21.3
TerraSAR-X*	RC $\alpha = 0.6$	1.19	-25.0
RADARSAT-2*	Kaiser $\beta = 2.4$	1.03	-17.9
-	Kaiser $\beta = 3.55$	1.16	-25.0
-	Kaiser $\beta = 3$	1.11	-21.3
-	Rectangular	0.89	-9.7

Table 1: Signal spectrum of default processing of various SAR systems. For comparison the resolution and integrated sidelobe ratio of some Kaiser windows are shown. *) Signal is corrected for antenna pattern during processing and has the same spectra in range and azimuth.



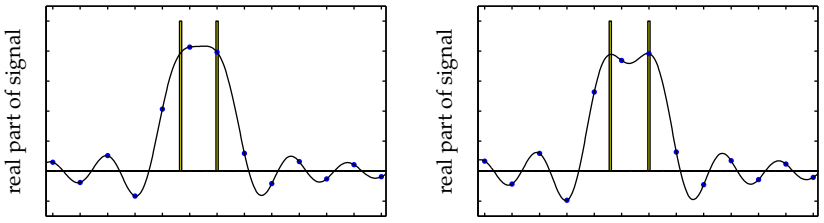
(a) rectangular window

(b) Hamming window

(c) raised cosine window with $\alpha = 0.6$

(d) Kaiser window with $\beta = 4$

Figure 4: Spectrum and impulse response of several windows. It can be clearly seen the resolution and sidelobe levels of the windows are very different.



(a) 1.0 mainlobe distance (b) 1.06 mainlobe distance

Figure 5: Resolution explained by two point scatterers: closer than 1 mainlobe width they are not separatable, further away they are separatable.

lobe width. This is the same as the distance that two point scatterers need to be apart to be separatable. This is shown in Figure 5.

2.5 INTERFEROMETRY

When the composition of some scatterers is the same in multiple observations, the phase difference between them can be directly related to the difference in signal travel time to the satellite, hence deformation is measured.

There are some more complications which must be accounted for. Because the orbit of the satellite is slightly different at each pass the height of a scatterer influences the signal travel time to the satellite. This allows the phase to be used to estimate elevation, but also makes that the elevation must be taken into account when only deformation is wanted.

When multiple acquisitions are available, and the deformation is modelled using few variables, the elevation can be estimated together with the deformation. This is further complicated by atmospheric delays, ambiguous phases, unmodelled deformations, and changing scatterers. The latter is further discussed in the next section.

2.6 SIGNAL, CLUTTER AND COHERENCE

In radar interferometry not all pixels of an image are usable. Scatterers for which the composition does significantly change between observations are not usable for interferometry, because the composition change also causes a phase change.

To describe the quality of scatterers on the basis of these changes some terms are introduced:

- signal to clutter ratio (SCR). SCR is the ratio between the power of the signal and the power of the clutter, shown in figure 6, where
  - Signal is the reflection from the scatterer. In the case of point scatterers this is a clear definition. In the case of distributed scattering it is more vague, there a useful definition is that it is the part of the signal that has a stable composition.

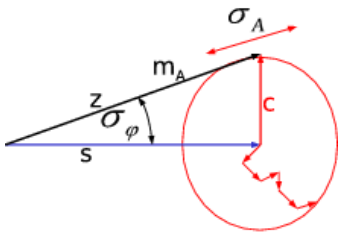


Figure 6: Signal and clutter explained. With s: signal, c: clutter, m_A: expected amplitude, σ_A: amplitude standard deviation, σ_ϕ: phase standard deviation. From Adam [2003]

  - Clutter is the rest of the signal, the unwanted part. This is the signal that comes from other objects, but ends up in the same pixel.

- Coherence. Coherence is defined for two images, as the part of the signal that has the same composition in both images. As formula for the signals y_1 and y_2:

$$\gamma = \frac{E\{y_1 y_2^*\}}{\sqrt{E\{|y_1|^2\} E\{|y_2|^2\}}} \tag{2.1}$$

So SCR is defined for one image, while coherence is defined for a pair of images. In formula form SCR becomes:

$$SCR = \frac{s^2}{c^2}. \tag{2.2}$$

where s^2 is the power of the signal and c^2 the power of the clutter.

One inconvenience is that SCR has a range of $[0, \infty]$, while coherence has a range of $[0, 1]$. To relieve us from this inconvenience the signal to signal+clutter ratio (SSCR) is defined:

$$\begin{aligned} SSCR &= \frac{s^2}{s^2 + c^2} = \frac{SCR}{SCR + 1} \\ SCR &= -\frac{SSCR}{SSCR - 1}. \end{aligned} \tag{2.3}$$

So coherence is very related to SCR, but for two images. As such it can be related to SCR or SSCR: [Just and Bamler, 1994, Hanssen, 2001]

$$\begin{aligned} \gamma &= \sqrt{SSCR_1 \cdot SSCR_2} \\ &= \frac{1}{\sqrt{\left(1 + SCR_1^{-1}\right) \cdot \left(1 + SCR_2^{-1}\right)}} \end{aligned} \tag{2.4}$$

where $SSCR_1$ and SCR_1 relate to the first image and $SSCR_2$ and SCR_2 to the second.

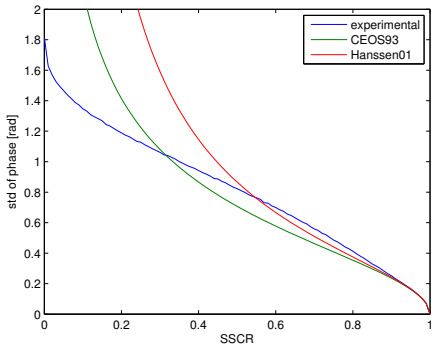


Figure 7: SCR can be related to the standard deviation of the phase error of a scatterer. Two different formulas for doing this are shown here, both only valid for SSCR close to 1. Formula 2.6 (CEOS93) and formula 2.7 (hanssen01)

Here the benefit of using SSCR can also be noticed.

Now if the SCR is the same at both observations this reduces to simply:

$$\gamma = \mathrm{SSCR}. \tag{2.5}$$

The phase error can be related to the SCR, for high SCR or SSCR close to 1 the following is valid: [CEOS, 1993]

$$\sigma_\psi \approx \frac{1}{\sqrt{2 \cdot \mathrm{SCR}}}, \sigma_\psi^2 \approx \frac{1}{2 \cdot \mathrm{SCR}} = \frac{1 - \mathrm{SSCR}}{2 \cdot \mathrm{SSCR}}. \tag{2.6}$$

Another estimate, also valid for SSCR close to 1,

$$\sigma_\psi^2 \approx \frac{1 - \mathrm{SSCR}^2}{4 \cdot \mathrm{SSCR}^2} = \frac{2 \cdot \mathrm{SCR} + 1}{4 \cdot \mathrm{SCR}^2}, \tag{2.7}$$

is directly derived from the following estimation for interferometric phase in relation to coherence: [Hanssen, 2001, 4.2.31]

$$\sigma_\phi^2 \approx \frac{1 - |\gamma|^2}{2\,|\gamma|^2} = \frac{1 - \mathrm{SSCR}^2}{2 \cdot \mathrm{SSCR}^2} = \frac{2 \cdot \mathrm{SCR} + 1}{2 \cdot \mathrm{SCR}^2}. \tag{2.8}$$

The last formula is a bit closer the the experimental values in the sensible range of $\mathrm{SSCR} > 0.5$, but both are very accurate above $\mathrm{SSCR} = 0.9$, as can be seen in figure 7.

The SCR of a scatterer can be estimated using various methods. For SCR estimation of corner reflectors on even fields the clutter level is directly estimated from a neighbourhood that is out of reach of the mainlobe and sidelobes of the scatterer. The best signal level estimation is the signal at the location of the corner reflector. See figure 8. This method is used for calibration of test corner reflectors [CEOS, 1993].

Also the commonly used metric amplitude dispersion can be related to SCR, assuming the SCR is the same in all observations. Amplitude

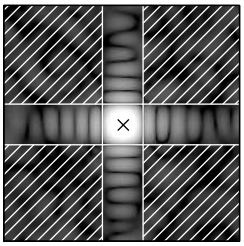


Figure 8: SCR estimation window used for calibration of corner reflectors. The hatched area is used for clutter level estimation, the cross for signal level.

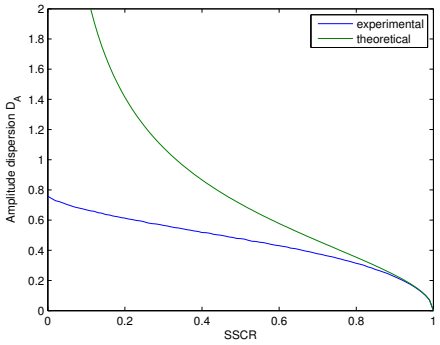


Figure 9: Amplitude dispersion related to the SCR. An experimentally derived relation and a theoretical estimation using formula 2.9 are shown. The theoretical estimation is accurate for $SSCR > 0.9$, but degrades quickly.

dispersion is calculated from a stack of images, using the standard deviation σ_A and the mean m_A of the amplitudes: [Ferretti et al., 2001]

$$D_A \triangleq \frac{\sigma_A}{m_A} = \frac{c}{s} = \frac{1}{\sqrt{2 \cdot SCR}} = \sqrt{\frac{1 - SSCR}{2 \cdot SSCR}} \tag{2.9}$$

This relation to SCR is also only valid for high SCR, as can be seen in figure 9.

The coherence between two images can be estimated too. This is done by comparing the images using a spatial window. This window is often chosen to cover an area containing a reasonable number of scatterers, which are then assumed to be stationary as a whole. This window can also be narrowed down to contain just about one scatterer, by using a weighted window which puts more weight in the centre. However, the smaller the window, the more bias will appear in the estimation [Hanssen, 2001, 4.3.4]. The formula for coherence estimation including the optional window can be constructed as follows:

$$\hat{\gamma}(t) = \frac{\int_{-k}^{k} s_1(t+u) \cdot s_2^*(t+u) \cdot w(u)\, du}{\sqrt{\int_{-k}^{k} |s_1(t+u)|^2 \cdot w(u)\, du \cdot \int_{-k}^{k} |s_2(t+u)|^2 \cdot w(u)\, du}} \tag{2.10}$$

where s_1 and s_2 are the radar images, k is the window size, and w the optional window weights. This is the 1D case, the 2D case follows logically from this.

This gives the complex coherence, and is often converted to real coherence by just taking the absolute value.

Another option is to project the complex coherence on the observation phase difference. This will cause the coherence estimate to be reduced when the interferometric phase of the observation is different from the average of the observation window.

$$|\gamma|(t) = \mathrm{Re}\left(\gamma(t) \cdot e^{-\phi(t)i}\right) \tag{2.11}$$

where $\phi(t)$ is the interferometric phase or phase difference.

This alternative conversion to real coherence in combination with a high resolution window allows coherence estimation of single scatterers, without the need for a stationary area surrounding the scatterer.

3 INSAR INVESTIGATION TOOLS

3.1 INTRODUCTION

InSAR techniques, and especially time series InSAR techniques, produce large amounts of information. The results contain thousands of scatterers, each having their own deformation pattern, and also other properties like height, shape, size, and quality measures. These properties are estimated using a complicated processing chain. During this processing decisions are made. For more insight on those processing decisions the original and intermediate data should be analysed, which increases the total amount of data involved even more.

An optimal way to effectively make this amount of information interpretable is visually. A good way to get an overview of what is happening is to plot a single parameter on a map. However, more advanced visualisation is necessary, since the information is spatially dense, multivariate, and relative.

It is spatially dense, as the covered area has more spatial detail than can be shown on a single image. A single house cannot be distinguished on an image showing the whole province. The information is multivariate, since for every point the behaviour in time is available, along with numerous other properties. Finally, the information is relative, a phase time series has no information on itself, it must be related to a reference point to contain information. Any point can be chosen as reference, and interpretability changes with this choice, so the number of visualisation possibilities is increased dramatically.

An interactive, multi-scale environment in which all this information can be visualised in multiple ways can help making this information interpretable, but is currently lacking.The current tools for time series InSAR analysis used at TU Delft are split in two steps. The radar data is first processed by DORIS, which creates coregistrated images and interferograms, after which the DORIS results are loaded into Matlab for further processing using persistent scatterer techniques.

DORIS, Delft Object-oriented Radar Interferometric Software [DORIS, 1998], is a powerful package for interferometry. However, it is intended to produce single interferograms, it has no time series analysis. It is hard to do interactive research or scripted processing, since all settings need to be passed through input files. Since it is written in C++ the barrier is high to change parts in the code, and one is limited to the available options.

Several toolboxes have been built on top of Doris for time series analysis, including Stamps, and DePSI. These are both written as a combination of shell-scripts and Matlab code. DORIS plus its successors provide a processing chain to get from single look complex (SLC) radar data to unwrapped time series.

None of those toolboxes provide an interactive environment, nor do they provide the easy interface to the metadata and calculation functions

that is needed for creating such an environment. Since all data come preprocessed from DORIS the processing functions are not available in an interactive environment, so to use different settings for filtering, oversampling, or other DORIS functions, DORIS must be called again.

The functional requirements for this environment are set as follows:

- Give easy access to all data, metadata and functions.
- Quick and interactive loading, calculating and displaying of data and metadata.
- Ability to handle large datasets faste and efficient.
- Reimplement DORIS functionality to be accessible from Matlab.
- Minimize possible error sources, such as geoid, continental drift, nonlinearity of topographic phase.
- Processing functions have sufficient precision, least possible error sources.
- Consistent coordinates, coordinates should not depend on the used crop of oversampling ratio.

In this study a suite of tools is developed that satisfies these requirements, which is described in this chapter.

First the methodology is explained, followed by sections about the graphical *InSAR Inspector* tool and the underlying object-oriented InSAR processing and metadata toolbox. The chapter concludes with a section about the outlook to the future of the tools, describing where it is used and experiments with features to be added.

3.2 METHODOLOGY

To make all data, metadata and functions easily accessible an object oriented structure is used. The available information can be seen as a big tree of objects containing the information. A scatterer is an object, which has properties and functions working with this data, just as a radar image. There are several programming languages available that are object oriented and interactive, such as Python and Matlab. For this toolbox Matlab is used, mainly because of easy reuse of existing code.

In order to assure quick loading, calculating and displaying several techniques are used, described in the following paragraphs.

Dependent data are computed ‘on the fly’. On various places data is transparently calculated when it is requested. For example when a part of an oversampled image is requested the corresponding part of the original data is loaded and oversampled to the requested oversampling level. Or when the location on earth of a scatterer is requested it is calculated using its coordinates and height. The same for phase corrected for atmosphere and topography, the atmosphere phase is interpolated, topographic phase is calculated using the height, and both are subtracted from the original phase. This way a consistent state is assured, when for example the height of a scatterer changes, the phase is automatically corrected.

Slow and complex computations are approximated by polynomials. In DORIS this is already done for some computations, like flat-earth

phase and h2ph, which are approximated by 2D polynomials. Here this is taken a step further: 3D polynomials are used for topographic phase and its derivative h2ph, so their dependency on height is taken into account. Also for geocoding 3D polynomials are used, both for $[X, Y, Z]$ and $[\mathtt{lat}, \mathtt{lon}, \mathtt{height}]$ coordinates. These polynomials are used in dependent data and evaluated transparently.

For handling large datasets all necessary data is stored on disk, and loaded only when needed.

3.3 INSAR INSPECTOR

The InSAR Inspector is a GUI that provides a way to make the spatially dense, multivariate, and relative information that is involved in InSAR processing fully explorable.

An overview map is shown at the main screen, on which all scatterers are shown. When a scatterer is selected, either on this overview map or using a unique point ID, all information about a scatterer is shown, both as numerical data and using plots. All information that is added during processing is shown, such as quality measures, height, and deformation. Information that varies per acquisition is plotted, such as amplitude or deformation. For deeper inspection several options are available, such as showing the underlying radar image data, calculating a coherence matrix, and data access from the command line.

The base components for the GUI are:

- Zoomable background.
- Phase time series plot.
- Point information table.
- Additional information plot.

The state and contained information is exposed to the programming environment as an object. This way it is possible to interact with the GUI in a programmatic way, and do custom processing on information shown in the GUI.

The GUI is initiated with a PointSet. This is an object that contains all information that is necessary for operation, such as access to the radar images, the mean amplitude, and a set of selected points.

In the following sections the GUI components are described.

3.3.1 *Zoomable background*

The largest panel contains a zoomable map on which the mean amplitude of the radar images or mean reflectivity map (MRM) is shown, with the selected scatterers on top, see figure 11. The canvas can be dragged and zoomed using the mouse. On loading a medium resolution MRM is shown, a higher resolution version can be loaded when zooming in to see the full resolution details of the radar image. On the canvas the following features are shown:

- Extracted scatterers, coloured with a property. Any property can be used.

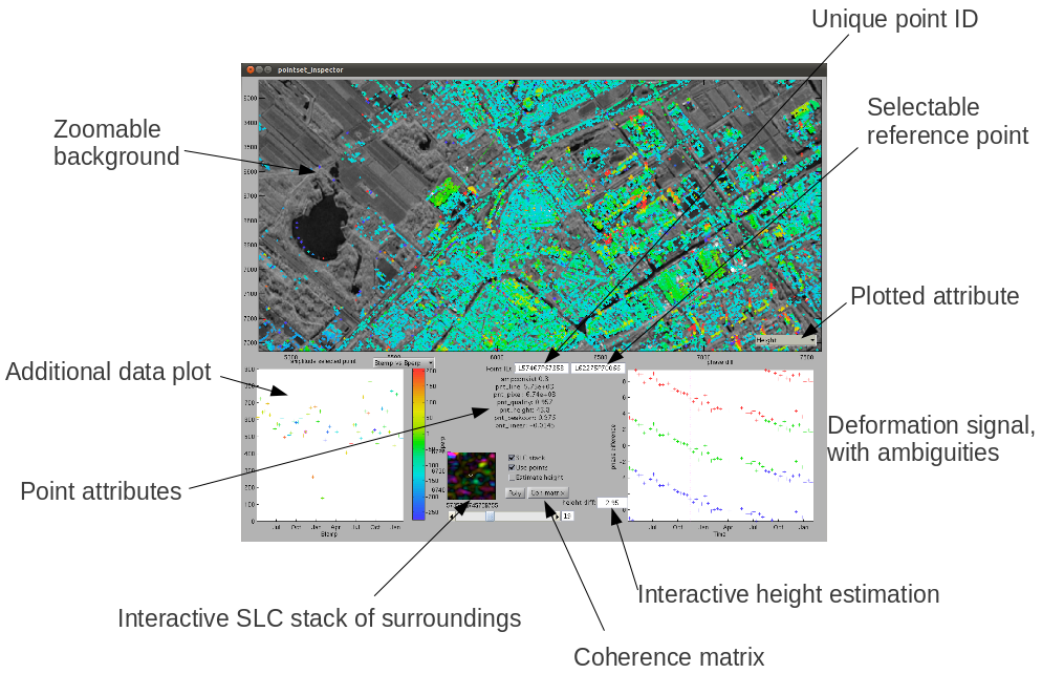


Figure 10: The elements that appear in the InSAR Inspector GUI.

- Selected point and reference point, connected by a line.
- Subpixel position of the point at each acquisition.

Using this map single scatterers can be selected. Depending on the 'use points' setting the scatterer is loaded from the PointSet, or will be loaded from the SLC images on the fly, including resampling to the clicked subpixel position. This makes it possible to examine even regions which were not selected using the automatic selection algorithms.

The reference point can be selected the same way with the right mouse button. A line to the reference point is always shown to remind the user of the fact that the shown information is relative, and which point it is relative to.

3.3.2 *Phase time series plot.*

One plot is dedicated to showing deformation. On this plot the phase of the selected point with respect to the selected reference point is shown. The phase is corrected for height and optionally for atmosphere, so only deformation is left, distorted by noise. The phase is shown in 3 ambiguities, wrapped to $[-3\pi, 3\pi)$. This shows how the data is the unwrapped and allows visual unwrapping, so the correctness of the unwrapping can be checked, see the example in figure 12.

Depending on the 'estimate height' setting the height is automatically estimated upon loading a scatterer. Otherwise the height available in the PointSet is used.

The height of the selected point can be changed interactively to manually find a correct decomposition of the phase in deformation and height difference, as shown in figure 13, or to check the validity of an automatic decomposition.

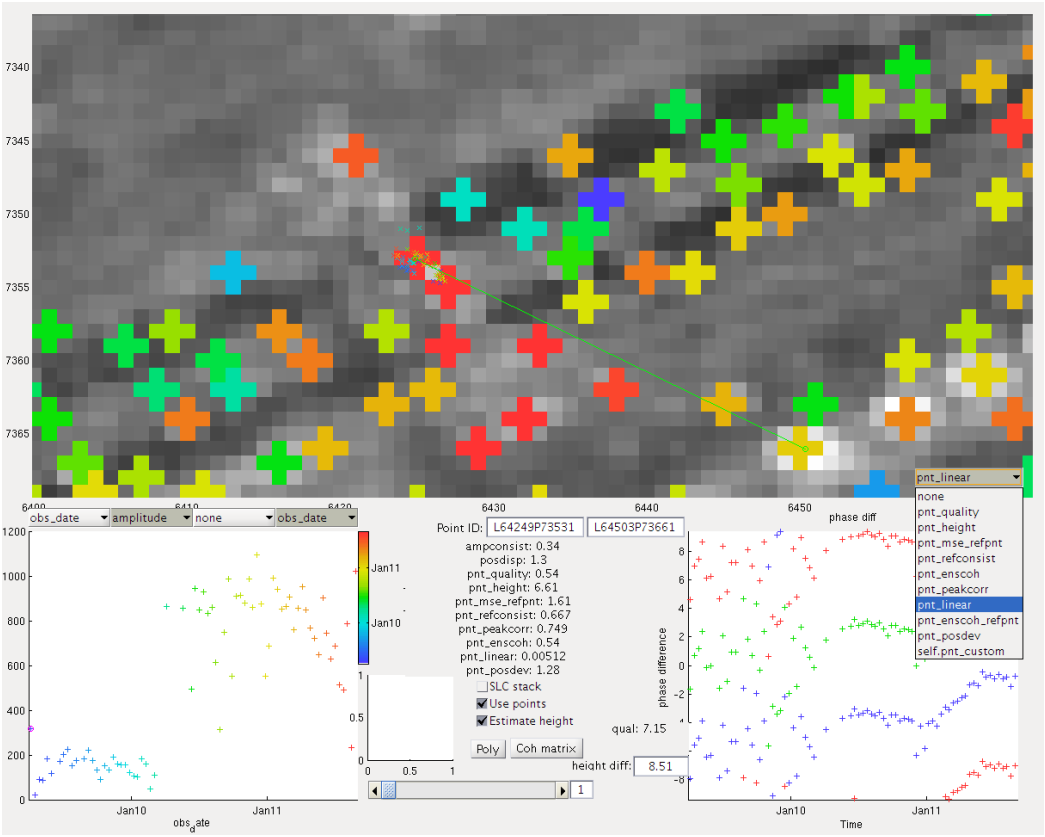


Figure 11: The zoomable background shows the selected points. The colour property can be chosen using the dropdown menu, which lists all point properties available in the pointset. The green line connects the selected point to the reference point, here a point on a bridge built in 2010 is shown with as reference a stable point on the ground. The amplitude peaks at each acquisition are shown, and show that the position was less stable before the bridge was built, as expected.

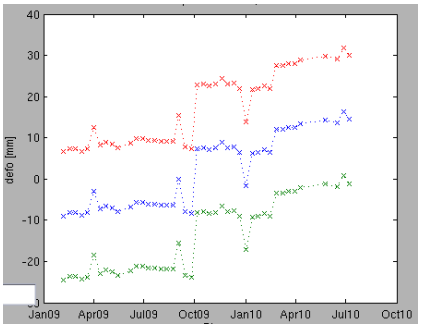


Figure 12: Automatic unwrapping clearly makes a mistake in the case shown here. Points with a suspicious linear deformation velocity can be inspected and errors can be tracked down.

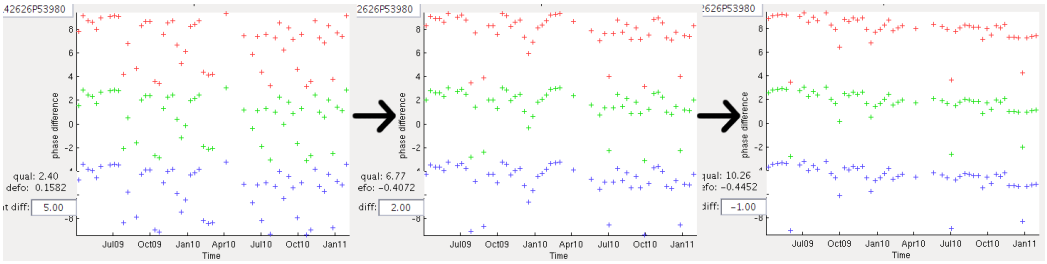


Figure 13: The phase shown in the plot is corrected for topographic phase. If the amount of topographic phase is wrong, the height can be changed until the residual phase is plausible to be deformation. By changing the height smoothly by sliding the mouse this can be done very quick.

3.3.3 *Point information table*

The point information table gives a quick overview of the point properties extracted during processing. All point properties that are available in the PointSet are shown.

The table starts with the point ID, a unique identifier based on the radar coordinates. This doubles as input field to quickly return to a previously selected point. This way interesting points can be written down and found back later on.

Following are all numerical point properties, such as the height, deformation rate and ensemble coherence. Only properties with one value per scatterer are shown, other properties, which have one value per acquisition, can be shown in the additional information plot.

3.3.4 *Additional information plot*

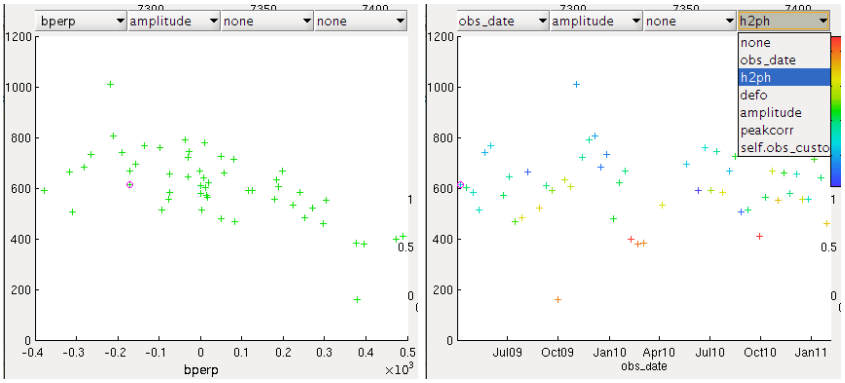
This plot is intended to show the relation between multiple properties. For example the amplitude behaviour versus time or perpendicular baseline can be seen, or by using colour even both together. Each of the x, y and colour axes of this plot can be set to a property containing time series information, such as date, amplitude or perpendicular baseline. All PointSet properties containing time series information can be chosen for any of the axes. See figure 14 for examples.

Time selection slider

Using this slider the currently selected radar acquisition can be changed. This selection is shown in the phase plot, additional plot, and subpixel position plot on the background. It also determines the displayed image of the SLC stack. This way changes in any of those properties can be related.

Settings

Some frequently used settings are shown in the GUI, other settings need to be accessed through the GUI object via the command line. The GUI contains toggles for:



(a) (b)

Figure 14: The additional information plot is used for finding a relation between properties. a) perpendicular baseline shows a clear relation with amplitude, which is useful for determining the scatterer size. b) Multiple relations can be shown together by using colour or size in addition to the x-, and y-axis.

- SLC stack: Toggles loading of the SLC data of the surroundings of the selected point.
- Use points: Toggles between preselected points and raw data.
- Estimate height: Toggles automatic height estimation.

3.3.5 *The GUI object*

The GUI object gives access to the selected points and all their properties, and also to all GUI elements and display functions. Interaction via the command line is necessary when more flexibility or more user input is required than can be given through the GUI. For example to show a custom property on the background, make high quality plots of selected points, or estimate new parameters or relations. See appendix A for the implementation details and examples.

3.4 OBJECT ORIENTED INSAR TOOLBOX

The information that is shown in the GUI, and also is used while processing or post-processing, needs to be stored in an accessible way. Functions, which are also used by the GUI and processing routines, must be able to access this information too, so it doesn't have to be supplied manually. All this is done in an object oriented structure. All information is stored in objects, and a lot of processing functions are added to the object classes, such as algorithms for radar image processing, scatterer selection, and geocoding.

The toolbox is able to perform the following actions:

- Load metadata from DORIS .res files
- Calculate metadata dependent properties
- Load SLC data with on the fly oversampling and filtering

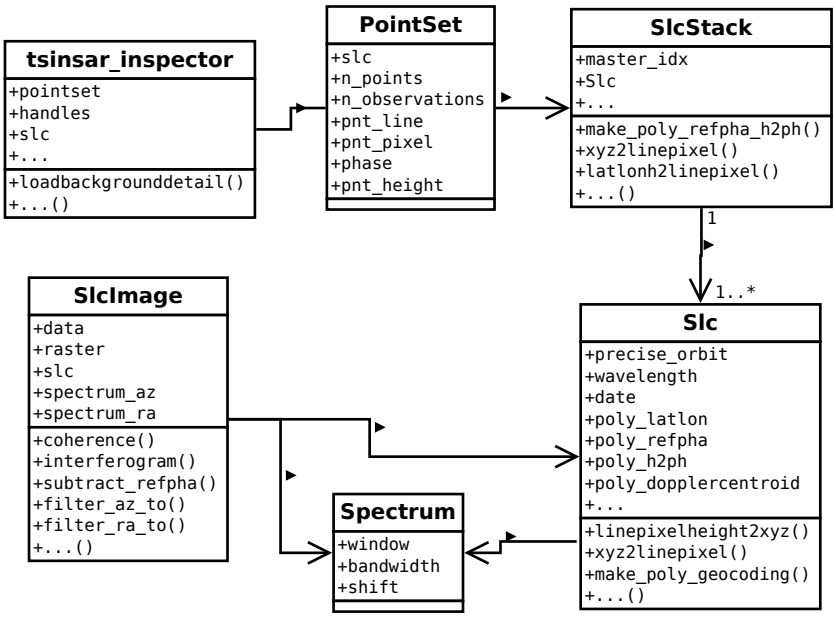


Figure 15: The most important classes and their connections are shown in an UML Class diagram. This shows for each class the properties and functions, and how they relate to each other.

- Store processing results and intermediate steps in a clear way
- Do radar image processing, such as spectral filtering, reference phase subtraction, interferogram creation and coherence calculation
- Do scatterer selection using existing and newly developed algorithms
- Work with sub pixel precision

All functions and data are grouped in several classes of objects, which use each other in order to break down everything in clear parts. The structure of the toolbox can be seen in figure 15 as a UML class diagram, showing the most important classes, properties, and functions. Several other smaller classes are used, such as a 3D polynomial class for the storage and evaluation of 3D polynomials, a raster class for the storage and transformation of coordinates of gridded data, and a disk storage class to store and read data on disk transparently.

In the following sections the classes and their functionality are explained.

3.4.1 *Slc and SlcStack*

In an Slc object all metadata of a radar image is stored. This includes the observation date, satellite orbit, and sensor and signal parameters, but also the location and format of the files containing the radar image data.

Furthermore it provides functions to access the SLC data, and functions to calculate metadata dependent information. In this category there is orbit dependent information, such as satellite velocity and looking

angle, but also pixel spacing and resolution, which depend on the signal parameters. Also functions for geocoding and inverse geocoding are included, which transform given coordinates using orbit and signal parameters.

The information is imported from DORIS result files, complemented with some satellite specific information from a small database, which is not available in the result files.

The SlcStack is mainly a container for a stack of Slc objects. It provides convenient access to properties that are either the same for the whole stack, or most often needed from only the master. These are mostly properties that are related to the coordinates, for example geocoding or ground resolution.

To use a stack of radar images the radar data should be coregistrated and resampled using DORIS, so from that moment the only the master coordinates have to be used. Any following DORIS processing steps, such as interferogram creation or topographic phase correction, are not used, as those steps can be performed more flexible using the toolbox. Oversampling in DORIS is allowed, the coordinates are automatically converted back to not oversampled coordinates, but it is not recommended due to the big increase in disk usage.

Topographic phase computation is done using the SlcStack, since the master SLC is necessary for this.

Geocoding and topographic phase computation is done using 3D polynomials. This way it is very fast even for millions of points, and an otherwise computationally expensive geoid model can be included at no extra cost. In chapter 5.2 the theory and implementation of the computation of these polynomials is discussed.

3.4.2 *SlcImage*

An SlcImage object holds SLC data in memory, together with underlying metadata. It is used to do the actual processing that is done on image level, before points are selected.

The metadata that is stored is necessary to relate the data with other data, and for a number of processing functions. The following metadata is stored:

- The crop position. This gives the coordinates and pixel spacing, in master coordinates.
- Processing state. Things such as whether or not topographic phase is subtracted, and the current signal characteristics in the form of signal spectra.
- The corresponding Slc object.

Many processing functions are implemented in this class. The most important of these processing functions are:

- Oversampling. Since the radar signal is a continuous defined it can be perfectly oversampled. This function takes into account the spectral shift, it adjusts the oversampling filter to match the spectral shift of the data.

- Spectral filtering. Spectral filtering is a very powerful tool to get more out of the radar data. It can be used to reduce sidelobe levels or increase resolution (chapter 2). Another use is to match the spectra of two radar images to increase the coherence. This can be done both in azimuth direction to correct for doppler shift, or in range direction to correct for incidence angle difference.
- Interferogram computation. Compute the phase difference between two or more images. Optionally compute coherence and use that as amplitude.
- Coherence estimation. Estimate the coherence between two images, either using a given window, or using a window that is optimised for high resolution coherence estimation.
- IRF correlation. New method for point scatterer selection, details can be found in chapter 4.
- Peak selection. Find the local maxima and determine their positions with sub pixel precision. This comes in two flavours, one that finds all peaks, and one that finds the nearest peaks for a given list of coordinates. This peak selection is part of the scatterer selection described in chapter 4.

SlcImages can be used in a stack, in which case the function is performed on all images, or if there is functionality specifically for stacks that is used. This is the case for interferogram computation or some forms of spectral filtering.

3.4.3 *PointSet*

The PointSet is a database with point properties of points that are extracted from a stack of SLCs. There are several types of properties:

- Base properties that are necessary for correct operation. This is the case for the point coordinates, phase time series and observation dates.
- Dependent properties. These are calculated on the fly from the base properties, and can be used just as normal properties. Some of these properties are:
  - Topographic phase. Calculated using the point coordinates and the 3D polynomial from the SlcStack.
  - Corrected phase. Scatterer phase minus the topographic phase, and optionally the atmospheric phase.
  - Deformation. Corrected phase minus the reference point phase, converted to meters.
  - Geocoded coordinates. Latitude and longtitude calculated using the point coordinates and a 3D polynomial from the SlcStack.
- Custom properties. Any custom property can be added during processing, or afterwards. Examples of such properties are deformation velocity, quality estimations, coordinates per observation and temperature time series.

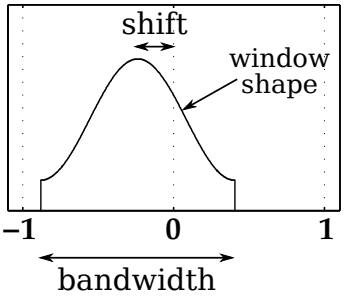


Figure 16: A normalised spectrum is defined with three properties: window shape, bandwidth, and center frequency shift.

All of those properties are either:

- One value per point, such as point height, or point quality measures.
- One value per observation per point, such as phase or deformation.
- Only one value per observation, being the same for all points. Such are observation date or temperature.

Furthermore a place for all metadata is provided, which can hold any kind of data. Here things such as a mean amplitude image, estimated atmosphere or reference point are stored.

Since a PointSet might store a large amount of data, any data can be stored on disk an only be loaded when necessary. This makes it possible to store millions of points without running out of memory. This is accomplished using the hdf5prop class, which makes properties stored on disk usable just as normal variables.

3.4.4 *Spectrum*

For processing radar images optimally the signal properties are necessary. Most of the radar signal properties are defined by its spectrum, such as the impulse response, resolution and spectral shift. For processing data only the normalised spectrum is important, which relative to the sampling rate, and so always goes from -1 to 1. This normalised spectrum is stored in this object. The absolute frequencies and bandwidth can be found in the Slc object, so can also easily be accessed when necessary.

The spectrum is defined by the following properties:

- Window shape. Defined as a polyline.
- Bandwidth. Relative to the full bandwidth, so a value in the range (0,1).
- Centre frequency shift. When the spectrum is not zero centred, as with zero doppler processed SAR images, this property defines the shift.

The spectrum class is used to represent the actual state of a radar image. This might be the original spectrum, or a modified spectrum due to filtering or oversampling.

It contains several functions to filter data to a new spectrum. Also functions are included which calculate spectrum dependent properties such as the impulse response, sidelobe positions and mainlobe width.

3.4.5 *Other classes*

Furthermore some other useful classes are included, which are more general and are used on various places. These are described below.

Poly3D is a class that stores a 3D polynomial, and includes a function to efficiently evaluate it.

A *Raster* object stores the range and step size of gridded data, and several functions to make intersections, subdivisions or convert between coordinates and indices.

RasterData contains gridded data, and the corresponding coordinates in a Raster object. It is similar to *SlcImage*, and contain the processing functions of *SlcImage* which also can be done on general images, such as peak selection or gaussian blur.

hdf5prop is a class which handles very large arrays by reading and writing data on disk transparently. Those variables can be used and indexed just as normal variables. The data is stored in a **HDF5!** (HDF5!) file, which is designed for storing large amounts scientific data in a fast an structured way.

dependentprop makes dependent properties in a *PointSet* act as normal variables. It contains a reference to a function which calculates the desired property.

3.5 OUTLOOK

The toolbox described in this chapter is in a very usable and complete state, and is now mainly in use at Hansje Brinker.

The DePSI persistent scatterer software has been rewritten around the toolbox classes, resulting in a modular and much more flexible system.

However, still some things could be improved.

Both on the pro- and postprocessing sides of the InSAR production chain there are improvements foreseen. On the pre-processing side, the current implementation of the software is very flexible in its deling with several InSAR implementation chains (focussing, resampling, cropping, and coregistration can be done through different currently used recipies, such as used by StaMPS, the DUT implementation of DePSI, and others, all of which are based on the DORIS software). An initial implementation in python, to replace these rather complex implementations has been developed and is currently used at Hansje Brinker. This preprocessing chain however cannot yet deal with raw imagery.

On the postprocessing side, several visualisation and reporting tools were developed, but these have not yet reached maturity. Examples of these are the pdf-maker which creates standardized reports, several plotting routines.

There are also several future projects that may improve the selection and characterisation of the point measurements. Below, a few of these future improvements are outlined.

3.5.1 *GIS Integration*

Precise geolocation and integration with external data sources is best served through seamless integration with existing geographical information systems in place at the parties using the InSAR results. Currently, exports towards Google Earth, ESRI Shapefiles, and simple text documents has been developed.

3.5.2 *3D AHN2 viewer*

One noteworthy feature that couldn't be integrated due to MATLAB limitations is integration with a 3D pointcloud viewer, with the main purpose of relating the scatterers to AHN2 data. Some tests are described here.

To identify the precise location of point scatterers they must be related to auxiliary real world information. This serves both as a validation that the point is representing a real object, and for deeper inspection of the scattering origin. This auxiliary data might be 2D maps, such as topographic maps or aerial photos, but even better 3D information. The best 3D information that is available is the Actueel Hoogtebestand Nederland versie 2 (AHN2), a very detailed and accurate laserscan of the Netherlands.

Since AHN2 is very detailed and consists of billions of points, it cannot be simply visualised using of the shelf viewers. The computer graphics group of the TU Delft has developed a viewer for visualising the AHN2 point clouds, with the possibility of showing other data along.

The Persistent Scatterer InSAR (PSI) results of Delft using TerraSAR-X have been loaded in the AHN2 viewer, which gives very promising results. An impression is shown in figure 17.

3.5.3 *Bayesian Point Selection*

Automatic selection of valid observations, thereby discarding unrealistic 'coincidental' observations from e.g. water or grass lands, can strongly improve the automation of the InSAR process. The level to which this discrimination can be made, determines the number of good observations that are produced. There are several steps foreseen in the improvement of this process, one of the more promising approaches being a bayesian point selection algorithm which combines several available quality measures in an optimal way, in contrast to separate thresholds on each of those.

Figure 17: TerraSAR-X PSI results of Delft shown along AHN2 in a AHN2 viewer. The coloured dots are point scatterers, the red are from the descending pass, from a stack of 72 acquisitions; the yellow from the ascending pass, using 68 acquisitions. The descending pass looks from the lower right with an incidence angle of 24°, the ascending from the upper left with an incidence angle of 39°. The area is the northern part of the Spoorzone, around the windmill De Roos. Reflections from roofs, ground-wall corners and train rails can be identified.

4

POINT SCATTERER SELECTION

4.1 INTRODUCTION

One of the major challenges in PSInSAR is the selection of usable scatterers. In this chapter a method is presented for the scatterer selection in PSInSAR techniques, which can replace amplitude dispersion as primary selection criterion in existing processing chains.

The selection method that is currently used most is using amplitude dispersion, followed by a phase based selection criterion and sidelobe filters for the final selection.

However, this prior quality description is not always accurate enough to make a decent selection, especially with small stacks. This means the first selection contains many incoherent scatterers, so strong filtering on phase is necessary, which might discard interesting scatterers with strong unmodeled deformation. Also the location of scatters is not accurately estimated, and multiple copies of the same scatterer might be included due to sidelobes.

The goal of this study is to develop a method to select coherent scatterers with an accurate quality description and accurate location.

To accomplish this the focus is set on point scatterers. Point scatterers have a high chance of staying coherent because they are less susceptible to both temporal and geometrical decorrelation [Hanssen, 2001]. Their quality can be easily defined by means of signal to clutter ratio. Also point scatterers have one specific reflection center, and thus can be located more precisely and interpreted easier. For the detection of point scatterers the signal characteristics are extensively used to compare the radar signal with a reconstructed ideal point scatterer, using a spatial window.

In section 2 the algorithm and implementation is discussed. Section 3 shows the results of IRF correlation by means of examples of processed real data, and compared to amplitude dispersion. Section 4 relates the developed measure to the quantifyable measure SCR, supported by simulation results.

4.2 ALGORITHM

The quality of a point scatterer can be described by its signal to clutter ratio. For the separation of scatterer signal and clutter signal the correlation between the radar signal and a reconstructed ideal point scatterer signal is calculated. This ideal signal is equal to the impulse response of the radar system. A weighted window is used to only include the local neighbourhood in the calculation and obtain a high resolution. The weighting function defines the estimation resolution and accuracy, where a tradeof between those must be found. A wider window can

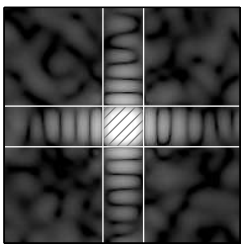


Figure 18: Estimation window. Only the main lobe is used for SCR estimation.

have higher accuracy, but has a lower resolution. The window is chosen so that only the mainlobe is included, as shown in figure 18.

The complex impulse response function (IRF) correlation at position x is defined in 1D as:

$$\rho_{irf_{cpx}}(x) = \frac{\int_{-\infty}^{\infty} s(x+u) \cdot f^*(u) \cdot w(u)\, du}{\sqrt{\int_{-\infty}^{\infty} |s(x+u)|^2 \cdot w(u)\, du \cdot \int_{-\infty}^{\infty} |f(u)|^2 \cdot w(u)\, du}} \tag{4.1}$$

where $s(x)$ is the signal at position x, f is the IRF and f^* its complex conjugate, and w is the weighting function.

This equation is constructed by taking the (cross) correlation function of the signals s and f [Cumming and Wong, 2005, p.23]:

$$\begin{aligned} \rho(x) &= \int_{-\infty}^{\infty} s(u) \cdot f^*(u-x)\, du \\ &= \int_{-\infty}^{\infty} s(x+u) \cdot f^*(u)\, du \end{aligned} \tag{4.2}$$

This is then normalised by scaling with the maximum possible value, just as it is done in coherence calculation:

$$\rho(x) = \frac{\int_{-\infty}^{\infty} s(x+u) \cdot f^*(u)\, du}{\sqrt{\int_{-\infty}^{\infty} |s(x+u)|^2\, du \cdot \int_{-\infty}^{\infty} |f(u)|^2\, du}} \tag{4.3}$$

To this equation weight term w is added that is aligned with the IRF. For the weighting function the power of the absolute value of the impulse response $w(u) = |f(u)|^2$ or $w(u) = |f(u)|$ is used. This way the weight is zero on places where nearby scatterers have no influence, and high where scatterers have more influence. In other words: it equals the amount of influence of other scatterers at each distance from the point scatterer. These windows gives good performance in resolution and accuracy. Other window function may shift the balance between resolution or accuracy.

The result is a complex correlation. For a real valued correlation, signifying the correlation of the signal with an ideal point scatterer with same phase as the signal, this complex correlation is projected on the signal phase:

$$\rho_{irf}(x) = \mathrm{Re}\left(\rho_{irf_{cpx}}(x) \cdot \frac{s^*(x)}{|s(x)|}\right). \tag{4.4}$$

This is an alternative to taking the absolute value, this way the value of ρ_{irf} is reduced when the signal phase differs from the average phase of the estimation window, which is the case when the scatterer is influenced by clutter. The value of ρ_{irf} can become zero or negative when the phase difference is very large.

4.2.1 *Implementation*

The given functions are in 1D, in reality the 2D case is used, taking both azimuth and range direction into account.

These functions are defined in the continuous domain, because the radar signal is a continuous signal. For evaluation of the functions a sampled signal is used. For sufficient accuracy the signal is oversampled at least 2 times. It can be implemented efficiently using two convolutions in both azimuth and range direction, respectively. The impulse response is truncated to the main lobe, which has little influence since the weighting values are very low outside the main lobe. With 2 times oversampling this results in a window size of about 5 pixels, which is small enough to be fast in computation.

4.2.2 *Stack processing*

For estimating the quality of a persistent point scatterer in a stack of radar images the IRF correlation of all images is computed separately, and then averaged. This results in an improved estimate, assuming the point scatterer is persistent. This way scatterers with a high mean IRF correlation are high quality and persistent all the time, while a lower mean IRF correlation means that it is either high quality and partially persistent, or lower quality but persistent throughout. By looking at the IRF correlation timeseries it is possible to discriminate between those cases.

The amplitude behaviour is not taken into account. This makes it independent from amplitude changes between acquisitions. Since that is the only information that is used for quality estimation using amplitude dispersion it is a completely independent quality measure, making it possible to use amplitude dispersion as an additional measure for an improved quality estimate.

4.2.3 *Subpixel position*

The center of a point scatterer has the highest IRF correlation, so by only taking local maxima into account it is possible to select only one point for a scatterer, and determine it's position with subpixel position. The sub pixel position is estimated by fitting a second order polynomial trough the local maximum and its four neighbour pixels, and computing the maximum of the polynomial. When the image is oversampled at least 2 times this leads to a very accurate estimate.

The sub pixel position can be estimated for the whole stack at once by using the stack average, or for each image separately. In the last case the obtained information can be used for several purposes, such as

improved coregistration, detailed speckle tracking, change detection, or for an additional quality measure.

Usage as an additional quality measure can be accomplished by calculating the stability of the position as a combination of the standard deviation in azimuth and range direction. This leads to the position standard deviation:

$$\sigma_{pos} = \sqrt{\sigma_{az}^2 + \sigma_{ra}^2} \tag{4.5}$$

where σ_{az}^2 and σ_{ra}^2 are the standard deviations of the subpixel position in all of the images in azimuth and range direction, respectively.

4.3 RESULTS

In this section some results are discussed to show the strengths of the technique. The advantages of treating point scatterers different from distributed scatterers are shown. Then the accuracy as quality measurement is compared with amplitude dispersion for different stack sizes. After that promising results for change detection and quasi persistent scatterers are shown in more detail.

4.3.1 *Focus on point scatterers*

The new method is specifically designed to only select point scatterers. Point scatterers have a high chance of staying coherent because they are less susceptible to both temporal and geometrical decorrelation. Also point scatterers have one specific reflection center, and thus can be located precisely and interpreted easier.

Additionally, distributed scattering can be very valuable too, and can be added in a densification step. This way no valuable information is lost, but more difficult scatterers are added later.

4.3.2 *Single observation per scatterer*

Sidelobes are automatically discarded due to the signal characteristics. Only one observation is given per scatterer, which is located in the center of the main lobe. This removes the need for other sidelobe filters.

The separation of the main lobe and sidelobes is effective, as can be seen in figure 19, and clearly distinguishes between nearby point scatterers and sidelobes. This separation is best when the radar image is filtered with a medium resolution filter, such as a Kaiser window with $\beta \geqslant 3$. This is because the mainlobe gets wider when the resolution is lower, while the sidelobes keep the same width, as can be seen in figure 4. This reduces the similarity between the sidelobes and the main lobe, and hence the IRF correlation is reduced.

4.3.3 *More accurate quality measure for smaller stacks*

IRF correlation is an estimate for the coherence of point scatterers. To test the accuracy of this estimate a comparison is done with other

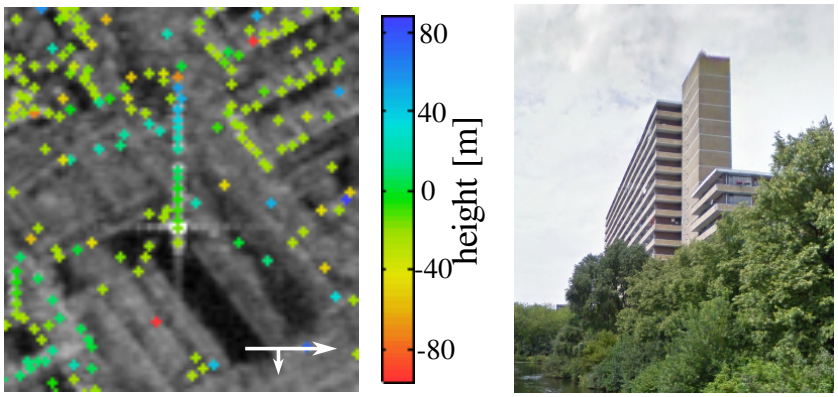


Figure 19: Point scatterers selected using IRF correlation are shown on a MRM. The sidelobes of the bright scatterer in the center are not selected, but nearby scatterers which stronger than the sidelobes are selected. In this case the strong scatterer with visible sidelobes is at the bottom of a building, while at each story above that scatterer is another scatterer. The building is the highrise at the Bosboom Tousaintplein in Delft, a 16 story building, the building has an interior corner, as can be seen on the photo, which causes strong reflections. The image is acquired by TerraSAR-X in descending orbit with a resolution of 3x3m.

quality estimates: amplitude dispersion and ensemble coherence. Amplitude dispersion is often used for primary point selection, for which IRF correlation is an alternative. Ensemble coherence is a quality estimate based on the phase behaviour, and is a measure for how well the phase behaviour adheres to a given deformation model. In this comparison it is used as reference quality, assuming it the best quality measure available.

For the comparison test a stack of 69 Envisat images covering Rotterdam, Delft and a part of the Dutch coast is used, as shown in figure 20. This area is very varied, and includes cities, sea, industry, and rural area. The stack has been processed using the DePSI software, with a full pixel processing, which means that all pixels are processed. For every pixel the local ensemble coherence is calculated, which is the ensemble coherence to a nearby reference point. A linear deformation model is used, and height is estimated in a range of 60 m. The ensemble coherence obtained this way can now be used as a reference quality for every pixel.

To test the accuracy IRF correlation and amplitude dispersion, points were selected using both methods for stack subsets of various sizes between 1 and 69 images. The used images were distributed evenly over time, to avoid a bias by temporary scatterers.

Of each of those processed sets the best 20 000, 60 000, 120 000, 200 000 and 300 000 pixels were selected, according to either amplitude dispersion or IRF correlation. For those selections stacked histograms of the ensemble coherence were made. This way the distribution between supposedly good and bad points can be easily seen. The histograms are shown in figure 21, for stack sizes of 8, 20 and 69 images.

There are several things that stand out in this comparison.

- For small stacks, up to 20 acquisitions, IRF correlation clearly selects better points, while for bigger stack the difference gets

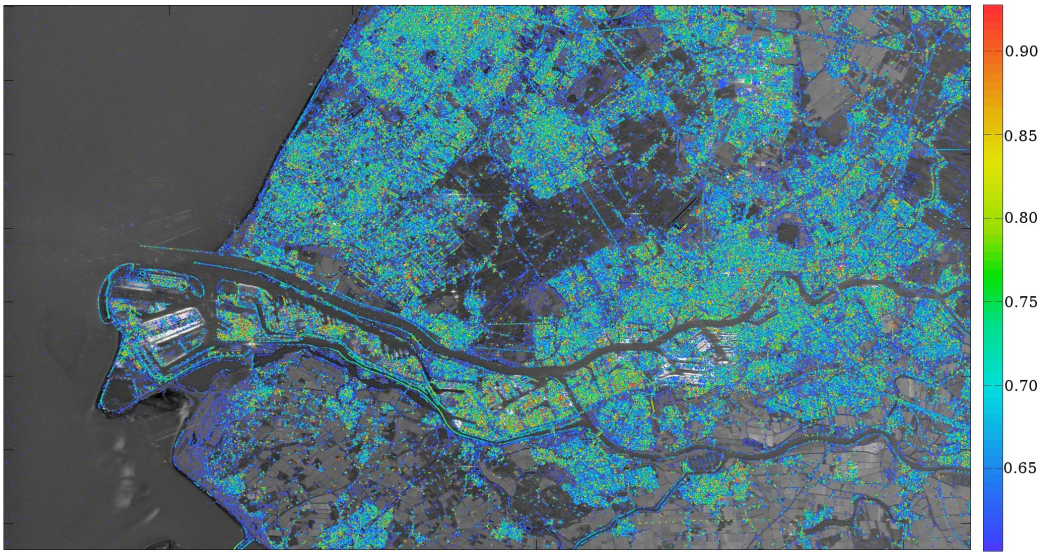


Figure 20: Overview of the processed area of 50x30 km, including the Maasvlakte, Rotterdam and Delft. The stack consists of 69 images and is acquired by Envisat in the descending pass. The colour shows IRF correlation, according to eq. (4.4), all points with $\rho_{irf} > 0.6$ are shown.

smaller. Even for a small stack of 8 images where amplitude dispersion is unusable, IRF correlation gives quite good results.

- IRF correlation selects much more points with a medium ensemble coherence. This can be points with a lot of unmodeled deformation, but more plausible are partially persistent scatterers. IRF correlation is less sensible to outliers than amplitude dispersion, so when scatterers are not present in all images they are still selected.
- IRF correlation selects a lower percentage of the points with a very high ensemble coherence, while amplitude dispersion selects almost all of them. This is because IRF correlation selects only the point scatterers, while sidelobes and distributed scatterers are left out.

The scatterers can be divided in several classes based on IRF correlation and ensemble coherence, to try to clarify the origin of those scatterers: ensemble coherence shows how linear the local deformation is, a medium value mean either that the phase is noisy, that the deformation is nonlinear, or that the deformation is linear with some noisy values, hence partially persistent. The IRF correlation value is the mean of all acquisitions, so a medium value can mean that it is medium throughout the stack, or that there are several high and several low values, which indicates a partially persistent scatterer. The classes created by this separation are shown in table 2, along with examples.

IRF correlation can already make a reliable scatterer selection for stacks much smaller than the number of images required for amplitude dispersion. Where amplitude dispersion requires at least 20 images, IRF correlation gives better results already using only 8 images. IRF correlation has higher accuracy as a quality measure for point scatterers. Combined with the explicit focus on point scatterers it gives a more discriminating measure for valid measurements. This makes it possible

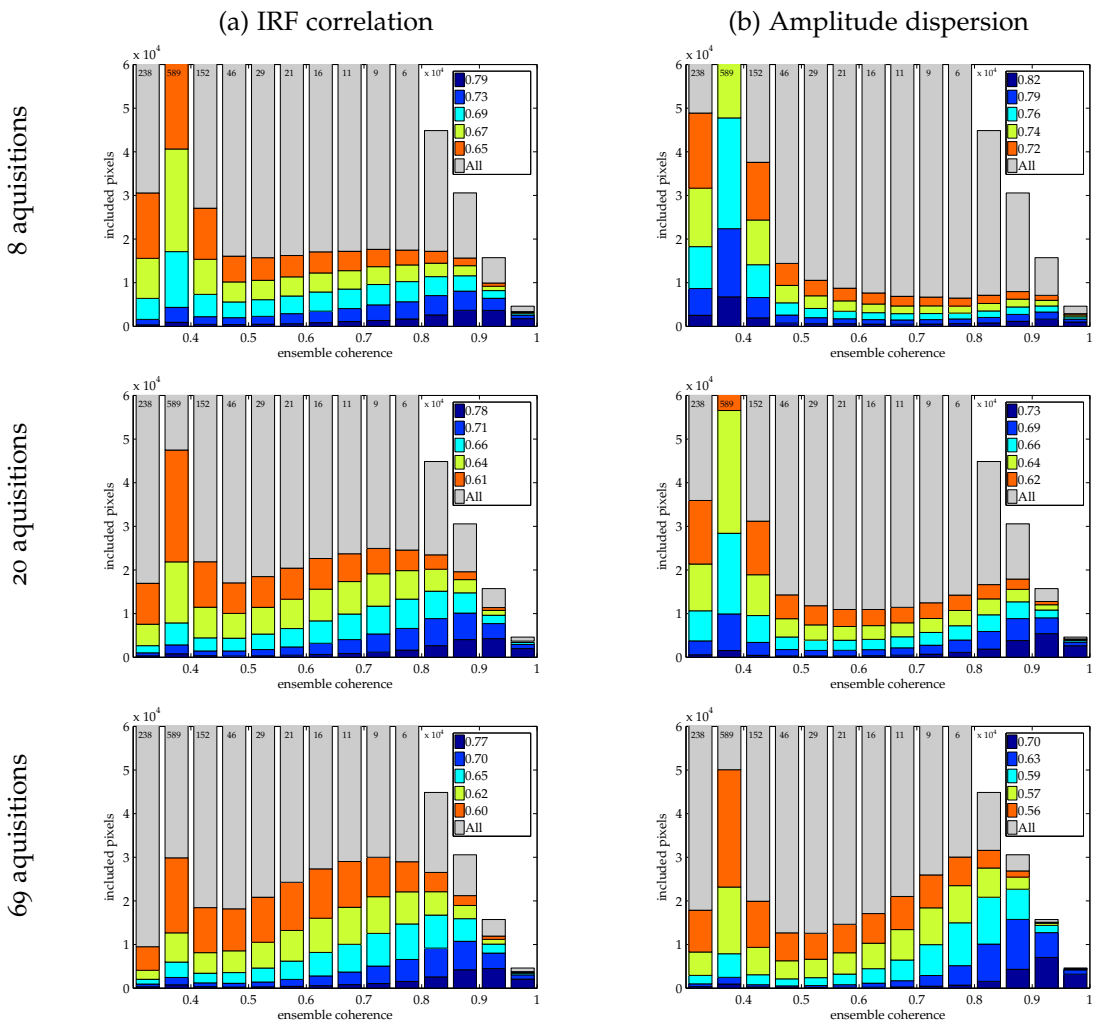


Figure 21: The accuracy of IRF correlation as quality estimate, compared with amplitude dispersion, for a stack size of 8, 20 and 69 images. For each stack size overlayed histograms of the ensemble coherence distribution are shown, with: in gray the total amount of included pixels, where it is clipped the values are shown in the bars; in blue till orange the distribution of the best 20 000, 60 000, 120 000, 200 000 and 300 000 points respectively, selected by IRF correlation or amplitude dispersion. In the legends the thresholds corresponding to those amounts are shown. (instead of amplitude dispersion amplitude consistency= $1 - D_A$ is used here). Roughly points with an ensemble coherence < 0.5 can be regarded as bad.

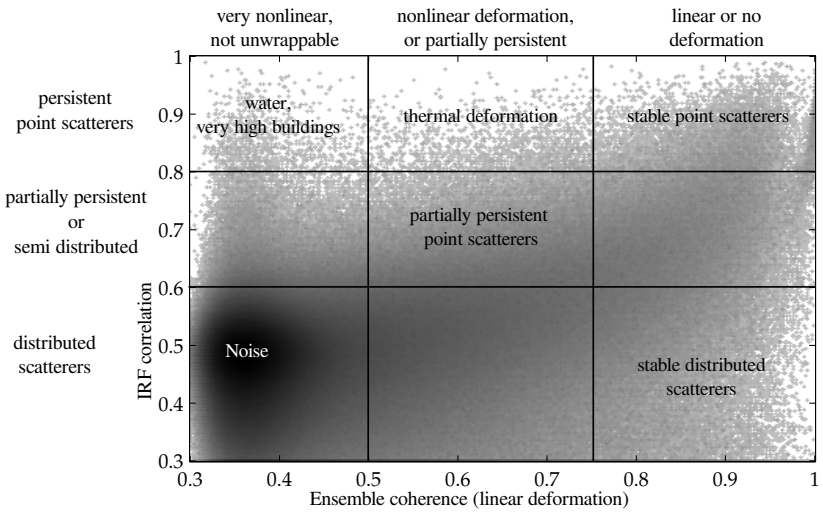


Table 2: Several scatterer classes based on IRF correllation and ensemble coherence. The classes that most prominent in the test data are shown. One example of this is water, while water surface scattering just ends up in as noise, dual bounce reflection on a water surface-wall corner gives a valid point scatterer, but moving along with the water level, which even in the Netherlands not stable on mm level.

to get same quality point selection results with smaller stacks, or better quality results with big stacks. This is especially important for monitoring campaigns, where it is a big burden to have to wait until more than 20 images are acquired, as that can easily take 2 years.

4.3.4 *Quality measure per acquisition*

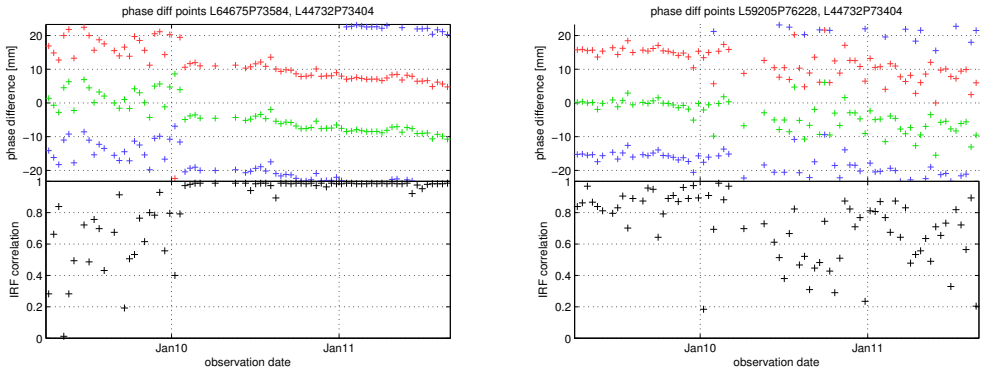
The IRF correlation is calculated for every acquisition, before being averaged to obtain the final estimate. The IRF correlation values per acquisition can be used too for change detection and to find incoherent observations in persistent point scatterers. For optimal change detection the IRF correlation per acquisition should be combined with amplitude and subpixel position per acquisition.

4.3.5 *Sub pixel precision location*

For each point scatterer the position is estimated with sub pixel precision, allowing more precise geocoding.

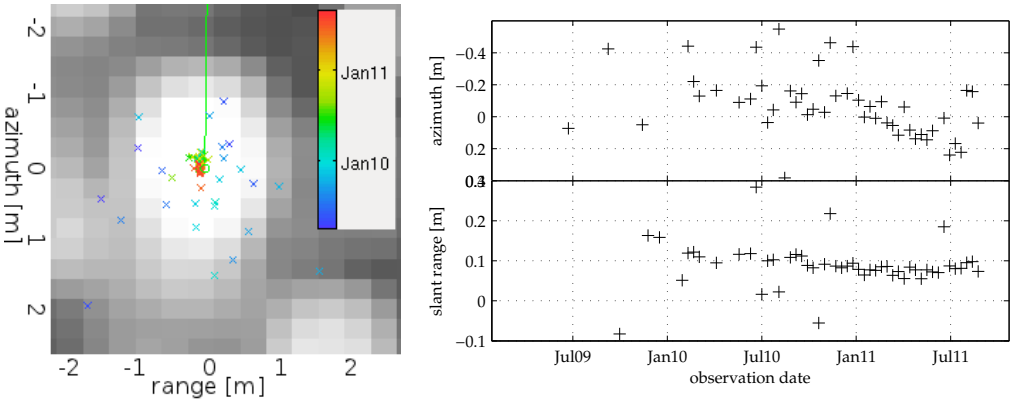
Besides this also the position can be estimated per acquisition, which can be used for an additional quality measure, for investigating bigger movements, or to improve coregistration. Figure 23 shows the estimated position through time.

The position standard deviation, as defined in equation 4.5, shows a clear correlation with ensemble coherence, as can be seen in figure 24. As such it is useful as additional quality estimate.



(a) (b)

Figure 22: IRF correlation through time can be used for change detection, to detect appearing and disappearing point scatterers. High IRF correlation values occur also during the incoherent time, this is because in a single observation random clutter has a fair chance of reflecting concentrated energy. Here two scatterers from the spoorzone Delft are shown, where construction of a train tunnel has started early 2011.



(a) (b)

Figure 23: Detected scatterer position through time for a newly appearing point scatterer in early 2011. The positions before the change are very varying, while after the change the position stays very much the same. On a closer look during the coherent period a movement in range and azimuth can be seen. This movement is about 30 cm in azimuth, to the south, and it is unclear what is causing the movement, since it seems unlikely that the point has moved so much in reality. (It is in the new bus station near Delft CS.). In slant range the movement is about 4 cm, while the deformation measured with the phase is less than 1 cm, as seen in figure 22a. These positions are all estimated relative to the coregistration, so care should be taken when interpreting, however a shift in coregistration correlated with time seems improbable.

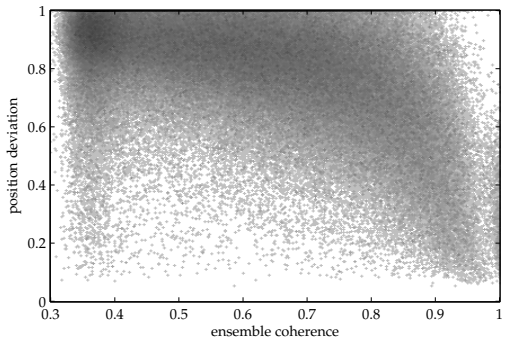


Figure 24: Position standard deviation, as defined in equation 4.5, can be used as a quality measure. In this figure the relation with ensemble coherence is shown for the stack used in 4.3.3. The problems with ensemble coherence as quality measure that are shown in table 2 are applicable here too.

4.3.6 *Amplitude calibration is not necessary*

Using IRF correlation amplitude differences between acquisitions are not used, and hence data from satellites wich suffer from unstable amplitudes, like ERS and RadarSat, can simply be used without prior amplitude calibration. Also changes in amplitude caused by other influences won't influence the selection, such as temperature, moisture, or observation geometry, which can have a significant effect, as shown by Perissin [2006]. This in contrast to amplitude dispersion, where the performance is dependent on correct amplitude calibration, making selection dependent on complicated calibration algorithms.

When the amplitude from a satellite is stable or well calibrated it is possible to use amplitude dispersion as an additional quality estimate, as they are independent measures, with amplitude dispersion only using the amplitude difference between acquisitions, and IRF correlation being insensitive to these differences.

4.4 RELATION TO SCR

The impulse response correlation can be related to the SCR. By doing that the accuracy of the phase can be quantified, independent on the actual phase behaviour. This knowledge can be very useful for various goals, such as estimation of the deformation using statistic smoothing.

The IRF correlation can be related to SCR through equation 2.4:

$$\gamma = \sqrt{SSCR_1 \cdot SSCR_2}$$

where γ can be exchanged with ρ_{irf}, $SSCR_1$ is the SSCR of the scatterer, and $SSCR_2 = 1$. This is because IRF correlation is in fact the coherence between the signal and a perfect clutter-free signal. So this leads to:

$$\begin{aligned} \rho_{irf} &= \sqrt{SSCR} \\ SSCR &= \rho_{irf}^2 \end{aligned} \tag{4.6}$$

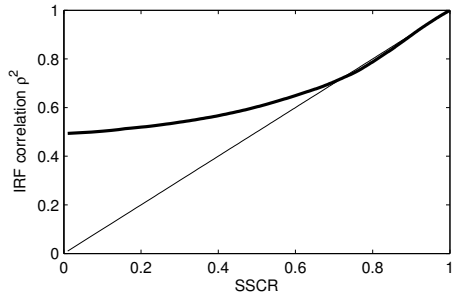


Figure 25: The relation for ρ^2_{irf} as a estimator for SSCR. It is very accurate for SSCR above 0.7, for lower values it is biased due to the small number of looks. The thick black line shows the experimentally obtained relation, the thin black line shows the theoretical relation.

To verify this relation a simulation is done to obtain ρ_{irf} for points with a known SSCR. For the simulation a 2D field of random scatterers is created, which is then resampled to a much lower sampling rate and filtered using a Kaiser window with $\beta = 3$. This image is used as clutter image, and is normalised to have a mean amplitude of 1. A point scatterer image is simulated the same way, but instead of a random field starting with a single scatterer surrounded by zeros, giving this point scatterer an amplitude of 1. The signal and clutter are then added with a given ratio. This is done 10 000 times for 200 different SCR ratios for the complete simulation. For every SCR ratio the mean ρ_{irf} of the simulations is calculated. The result is shown in figure 25.

As a estimate for SCR the impulse response correlation is very accurate for $SSCR > 0.7$, which equals to $SCR > 2.3$. For lower values it is biased. At $SSCR = 0.7$ the phase standard deviation already is 0.6 rad, as can be read from figure 7. This bias is shown to be dependent on the number of looks and the actual coherence in the work of Bamler and Hartl [1998, p.32] or Hanssen [2001, 4.3.4], and agrees with the fact that the effective number of looks in the weighted window used in the calculation is very low.

4.5 CONCLUSIONS

The new method for selection of stable point scatterers based on their theoretical shape is shown to be very effective. As a quality estimate it creates new possibilities for processing small stacks, and for large stacks improvement is shown compared to amplitude dispersion. Besides being a quality estimate it also neglects sidelobes, solving the problem of sidelobe filtering inherently. Another benefit is the subpixel precision position estimation, which is shown to be useful as quality measure and for increasing geocoding accuracy. Furthermore a quality estimate per acquisition is available, creating opportunities for change detection.

Since it could be used as a replacement for amplitude dispersion both methods are compared. The IRF correlation method differs from amplitude dispersion in the following ways:

- Only point scatterers are selected, with sub pixel precision location and without sidelobes.

- As a more accurate quality estimate it makes it possible to process smaller stacks with good results.
- A quality estimate per acquisition is available. Since the quality is estimated from the signal characteristics spatially a quality estimate can be given already for a single image. This makes it possible to do change detection and find quasi persistent scatterers.
- Amplitude calibration is not necessary since the absolute amplitude is not used for quality estimation. This has major benefits for ERS and RadarSAT.

It is also possible to use IRF correlation together with amplitude dispersion, as they are independent measures, with amplitude dispersion only using the amplitude difference between acquisitions, and IRF correlation being insensitive to these differences. The effectiveness of this combination is left for further research.

5 GEOCODING IMPROVEMENTS

5.1 INTRODUCTION

The most obvious method for identifying scatterers is by looking at their location on earth. This is a very intuitive method, and can be easily done by relating the scatterer locations to other data such as maps, aerial photos or DEMs. The scatterer locations must be estimated correctly to make this method work. The process of estimating the real location of a scatterer is geocoding.

Currently there are problems with the geocoding accuracy, resulting in shifted coordinates and suboptimal relative precision. This is mostly noticeable in cases where a high accuracy is necessary, such as building monitoring [Arroyo et al., 2009].

The goal of this chapter is to provide improvements in several processing steps that influence geocoding. These steps include the actual geocoding, which transforms radar coordinates plus height to world coordinates, and height estimation, which is the separation of the scatterer phase into topographic phase and other phase. The selection of scatterers also influences the location, but this is already covered in chapter 4.

Geocoding will be done using a 3D polynomial, which is very fast to evaluate, while the iterative geocoding methods only are used once for polynomial construction. Also the geoid height will be used as reference instead of the ellipsoid height. Geocoding is currently done with respect to the ellipsoid, while heights are mostly expressed with respect to the geoid. While transformation can be done afterwards that increases the number of steps and hence the chance of errors and the amount of necessary computation time.

For accurate height estimation the height difference must be related to topographic phase. The topographic phase is the phase difference that is caused by height difference. The way the topographic phase is calculated will be changed to minimize both structural errors and computation steps.

5.2 GEOCODING

Geocoding is done by finding a point on earth such that it corresponds to a (line, pixel) coordinate and a given height. To accomplish this efficiently a set of 3D polynomials is set up, with as input the radar coordinates and height of the scatterer. Upon evaluation this results in the (latitude, longitude), or (X, Y, Z) coordinate.

For the construction of the polynomial a 3D grid of 1000 points, covering the whole image with a height range of -500 to 9000 meter, are geocoded using the traditional iterative method. Then the height of the geoid

above the ellipsoid is calculated for the geocoded coordinates and subtracted from the original heights. After that the polynomials are fitted. While it is not obvious to take the geoid height into account in the traditional method, using a polynomial it can be done almost without computational cost.

A degree of 5 is used in order to get a high accuracy. For a 3D polynomial that means 56 values are necessary. The maximum error using this degree in the order of cm, even when the polynomials are set up for a full frame.

For inverse geocoding another set of 3D polynomials is fitted, giving the (line, pixel, height) coordinates with as input either (latitude, longitude, height), or (X, Y, Z) coordinates.

5.3 TOPOGRAPHIC PHASE COMPUTATION

Because the orbit of the satellite during each observation is slightly different, it is possible to estimate the height difference between scatterers from the phase. The phase change that is caused by the height in relation with the orbit difference is called the topographic phase.

In DORIS a separation between ellipsoid phase and the topographic phase is made. While this is useful to work stepwise towards an easier to unwrap interferogram, it is much easier to combine them when they both need to be done at once.

Calculating the topographic phase for an (line, pixel, height) coordinate is done using a 3D polynomial. This polynomial is set up using the following steps:

1. Geocode a 3D grid of 1000 points, covering the whole image with a height range of -500 to 9000 meter, using the geocoding polynomial. Extract the geoid heights from these coordinates and geocode again using the traditional method. This is necessary because the polynomial is not accurate enough for phase calculations.
2. Calculate the distance from those points to the master orbit. This is easy because the (line, pixel) coordinates are still known.
3. Calculate the distance from the points to the slave orbit, and subtract them from the distance to the master orbit.
4. This give the parallel baseline as seen from each location, this is converted to phase and the polynomial is fitted.
5. Repeat step 4 and 5 for each slave orbit.

To estimate the height difference from the phase the derivative of the topographic phase to height is used, which is sometimes called H2PH. For ease of use a separate polynomial is set up for this, by evaluating the topographic phase twice for the gridded points, with one meter difference in height. To the difference of these phases the H2PH polynomial is fit.

While creating the topographic phase polynomials optionally the satellite orbits are converted to ETRS, the European Terrestial Reference System. This corrects for deformation trends due to tectonic plate movement of Europe. As a result for a 100 km wide strip this deformation

trend is corrected with about 2 mm/y. While this reduces the mysterious trend seen in Envisat results, it doesn't explain the whole 15 mm/y that is observed.

5.4 CONCLUSIONS

Two parts which influence geolocation accuracy have been improved, one is the geocoding itself and the other deals with height estimation accuracy. For geocoding the geoid heigth is used as reference, instead of using the ellipsoid. This makes it compatible with almost all other places where height is used, like in DEM's. Height estimation accuracy is improved for big height differences, where the derivative of the topographic phase no longer can be assumed constant. The most noticeable part of the improvements however might the improved speed and ease of use.

There are a couple of other improvements possible that have been worked on but are not finished. Those are listed below.

- Orbit averaging. Calculate timing or orbit errors of the master orbit used for geocoding by using all other orbits, coregistration polynomials, and a DEM.
- New height estimation methods, decreasing the deformation model influence and increasing the density of the network.

6 CASES

6.1 INTRODUCTION

The techniques presented in this paper are beneficial for almost all InSAR processing, in this chapter several cases which benefit most are described.

The first case shown is the case that was the main reason for this research, monitoring the construction of the Delft train tunnel using InSAR.

Second, a case involving detailed monitoring of unstable storage tanks in harbours is treated, combining speckle tracking with interferometry on a very detailed scale.

At last the benefits for the corner reflector experiments undertaken by the TU Delft are shown.

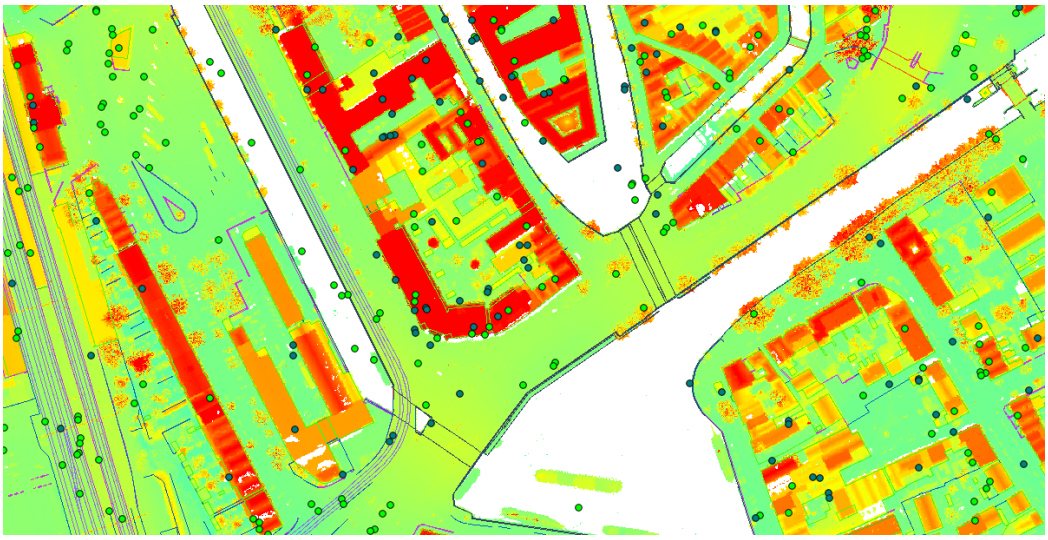
6.2 DELFT TRAIN TUNNEL

In Delft the construction of a new tunnel and underground railway station has started early 2011, which goes very close to the historical centre and a lot of personal houses. While the direct surroundings are monitored using traditional methods the monitoring data is not open to public, and anything further than the first row of buildings is not monitored at all. A feasibility study for monitoring the area using InSAR has been done in 2009 [Arroyo et al., 2009], and high resolution TerraSAR-X images are acquired since April 2009, twice every 11 days, both ascending and descending. This gives a great opportunity for such a monitoring project. The main conclusions of the feasibility study were that the geolocation accuracy of scatterers is a main bottleneck, being lower than would be expected based on the satellite image resolution, and, probably even more important, that better filtering is necessary to remove bad points and keep good points.

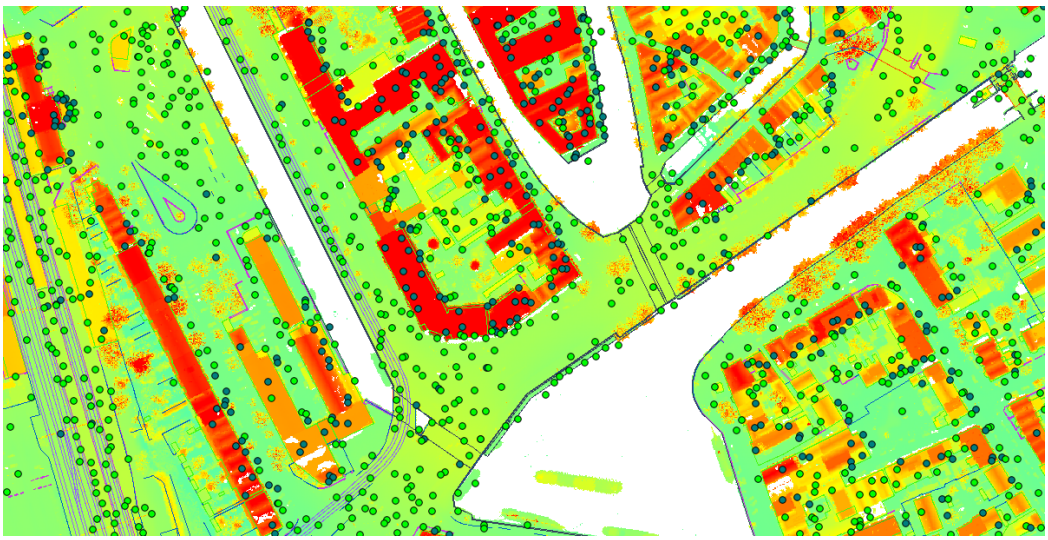
The InSAR processing is done again using the new methods for point selection and geocoding. To make a fair comparison a processing is done using only the first 16 images that were also available during the feasibility study. Using all 72 images that are available now also a processing is done, giving of course even better results, and show already the movement that is very likely caused by the tunnel construction.

The difference between the old and the new processing is very large, as can be seen in figure 26. While using amplitude dispersion with sidelobe filtering only a handful of scatterers is selected, the results of the IRF correlation method show almost continuous lines of scatterers. Also the geocoding looks better, and there are less water scatterers.

Using more acquisitions the quality of course gets better, as can be seen in figure 27, which show the results of a processed stack of 72



(a)



(b)

Figure 26: Selected scatterers shown on top of AHN2 and GBKN as very accurate reference data. Light points are below 3m NAP, while dark points are above. A stack of 16 TerraSAR-X images in descending orbit is used a) Scatterers selected using amplitude dispersion with sidelobe filtering. This is the data used in Arroyo et al. [2009]. Even with strong filtering there are points in the water. There is a shift in geocoding that is variable spacially. b) Scatterers selected using IRF correlation. The density has been improved manyfold, while no scatterers in water are selected. Clear straight lines of scatterers are visible, making it easy to relate them to buildings. There is still a spacially varying shift due to a small trend in the height estimation which has to be removed.

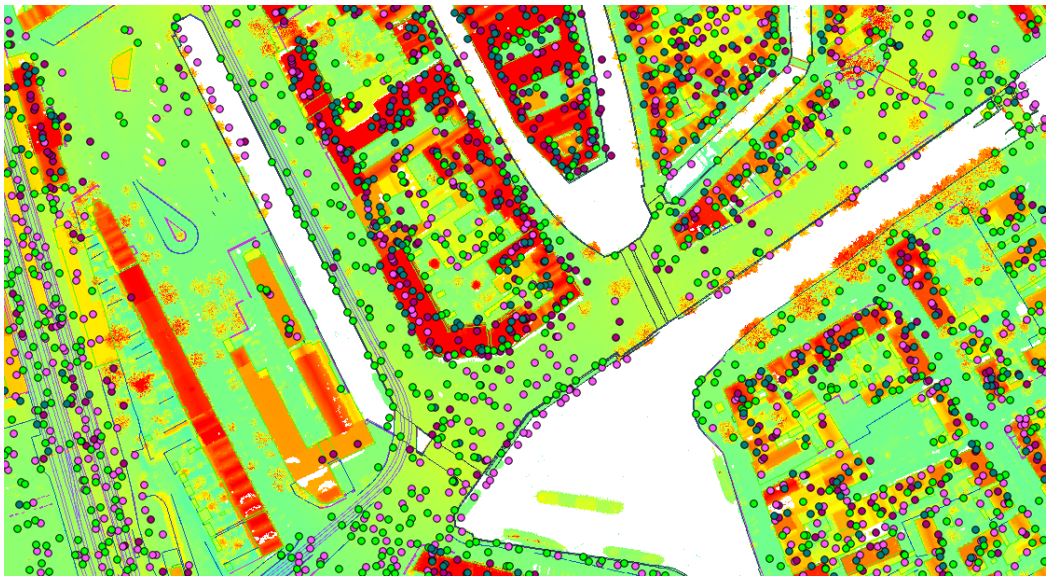


Figure 27: Scatterers selected using IRF correlation, in green: descending stack of 72 images, in pink: ascending stack of 68 images. Dark points are high and light point low. The accuracy is clearly improved compared to the 16 image stack in figure 26b. The accuracy of geocoding is a lot better, because the trend in the height estimation is greatly reduced. A lot of scatterers disappeared because the area on the left has been demolished for the train tunnel construction. In both ascending and descending data still a shift is visible, both relating to about a meter atmospheric delay in range.

images. Since the results using 16 images are already so good, the improvement is less obvious than the impact the use of a new method has. The scatterers in the 72 image results are more on straight lines, which reveals improved local accuracy. Also the absolute accuracy is higher, the points are accurately positioned in the whole processed area, while using 16 images a trend in estimated height appeared, influencing the geocoding. The difference in both relative and absolute positioning accuracy is mainly caused by height estimation, which is less accurate with less acquisitions, and depends on the correctness of the used deformation model.

Besides better point selection and improved geolocation another major improvement is the ability to easily inspect all scatterers in detail using the InSAR Inspector. This way it is easy to search for interesting points. For finding points that go down because of the tunnel construction a quadratic term is fitted to the deformation in addition to the linear one, and this quadratic term is selected for the point colour. It is immediately visible that the area around the spoorzone is accelerating, so one of the points, near the Binnenwatersloot, is selected to see its timeseries and other properties, as seen in figure 28. Until January 2011 it is stable, after that it is going down with about 10 mm/y. To identify the selected scatterer better a secondary screen can be opened, showing the scatterers on an aerial image, see figure 29. Also this secondary screen can be zoomed in further, and points can be selected from here. As can be seen in the table, the point is on the ground, and it appears to be on the street.

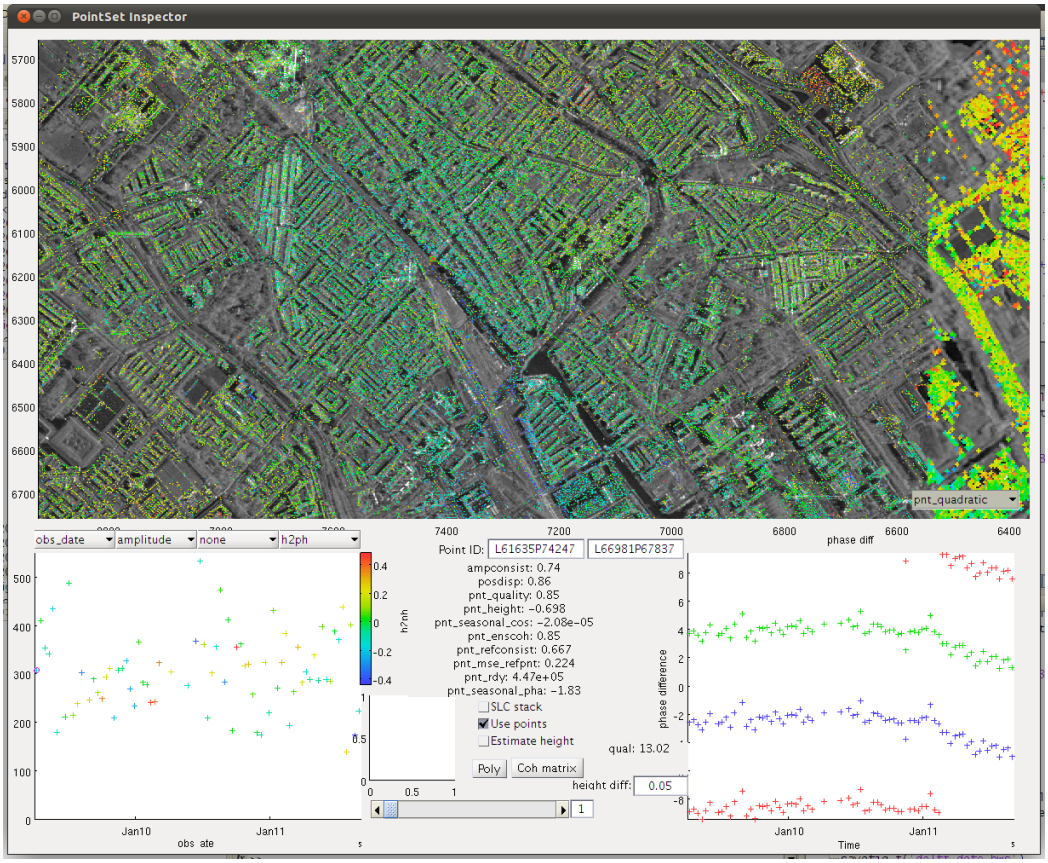


Figure 28: Spoorzone data loaded into the InSAR Inspector, with the quadratic deformation term selected for display.

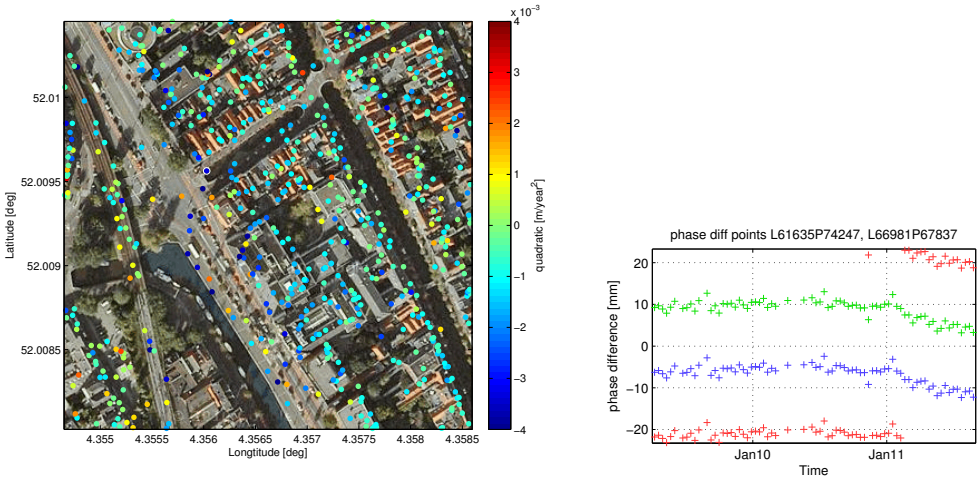


Figure 29: For further identification a secondary screen with an aerial photo is shown. The white circle denotes the selected point. It is on ground level on the Binnenwatersloot.

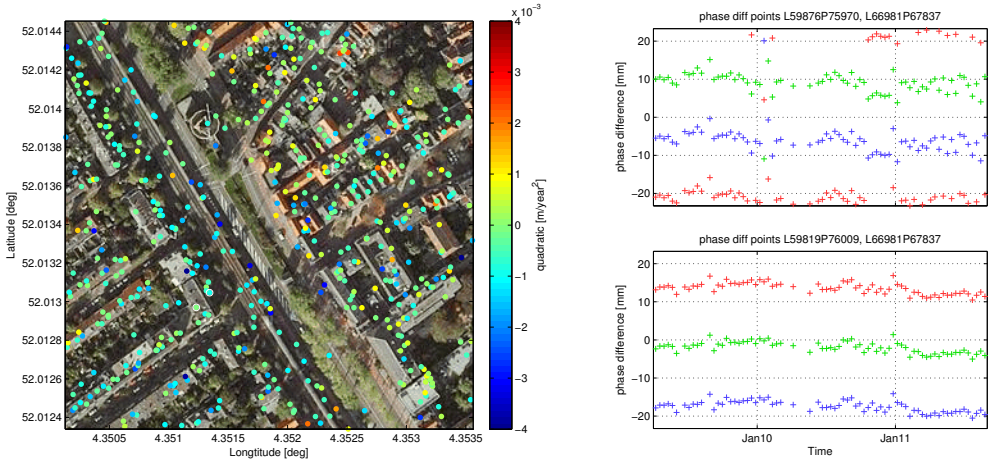


Figure 30: Some scatterers on a house in the Spoorsingel, one scatterer on top of the house, at 7.7 m, and one in front of the house at 1.2 m, shown with white rings (if you look closely). The high point shows a seasonal effect, most likely thermal expansion, while the low point did go down a few mm in January, but appears to be stable again.

Some other points have been selected on a house, for one of the people who reads this, see figure 30. Neither the top and the bottom front of the house are moving alarmingly, but there is clear movement visible in both.

6.3 STORAGE TANK MONITORING

In harbours very large tanks are used for storage of liquids. These tanks have to be monitored accurately to guarantee their safety. As an alternative for traditional measuring techniques a feasibility study for the use of InSAR for monitoring those tanks has been done by Hansje Brinker.

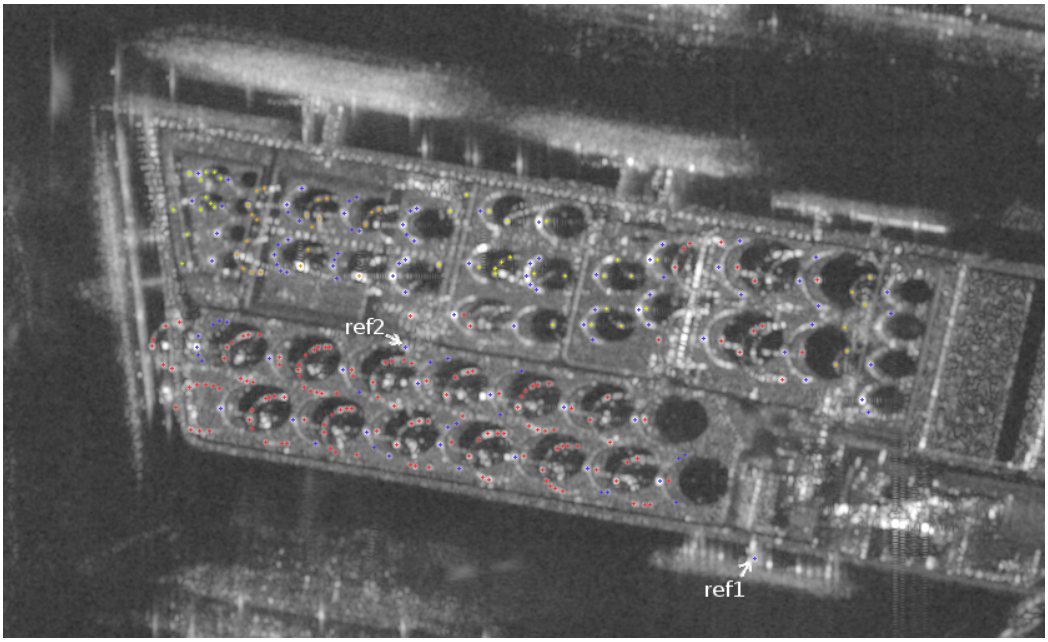
A processing using the standard processing chain didn't give satisfying results, almost all scatterers on the tanks were discarded because of unwrapping problems.

A detailed analysis follows, exploring the raw data, alternative unwrapping techniques and other new estimation methods. Also auxiliary information such as tank diameter and height is incorporated. For this analysis the toolbox showed to be extremely useful, being flexible enough to completely custom processing, while it still takes care of all the metadata, coordinate transformations and easy data access.

The rest of this section will show some results, while showing the ease of implementation.

Both EnviSat and TerraSAR-X are investigated, but the resolution of EnviSat turned out to be too low to distinguish separate tanks. Using TerraSAR-X the tanks are very distinguishable, each tank having at least a few bright scatterers, and the bigger tanks having the entire top edge clearly visible, as can be seen in figure 31. The downside of TerraSAR-X is the short wavelength, which makes that a phase ambiguity already happens after 16 mm deformation.

(a)



(b)

Figure 31: Overview of the inspected tanks. a) Reference optical image for interpretation purposes. b) Selected scatterers on the mean amplitude radar image. The image was acquired in descending orbit, hence the satellite is looking from the left. Reference point 1 was chosen for its recognisable location, but point 2 turned out to be more stable, so is used instead for further investigation.

The stability of the tanks is monitored using several techniques:

- Interferometric phase between tanks and a carefully selected reference point.
- Interferometric phase between bottom and top of the same tank.
- Subpixel precision position of a scatterer on the top edge of the tanks.
- Tilt estimation using the interferometric phase of the whole edge of the tanks.

Selection of the scatterers has been done using the InSAR Inspector GUI. This way scatterers could be selected while directly searching manually for possibilities where the most coherent scatterers are found. A very simple utility was written to label scatterers with their tank number and whether on the top or bottom edge of the tank. This could be easily done because the information about the current scatterer is exposed via the GUI object.

All selected scatterers are corrected using the known tank height, while assuming the bottoms of the tanks to be at the same altitude. The reference scatterer height was interactively estimated in the InSAR Inspector.

The phase difference between the tanks and the reference point is only interpretable for part of the tanks, for a lot of tanks it is completely unstable. This movement is apparently too variable for InSAR to be useful, at least when X-band radar is used. So other options for using InSAR were sought.

The phase between the top edge and the bottom edge turned out to be unwrappable for quite some tanks, but besides that still not all tanks are covered it is difficult to interpret this deformation. It might be thermal expansion, but also tilt or floor deformation can be plausible explanations.

As an attempt to make larger movements visible the subpixel precision estimated positions on the individual observations are investigated. The deformations measured that way are with respect to the coregistration, so while not optimal because the reference is not clear that way, it still gives an indication of the usability as the reference is sort of an average over the surroundings. The tanks generally show a movement of around 7 cm/y over the course of two years, with varying amounts of noise.

The final and most delicate method attempted is to estimate the tilt of the top edge through internal phase differences. For this the phase information from the top edge is extracted from the radar images by mapping the tank geometry to radar coordinates and requesting the phase from a stack of oversampled SLC images. The tilt is estimated from the phases using a 2D periodogram, after the phase from the first image is subtracted in time from the whole stack. This results in a tilt estimate in two directions. The estimate in azimuth direction is more accurate because the strongest scatterers are on the half tank facing the radar, causing a longer integration length in azimuth, and hence a higher accuracy.

The results of one of the tanks are shown in figure 32 as an example.

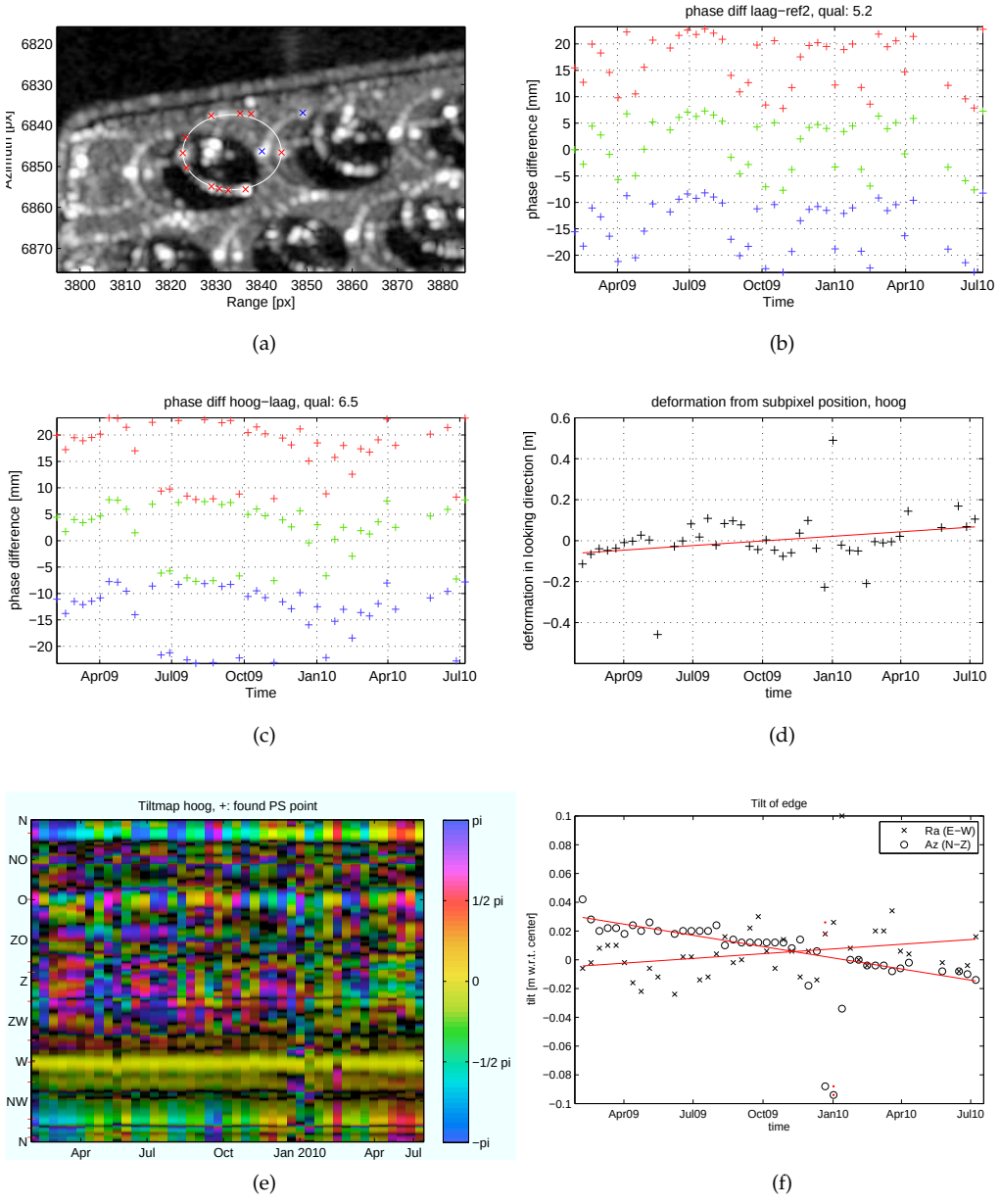


Figure 32: The results for one of the tanks, a tank of 18.5 m high and 39 m wide. a) Point scatterers selected on this tank, red=high, blue=low. b) Phase difference with reference point, a possible interpretation is that the deformation is accelerating with about 32 mm/y^2, which would result in such a pattern. c) The top-bottom difference shows a seasonal pattern, which can be explained by thermal expansion, the variation of 8mm on 18.5m relates to a temperature variation of 36°. d) The subpixel movement shows about 7 cm/y, but the accuracy is probably not high enough to fix ambiguities. e) The tilt of the top is estimated using the phase of the entire top edge, shown here for each acquisition, with the first acquisition as reference. f) The estimated tilt through time, both in azimuth and range direction. The red dots indicate discarded observations due to low quality of fit.

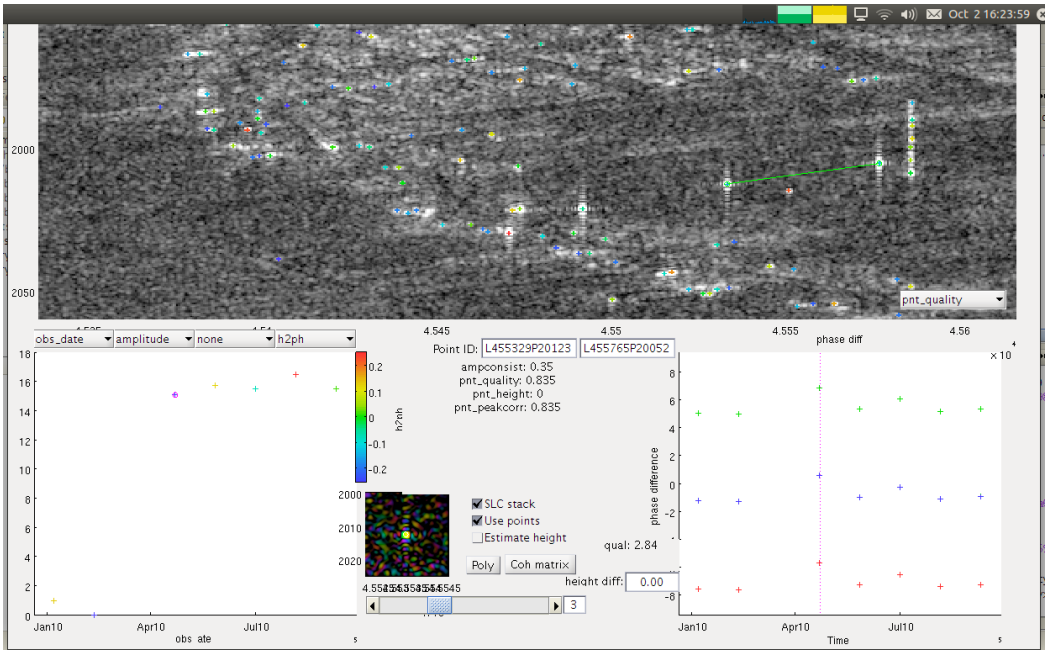


Figure 33: The corner reflector experiment in Cabauw. It lasted only 5 observations before the final EnviSat image was acquired.

6.4 CORNER REFLECTOR EXPERIMENTS

The TU Delft has done a series of experiments with corner reflectors and compact active transponders (CAT), which both act as perfect point scatterers. These experiments are done to measure and prove the accuracy of InSAR deformation measurements by comparing them with levelling.

For the corner reflector experiments generally not a full processing is necessary, since the main interest is in the phase differences between the corner reflectors, so no full network or atmosphere estimation is necessary.

The IRF correlation method is used to select the point scatters in an area around the corner reflectors. The resulting pointset is then directly loaded in the InSAR Inspector to locate the corner reflectors.

An experiment was done in Cabauw in 2010, which only lasted until the EnviSat mission was finished. The five observations that have been acquired can be seen in figure 33.

7

CONCLUSIONS AND RECOMMENDATIONS

The main goal of this research, improving the interpretation and identification of the measurements carried out using radar interferometry, has been reached in several aspects.

On one side, in the domain of automated processing, improvements in valid scatterer selection and geolocation accuracy are accomplished by using more knowledge about the radar signal, and taking the geoid height into account. These developments improve the results of automatic processing.

On another side, which is more visible and in the the domain of development and manual analysis, tools for easy access to data, results and metadata, and a GUI for visual inspection of processed results, or even unfinished processing, are developed. This improves the workflow, making result interpretation and understanding the main task, instead of all sort of management tasks to access the right data in the right format, apply the right corrections, and wonder if the right corrections were made. Since the tools are so flexible, trying something new is often just a matter of minutes work, improving the speed of development of new methods. These developments can influence the way people work, and help them to be more effective in reaching their goals.

Of course not all that can be done is done, and there are a lot of features waiting to be implemented, new methods to be tried, and implemented methods to be thoroughly tested. A incomplete list of the most relevant with respect to this thesis is:

- Improved methods for height estimation, decreasing the deformation model influence and increasing the density of the network.
- Research on the origin of trends in estimated height. To be thought of are influence of atmosphere estimation, network construction and influence of deformation models.
- Easier DEM handling, now DEMs still need to be preprocessed by DORIS, while this can easily be circumvented.
- Improved filtering, combining available quality measures in an statistically optimal way, in contrast to separate thresholds on each of those measures.
- Amplitude variation characterisation with respect to baselines. Although this was one of the original research ideas it appeared to be only useful for a small percentage of all scatterers, but this statement has to be more thoroughly researched.

BIBLIOGRAPHY

Nico Adam. Ps processing: Noise budget. 10 2003. (Cited on page 11.)

Ken Arroyo, Prajnaparamita Bhattacharya, Filip Biljecki, Dinesh Kalpoe, Álvaro Muñoz, Piers Titus van der Torren, Stijn Verlaar, and Hang Yu. Satellite radar observation feasibility study for large infrastructural public works, a case study on the delft train tunnel. *Geomatics Synthesis Project – Argus Panoptes*, 2009. (Cited on pages 41, 45, and 46.)

Richard Bamler and Philipp Hartl. Synthetic aperture radar interferometry. *Inverse Problems*, 14:R1–R54, 1998. URL http://doris.tudelft.nl/Literature/bamler98.pdf. (Cited on page 39.)

CEOS. Sar callibration workshop, 1993. (Cited on page 12.)

Ian Cumming and Frank Wong. *Digital Processing Of Synthetic Aperture Radar Data: Algorithms And Implementation*. Artech House Publishers, New York, 2005. ISBN 1580530583. (Cited on pages 8 and 30.)

DORIS. *Delft Object-oriented Radar Interferometric Software (DORIS), User's manual and technical documentation*. Delft Institute of Earth Observation and Space Systems (DEOS), Delft University of Technology, Delft, The Netherlands, 1998. (Cited on page 15.)

Alessandro Ferretti, Claudio Prati, and Fabio Rocca. Permanent scatterers in SAR interferometry. *IEEE Transactions on Geoscience and Remote Sensing*, 39(1):8–20, January 2001. (Cited on page 13.)

Dirk Geudtner. *Die Interferometrische Verarbeitung von SAR-Daten des ERS-1*. PhD thesis, Universität Stuttgart, 1995. Draft. (Cited on page 8.)

Ramon Hanssen and Freek van Leijen. On the potential of persistent scatterer interferometry for monitoring dikes in the netherlands. 9e Nederlands Aardwetenschappelijk Congres 18-19 March, 2008, Veldhoven, 2008. URL http://doris.tudelft.nl/Literature/hanssen08c.txt. (Cited on page 1.)

Ramon F Hanssen. *Radar Interferometry: Data Interpretation and Error Analysis*. Kluwer Academic Publishers, Dordrecht, 2001. (Cited on pages 1, 5, 11, 12, 13, 29, and 39.)

Andy Hooper, Howard Zebker, Paul Segall, and Bert Kampes. A new method for measuring deformation on volcanoes and other non-urban areas using InSAR persistent scatterers. *Geophysical Research Letters*, 31:L23611, doi:10.1029/2004GL021737, December 2004. (Cited on page 1.)

Dieter Just and Richard Bamler. Phase statistics of interferograms with applications to synthetic aperture radar. *Applied Optics*, 33(20): 4361–4368, 1994. (Cited on page 11.)

Gini Ketelaar and Ramon Hanssen. Monitoring subsidence due to gas extraction in Groningen using satellite radar interferometry. 8ste Nederlands Aardwetenschappelijk Congres 24-25 April, 2006, Veldhoven, 2006. URL http://doris.tudelft.nl/Literature/ketelaar06a.pdf. (Cited on page 1.)

V.B.H. (Gini) Ketelaar. *Satellite Radar Interferometry: Subsidence Monitoring Techniques*. Remote Sensing and Digital Image Processing. Springer, 2009. (Cited on page 5.)

Daniele Perissin. *SAR super-resolution and characterization of urban targets*. PhD thesis, Politecnico di Milano, Italy, 2006. (Cited on page 38.)

L M Th Swart. Spectral filtering and oversampling for radar interferometry. Master's thesis, Faculty of Geodetic Engineering, Delft University of Technology, December 2000. URL http://doris.tudelft.nl/Literature/swart00.pdf. (Cited on page 8.)

Part I

APPENDIX

A

TOOLBOX MANUAL

On the following pages a full reference manual of the created toolbox classes is given, including the visual InSAR Inspector tool, and the SlcStack, Slc, SlcImage, PointSet, Filter, Poly3D, Raster, RasterData, hdf5prop, and dependentprop classes.

Chapter 1

Class: `pointset inspector`

Visual inspector tool for analysis of PointSet objects. The InSAR Inspector is a GUI that provides a way to make the spatially dense, multivariate, and relative information that is involved in InSAR processing fully explorable.

An overview map is shown at the main screen, on which all scatterers are shown. When a scatterer is selected, either on this overview map or using a unique point ID, all information about a scatterer is shown, both as numerical data and using plots. All information that is added during processing is shown, such as quality measures, height, and deformation. Information that varies per acquisition is plotted, such as amplitude or deformation. For deeper inspection several options are available, such as showing the underlying radar image data, calculating a coherence matrix, and data access from the command line.

The GUI object gives access to the selected points and all their properties, and also to all GUI elements and display functions. Interaction via the command line is necessary when more flexibility or more user input is required than can be given through the GUI. For example to show a custom property on the background, make high quality plots of selected points, or estimate new parameters or relations.

`psinspector = pointset_inspector( pointset )` Start an instance of the PointSet Inspector on the given PointSet. The returned object can be used to interact with the GUI via the command line.

1.1 Properties

`handles` Struct containing all graphics handles. Using this custom data can be added to the plots.

`pointset` The PointSet with which the GUI was initiated. convenience copies

`slc` SlcStack object from the pointset

`poly` Polygon selected using the polygon selection tool image buffers

`mrm` Stack of RasterData objects shown on the background.

`selected1` PointSet containing the selected point.

selected2 PointSet containing the reference point.

timesel Selected acquisition index.

pnt_custom Custom point attribute for visualisation, can for example be set to interferemetric phase using pnt_custom = wrap(diff(pointset.phase_corrected(:,[master slave]),[],2));

obs_custom Custom observation attribute for visualisation.

filt Optional selection of the points that are displayed on the background. settings

use_points Setting whether to use points from the pointset or raw data.

show_slc Setting whether to load and show the slc stack on selecting a point.

estimate_height Setting whether to estimate the height difference on the fly or use the given point height.

subtr_atmo Setting whether to correct the phases in the phase plot for atmosphere.

norefpoint Setting whether to subtract the reference point or not.

1.2 loadbackgrounddetail

Load high resolution background around selected point.

gui.loadbackgrounddetail(ovs) Use given oversampling rate for the background. If omitted it defaults to 1. If ovs == 0 the background will be removed.

1.3 loadaerial

Load an aerial view with selectable points.

gui.loadaerial(zoom) Load an aerial view with Google zoomlevel zoom. Default is 17.

1.4 set_orientation

Set the orientation of the map.

tsi.set_orientation() Set the orientation to rounded north.

tsi.set_orientation('north') Set the orientation to exact north.

tsi.set_orientation(dir) Set the orientation to dir degree.

Chapter 2

Class: `Slc`

Radar image class. Stores all metadata of a radar image.

`slc = Slc.fromResFiles(basedir, slaveres, ifgres, masterres)` Reads radar image metadata from DORIS .res files.

2.1 Properties

`id` short identification string: acquisition date as number.

`sensor` Sensor description string.

`date` The date of observation (including time) [days].

`timeOfDay` The time of day of the first pixel, for ms accuracy, also the time used for orbits [s].

`timeToFirstPixel` two way travel time to first pixel [s].

`wavelength` Radar wavelength [m].

`PRF` Pulse Repetition Frequency [Hz].

`RSR` Range Sampling Rate [Hz].

`bandwidthAz` Bandwidth in azimuth direction [Hz].

`bandwidthRa` Bandwidth in range direction [Hz].

`poly_fDC` Doppler centroid frequeny [Hz]. Poly2D object with input in seconds from first pixel, [line / PRF, pixel / RSR]. For TerraSAR it is given 2D (but not in Doris .res file), for Envisat only 1D.

`preciseOrbit` Satellite orbit, n-by-4 array with columns [time_of_day, x, y, z]

`is_ascending` True if the orbit is ascending, false if descending.

`orig` Info about original file. .filename .fileformat .window [Raster]

`crop` Info about crop file. .filename .fileformat .window % Raster

rsmp Info about resampled file (to master coordinates). .filename .fileformat .window [Raster]

bperp Perpendicular baseline (from DORIS)

bpar Paralel baseline (from DORIS)

btemp Temporal baseline (from DORIS)

poly_coreg Coregistration polynomial, 1x2 Poly2D object for line and pixel.

n_coregwin Number of coregistration windows.

poly_h2ph h2ph polynomial as calculated by DORIS. Poly2D object. Not used anymore, superceded by poly3_h2ph.

poly_flat flat earth polynomial as calculated by DORIS. Poly2D object. Not used anymore superceded by poly3_refpha.

poly3_refpha Topographic phase polynomial. Poly3D object. Returns the topographic phase on a [line, pixel, height] coordinate. Height is relative to the geoid.

poly3_h2ph h2ph polynomial. Poly3D object. Returns the derivative of poly3_refpha to height on a [line, pixel, height] coordinate.

poly_geocoding Geocoding polynomial for geocoding to WGS84 XYZ coordinates. 1x3 Poly3D object. Returns the XYZ coordinates for a [line, pixel, height] coordinate. Height is relative to the geoid.

poly_latlon Geocoding polynomial for geocoding to WGS84 [lat,lon] coordinates. 1x2 Poly3D object. Returns the [lat,lon] coordinates for a [line, pixel, height] coordinate. Height is relative to the geoid.

poly_xyz2linepixel Inverse geocoding polynomial from WGS84 XYZ coordinates. 1x2 Poly3D object. Returns the [line,pixel] coordinates for a [X,Y,Z] coordinate.

poly_latlonh2linepixel Inverse geocoding polynomial from WGS84 [lat,lon] coordinates. 1x2 Poly3D object. Returns the [line,pixel] coordinates for a [lat,lon,height] coordinate.

geocoding_shift Shift in [line,pixel]

calibrationFactor General calibration factor to apply when reading data.

res_slave Content of DORIS slave res file. Only for debugging.

res_ifg Content of DORIS interferogram res file. Only for debugging.

res_master Content of DORIS master res file. Only for debugging.

spectrumAz Spectrum in azimuth, Filter object. Contains relative bandwidth and spectrum shape. Not shifted to doppler centroid because that depends on location.

spectrumRa Spectrum in range, Filter object. Contains relative bandwidth and spectrum shape.

velocity Indication of satellite velocity

heading Satellite flight direction, calculated at center of original frame

preciseOrbit_etrs Satellite orbit converted to from ITRS to ETRS

look_angle Indication of satellite look angle (angle between earth center and look direction)

incidence_angle Indication of satellite signal incidence angle (angle at which the radar signal reaches the ground)

spacing_az Pixel spacing on ground in azimuth/flight direction, or line spacing [m]. Calculated at center of original frame.

spacing_ra_slant Range pixel spacing, in looking direction [m].

spacing_ra_ground Range pixel spacing projected to geoid [m]. Calculated at center of original frame.

fDC_mean

fDC

poly_coregRa

poly_coregAz

2.2 clone

Create a clone of self.

2.3 make_poly_refpha_h2ph

Creates polynomials for topographic phase computation. The following polynomials are created by this function: poly3_refpha and poly3_h2ph. poly3_h2ph is the derivative of refpha to height, and can be used for iterative estimation of elevation.

accuracy = slc.make_poly_refpha_h2ph(degree, height_range, refframe) Create topographic phase polynomials and store them in slc. All arguments are optional. `degree` overides the polynomial degree of 7. `height_range` the valid height range of [-500 9000], which includes all places on the earth surface from the Dead Sea to the Mount Everest. `refframe` can be `ETRS` to convert the orbits to ETRS before usage and as such correct for tectonic plate movement in Europe. The returned structure includes quality descriptions of the polynomials, with the fields std for standard deviation and max for maximum error.

2.4 readDataRsmp

Read resampled SLC data from disk, and return as an SlcImage.

imc = slc.readDataRsmp(crop, filterwindow) Read the specified crop from the SLC. `crop` must be given as a `Raster` object. If omitted the whole file is read. If the crop is oversampled the SLC is automatically oversampled to meet the crop. If `filterwindow` is specified the SLC data is spectrally filtered to the given window, which can be used to reduce sidelobe levels or increase resolution.

Example:

imc = slc.readDataRsmp(Raster(5101:6100,2401:2500).oversample(2), kaiser(100, 3)) Load a crop of 1000x100 pixels, oversampled 2 times, and filtered to a Kaiser window with beta = 3.

2.5 make_poly_geocoding.bak

MAKE_POLY_GEOCODING Creates polynomials for fast geocoding w.r.t. the geoid. The following polynomials are created by this function: poly_geocoding, poly_latlon, poly_xyz2linepixel and poly_latlonh2linepixel.

accuracy = slc.make_poly_geocoding('option', value, ...) Create geocoding polynomials and store them in slc. Optional arguments are:

shift_linepixel Define a shift in the radar coordinates, as [line, pixel].

degree Polynomial degree used, default 5.

height_range Valid height range, default [-500 9000], which includes all places on the earth surface from the Dead Sea to the Mount Everest.

geoid Whether to use the geoid as reference heigth, otherwise the ellipsoid is used. default true.

refframe Can be ETRS to convert the orbits to ETRS before usage.

The returned structure includes quality descriptions of the polynomials, with the fields std for standard deviation and max for maximum error.

2.6 lph2xyz

Calculate the XYZ coordinate from line, pixel and height above ellipsiod.

[x,y,z] = slc.lph2xyz(line, pixel, height, refframe) refframe can be ETRS to convert the orbits to ETRS before usage.

In general the geocoding polynomials can be used instead of this function, either direct or via the geocoding functions in SlcStack

2.7 make_poly_geocoding

Creates polynomials for fast geocoding w.r.t. the geoid. The following polynomials are created by this function: poly_geocoding, poly_latlon, poly_xyz2linepixel and poly_latlonh2linepixel.

accuracy = slc.make_poly_geocoding('option', value, ...) Create geocoding polynomials and store them in slc. Optional arguments are:

shift_linepixel or shift_xy Define a shift in the radar coordinates, either as [line, pixel] or as [x, y] in metre.

degree Polynomial degree used, default 5.

height_range Valid height range, default [-500 9000], which includes all places on the earth surface from the Dead Sea to the Mount Everest.

geoid Whether to use the geoid as reference heigth, otherwise the ellipsoid is used. default true.

refframe Can be ETRS to convert the orbits to ETRS before usage.

The returned structure includes quality descriptions of the polynomials, with the fields std for standard deviation and max for maximum error.

2.8 xyz2azra

Calculate the [line, pixel] coordinates from XYZ coordinates.

[line,pixel,dist] = slc.xyz2azra(xyz, refframe) refframe can be ETRS to convert the orbits to ETRS before usage.

In general the geocoding polynomials can be used instead of this function, either direct or via the geocoding functions in SlcStack

2.9 makeInterferograms

make interferograms of an slc stack and save them as png. Input parameters: dirname Directory where to save images master Master index for single master ifgs, or incremental, positive, all, or magnitude crop Raster object of crop doFilter do range and azimuth filtering window apply window, kaiser(N,4) is default flip lr, ud ovs Oversample factor range magnitude range (for mixed), logarithmic. Adaptive is default ref reference coordinate phaseonly don't mix phase and magnitude cohAsMag estimate coherence and use that as magnitude

TODO: Too many input parameters...

Chapter 3

Class: SlcStack

The SlcStack class represents a stack of Slc images. Properties of the class apply to the stack as a whole, such as wavelength and pixel spacing. Properties that are aqcuisition-specific can be found in the individual Slc objects in the slc array.

3.1 Properties

n_images The number of Slc images in the stack.

slc An array of Slc objects with length equal to n_images.

date Array of dates (datenum) of all aqcuisitions.

dir The base directory of the raw Slc data.

master_idx Index in the Slc array of the master image, to which all the other images are coregistred.

master The master Slc object

sensor Sensor(s) description string.

wavelength Radar wavelength [m].

PRF Pulse Repetition Frequency [Hz].

RSR Range Sampling Rate [Hz].

is_ascending True if the orbits are ascending, false if orbits are descending.

poly_geocoding Geocoding polynomial for geocoding to WGS84 XYZ coordinates. 1x3 Poly3D object. Returns the XYZ coordinates for a [line, pixel, height] coordinate. Height is relative to the geoid. See Slc/make_poly_geocoding for technical details.

poly_latlon Geocoding polynomial for geocoding to WGS84 [lat,lon] coordinates. 1x2 Poly3D object. Returns the [lat,lon] coordinates for a [line, pixel, height] coordinate. Height is relative to the geoid.

poly_xyz2linepixel Inverse geocoding polynomial from WGS84 XYZ coordinates. 1x2 Poly3D object. Returns the [line,pixel] coordinates for a [X,Y,Z] coordinate.

poly_latlonh2linepixel Inverse geocoding polynomial from WGS84 [lat,lon] coordinates. 1x2 Poly3D object. Returns the [line,pixel] coordinates for a [lat,lon,height] coordinate.

poly_h2ph h2ph polynomial as calculated by DORIS. Poly2D object. Not used anymore, superceded by poly3_h2ph.

poly_flat flat earth polynomial as calculated by DORIS. Poly2D object. Not used anymore superceded by poly3_refpha.

poly3_refpha Topographic phase polynomial. Poly3D object. Returns the topographic phase on a [line, pixel, height] coordinate. Height is relative to the geoid. See Slc/make_poly_refpha_h2ph for technical details.

poly3_h2ph h2ph polynomial. Poly3D object. Returns the derivative of poly3_refpha to height on a [line, pixel, height] coordinate. Difference to the DORIS computed is that multiplying with m2ph is not necessary.

poly_accuracy polynomial accuracy and range information

spacing_az Pixel spacing on ground in azimuth/flight direction, or line spacing [m]. Calculated at center of original frame.

spacing_ra_slant Range pixel spacing, in looking direction [m].

spacing_ra_ground Range pixel spacing projected to geoid [m]. Calculated at center of original frame.

velocity Satellite velocity

heading Satellite flight direction, calculated at center of original frame

look_angle Indication of satellite look angle (angle between earth center and look direction)

incidence_angle Indication of satellite signal incidence angle (angle at which the radar signal reaches the ground)

3.2 SlcStack

Create an SlcStack object. The first argument is mandatory, the rest is optional and follows the key-value convention. Any of the optional arguments can be combined.

slcstack = SlcStack(dir) Creates an SlcStack object from data in directory `dir`, using autodetection to determine the data format. Supported formats are DORISstack, DePSI and Stamps, relying on the specific constructors `fromDorisStackDir`, `fromDepsiDir` and `fromStampsDir`.

slcstack = SlcStack(dir,'reject',reject) Rejects images based on a boolean or integer `reject` array.

slcstack = SlcStack(dir,'fromdate',from,'todate',to) Rejects images taken before `from` or after `to`, both specified as datenum.

`slcstack = SlcStack(dir,'refframe','ETRS')` Use ETRS reference frame for orbits in computation of refpha and h2ph.

3.3 linepixelheight2xyz

Convert the given coordinates to WGS84 XYZ.

Height is interpreted as height w.r.t. the geoid.

```
xyz = slcstack.linepixelheight2xyz([line, pixel, height])
```

```
[x, y, z] = slcstack.linepixelheight2xyz([line, pixel, height])
```

3.4 linepixelheight2latlon

Convert the given coordinates to WGS84 [latitude, longtitude].

Height is interpreted as height w.r.t. the geoid.

```
latlon = slcstack.linepixelheight2latlonh([line, pixel, height])
```

```
[lat, lon] = slcstack.linepixelheight2latlonh([line, pixel, height])
```

3.5 xyz2linepixel

Convert the given coordinates to line, pixel. Coordinates are interpreted as WGS84 XYZ coordinates.

```
linepixel = slcstack.xyz2linepixel([x, y, z])
```

```
[line, pixel] = slcstack.xyz2linepixel([x, y, z])
```

3.6 latlonh2linepixel

Convert the given coordinates to line, pixel. Coordinates are interpreted as WGS84 coordinates with height w.r.t. the geoid.

```
linepixel = slcstack.latlonh2linepixel([lat, lon, height])
```

`[line, pixel] = slcstack.latlonh2linepixel([lat, lon, height])` If the height column is omitted it is set to 0.

Chapter 4

Class: `SlcImage`

Class for storage and processing of SLC data.

An `SlcImage` object contains a crop of data of an SLC image, and has functions for processing this data.

The normal way to obtain an `SlcImage` is using the readData(Rsmp) function in `Slc`.

4.1 Properties

`raster` Coordinates and spacing of the data. Raster object.

`data` Data array.

`slc` `Slc` object of the SLC.

`coordinateSystem` Coordinate system in which the data is sampled. Can be `original` or `resampled`. `resampled` means resampled to master coordinates.

`subtrflat` True if flatearth is removed.

`filtAz` Azimuth spectrum of data.

`filtRa` Range spectrum of data.

`filtFreq`

`crop`

4.2 `oversample`

Oversample Slc data. The filter used for oversampling is the MATLAB default with a filter order of 10. The shift of the spectrum in azimuth direction, the doppler shift, is taken into account by shifting the filter.

`imc = imc.oversample(ovs)` Oversample data `ovs` times.

4.3 cropTo

Crop the data to the given raster.

imc = imc.cropTo(raster) Crop the data to raster, where raster is a Raster object.

4.4 filterRa

Filter the data in range direction. The filter to be applied is given as a Filter object, which can be used to construct spectral filters.

imc = imc.filterRa(filter, n) Filter the data in range direction, using the given Filter, using a n point filter. If n is omitted n=64.

4.5 filterAz

Filter the data in azimuth direction. The filter to be applied is given as a Filter object, which can be used to construct spectral filters.

imc = imc.filterAz(filter, n) Filter the data in azimuth direction, using the given Filter, using a n point filter. If n is omitted n=64.

4.6 rewindow

Filter the data to the given window. A very useful window is the Kaiser window, constructed in MATLAB with `window = kaiser(N, beta)`, where N is sufficiently large and beta defines the tradeoff between resolution and side lobe levels.

imc = imc.rewindow(window, n) Filter the data so that the spectrum shape is equal to window, using a n point filter. If n is omitted n=64. Example:

```
imc = imc.rewindow(kaiser(1025, 3))
```

4.7 subtractReferencePhase

Subtract the reference phase from the data. The reference phase is calculated using the polynomial in slc.poly3_refpha, evaluated at height = 0. See Slc.make_poly_refpha_h2ph() for the polynomial calculation.

imc = imc.subtractReferencePhase() Subtract the reference phase from the data.

4.8 show

Show image or stack of image using Image class.

4.9 plotSpectrumRange

Plot the real and theoretical spectrum of the data in range direction.

4.10 plotSpectrumAzimuth

Plot the real and theoretical spectrum of the data in azimuth direction.

4.11 mean

Return mean of a stack.

4.12 std

Return std of a stack.

4.13 mean_abs

Return mean of the absolute values of a stack.

4.14 std_abs

Return std of the absolute values of a stack.

4.15 value

Return the values at the given coordinates.

`val = imc.value(lines, pixels)` Return the values at the given coordinates. The values are interpolated using a cubic kernel if they are not exact on data points.

4.16 clone

Create a clone of self.

4.17 filterRaOverlap

Filter all SLC images to have the same spectrum in range (pixel) direction, so to the part of the spectrum that is overlapping. In this direction the spectrum can be shifted due to incidence angle differences. For this filtering the inclination and pointing angle of the ground are needed. For this poly_flat is now used, which follows the ellipsoid.

By supplying an additional slope value the incidence angle can be multiplied to get another angle. This is still experimental.

4.18 nearest_peaks

Find the coordinates of the local maxima nearest to the given coordinates.

`[lines, pixels] = nearest_peaks( self, linepixel )` Find the [line, pixel] coordinates of the local maxima nearest to the given coordinates. Those coordinates must be supplied as a n-by-2 array [lines, pixels]. The coordinates are estimated with subpixel precision.

4.19 irf_corr

Calculate the impulse response correlation

4.20 coherence

Calculates the coherence between two SLC images.

`coh = imc1.coherence(imc2 [, win, type])` Calculate the coherence between `imc1` and `imc2`, using the window `win`. If `win` is omitted a window is constructed based on the impulse response of the signal, giving a high calculation resolution, and independence from deformation or other phase signals. `type` can be `projected`, `complex`, or `absolute`, where `projected` is the default.

4.21 peakcoords

Find the coordinates of all local maxima in an image with subpixel precision.

`[lines,pixels] = im.peakcoords( mask )` Return the line and pixel coordinates of all local maxima, based on absolute values. The coordinates are estimated with subpixel precision. An optional mask can be supplied to mask out unneeded areas.

4.22 interferogram

ifg = interferogram(self, master, doFilter, window, doSubtrflat, cohAsMag) TODO: must be cleaned up/generalised

4.23 filterAzOverlap

Filter all SLC images to have the same spectrum in azimuth (line) direction, so to the part of the spectrum that is overlapping. In this direction the spectrum can be shifted due to squint angle or doppler centroid differences.

4.24 coherencematrix

Calculate the coherence between each two SlcImages in an array. The upper triangle gives the average coherence over the area, the lower triangle gives the maximum coherence in the area.

`cohmat = coherencematrix([imc1, imc2, ...], crop, win, type)` Calculate the coherence between each two SlcImages in the array. `crop` defines the area that should be included in the calculation, and must be a boolean matrix of the same size as imc.data. `win` and `type` are described in `coherence` and are usually omitted.

Chapter 5

Class: PointSet

The PointSet class represents a database of point data such as phase, coordinates, etc.

Point data is organized in matlab arrays, or similar reacting object as hdf5prop or dependentprop, and follows the convention that the first dimension equals the number of points on the pointset (n_points) and the second the number of observations (n_observations), i.e. the number of images in the stack. Data that is the same for all points (such as the aqcuisition date) has its first dimension equal to one (row vector) and a name that starts with obs_, such as in obs_date. Data that is independent of aqcuisition has its second dimension equal to one (column vector) and a name that starts with pnt_, such as in pnt_quality. Data that changes with both point and aqcuisition has no prefix, eg. amplitude and phase. Properties can be added dynamically, and should follow this convention. The only properties that are exempted are n_points, n_observations, meta, log, and slc.

The amplitude is not an element of the PointSet by default as it is not required in any of the main functions and dependent properties, but it is usually present as a dynamic property. The pnt_quality can mean different things at different processing stanges, but should be the most recent quality measure of the points in the set. For example, initially it can be the amplitude consistency, and later after network estimation it will be the ensemble coherence. Separate, specific properties for these values are likely added in future versions.'

5.1 Properties

n_points Number of points.

n_observations Number of observations.

meta Structure of meta data that is not attributed to points or observations, such as the mrm or a polygon that was used to select the current set of points.

log A Logger object that maintains a history of standard output messages.

slc The SlcStack object that the point data is obtained from.

phase The real-valued phase of each point per aqcuisition.

pnt_line The line coordinate of each point.

pnt_pixel The pixel coordinate of each point.

pnt_quality The quality of each point.

pnt_height The (current estimate of the) height of each point.

pnt_lat The estimated lattitude of each point based on the estimated height.

pnt_lon The estimated longitude of each point based on the estimated height.

obs_date The date of each aqcuisition.

h2ph The height-to-phase of each point per observation.

topo_phase The phase corresponding to the current `pnt_height`.

atmo_phase The phase corresponding to the estimated atmosphere in `meta`.

phase_notopo The measured phase corrected for the topographic phase `topo_phase`.

phase_notopo_noatmo The measured phase corrected for the topographic and atmospheric phase `topo_phase` and `atmo_phase`.

phase_corrected The measured phase corrected for the topographic phase and, if available, the atmospheric phase. This is the best available phase for model estimations.

defo line of sight deformation w.r.t. first observation and w.r.t. meta.refpoint [m]

5.2 PointSet

Create a PointSet object.

pointset = PointSet(slcstack) Creates an empty pointset for an SlcStack object, setting `n_observations` equal to the images in the stack and `n_points` equal to zero.

pointset = PointSet(slcstack,props) Adds the properties named in the `props` cell array, following the naming conventions as outlied in this object's documentation, and initialized with nan values.

5.3 append

Add point data to the pointset, increasing the `n_points` number of points. This modifies the pointset in place.

pointset.append(other) Appends the contents of the `other` pointset to `pointset`. The new value of `n_points` is incremented correspondingly. Data fields of the two pointsets are merged, and nan values are put where data is missing due to the merge. For example, if `phase` exists in `other` but not in `pointset`, then `phase` is added to `pointset`, initialized with nan values, and phase data from `other` is copied into the second part.

5.4 add_points

Allocates space for extra points in the pointset. This modifies the pointset in place.

index = pointset.add_points(n) Increases n_points by n and and grows data fields accordingly with nan values. The index vector points into the modified pointset at the newly allocated space to aid subsequent filling of data.

5.5 add_observations

Allocates space for extra observations in the PointSet. Care must be taken to update the slc property consistently.

5.6 add_property

Dynamically adds data fields to the pointset. This modifies the pointset in place. If a property with given name already exists it is overwritten.

pointset.add_property(name) Adds property with given name, and initializes it with nan values according to the naming conventions as outlined in the PointSet help text.

pointset.add_property(name, prop) Adds property with given name, and value prop.

pointset.add_property(name, type) Adds property with given name, and initializes it with nan values of type according to the naming conventions as outlined in the PointSet help text.

5.7 has_property

Test if pointset contains a certain property.

retval = pointset.has_property(name) Returns true if pointset.(name) exists, or false otherwise.

5.8 del_property

Deletes a certain property from the property. This modifies the pointset in place.

pointset.del_property(name) Deletes pointset.(name) if it exists.

5.9 clone

Makes a deep copy of the pointset.

cloned = pointset.clone() Creates a new pointset cloned that is a replica of the original pointset. Of the special properties, log is not copied. meta is copied and may share data with the original pointset if meta contains objects.

5.10 xyz_wgs84

Calculates the X,Y,Z coordinates of the points in the coordinate frame of WGS84.

xyz = pointset.xyz_wgs84(sel)

[x, y, z] = pointset.xyz_wgs84(sel) Return the WGC84 XYZ coordinates of the subset sel.

5.11 rdnap

Calculate the coordinates in the Dutch RDNAP coordinate system, computed with sub cm accuracy.

rdnap = pointset.rdnap(sel)

[rdx, rdy, nap] = pointset.rdnap(sel) Return the rdnap coordinates of the subset sel.

5.12 subset_index

Create a subset of a pointset based on index.

sub = pointset.subset_index(I) Direct selection of points based on a boolean or index vector.

5.13 subset_coord

Create a subset of a pointset based on radar coordinates.

sub = pointset.subset_coord(C) Select all points that are (within epsilon radius) on the given (line, pixel) coordinates. If specified as a matrix, C should consist of two columns. If specified as cell array then each item C{i} should be a two-element vector. If specified as a string then the format is L...P...\nL...P...\n..., where the dots contain the line and pixel coordinate times 10, for exampe, 10.23,14.25 becomes L102P143.

5.14 shift_geocoding

Change the geocoding to correct for timing errors, orbit errors, atmosphere delays, etc.

out = pointset.shift_geocoding('shift_linepixel',[lineshift, pixelshift]) Change the geocoding to shift

out = pointset.shift_geocoding('shift_xy',[xshift, yshift]) Change the geocoding to shift points. `xshift` is the shift to be applied in west-east direction, `yshift` in south-north. Both given in meter.

5.15 plot_mrm

TODO: document

5.16 plot_highres

make high resolution image of a crop

5.17 subset

Create a subset of a pointset based on a variety of criteria. Arguments are passed by the key-value convention. Any of the following settings can be combined.

sub = pointset.subset('index',I) Direct selection of points based on a boolean or index vector.

sub = pointset.subset('coord',C) Select all points that are (within epsilon radius) on the given (line, pixel) coordinates. If specified as a matrix, `C` should consist of two columns. If specified as cell array then each item `C{i}` should be a two-element vector. If specified as a string then the format is `L...P...\nL...P...\n...`, where the dots contain the line and pixel coordinate times 10, for exampe, 10.23,14.25 becomes L102P143.

sub = pointset.subset(property, limits) Select points based on `pnt_property`. `limits` should be a minimum value or a [minimum, maximum] pair. `property` should be a property for which a `pnt_property` exists, like `quality`, `height`, `line`, etc.

sub = pointset.subset('crop', raster) Select only points within the extent of `raster`

sub = pointset.subset('poly', poly) Select only points within the polygon or multipolygon `poly`. Also see `INMULTIPOLYGON`.

sub = pointset.subset('nearlatlon', [lat lon], distance) Select only points within distance of a given coordinate Distance can be given in meter (eg `100m`) or in mainlobes (eg. 3)

sub = pointset.subset('nearlinepixel', [line pixel], distance) Select only points within distance of a given coordinate Distance can be given in meter (eg `100m`) or in mainlobes (eg. 3)

[sub,sub2,...] = pointset.subset('mindist',d) Select points that are at a certain minimum distance to each other. Points are added in descending order of quality. After each addition the radius around the added point is discarded, and the next best point is added, until the set is exhausted. If specified as a number the implied unit is the mainlobe size: 1 for one time the mainlobe. If the distance is specified as a string then the unit is specified, and

currently only meters are supported; a distance of `100m` is 100 meters in both azimuth and range. If two or more outputs are requested, the algorithm is repeated for points that are not selected to create independent subsets. This is possibly useful for testing purposes.

`[sub,sub2,...] = pointset.subset('mindist',d,'seedpoints', points)` Make sure the given seedpoints are present in the subset. Points can be given as either PointSet, LxxxxxPxxxxx or [lines, pixels].

`[sub,sub2,...] = pointset.subset('mindist',d,'seedpoint', points, 'onlytoseedpoints')` Only consider the distance to the given seedpoints, and don't include the seedpoints themself.

5.18 `export_csv`

Export PointSet data to CSV

`pointset.export_csv(file, props)` Export the properties in the cell array `props` to `file`. Each string in props should be a property, with optionally a printf format string appended. Example: {`pnt_line`,`pnt_pixel`,`pnt_lat%0.10g`,`defo%.3g`}

`pointset.export_csv(..., 'noheader')` Don't include column headers.

5.19 `plot_interferograms`

TODO: improve and document

5.20 `plot_latlon`

TODO: document

`gridsize` : grid size in [m] `range` : plot range for colorbar `colormap` : colors to use `function` : e.g. @mean or @abs etc [default @max]

5.21 `newfig_mrm`

TODO: document

5.22 `export_KML`

Exports a transparant plot and a kml file that locates it TODO: document

`subset` points that should be displayed (indices) `raster` Raster object

Chapter 6

Class: `Filter`

Spectral metadata and filter class A `Filter` object contains information about the spectrum of a signal, and creates and applies filters taking this information into account. The spectrum is defined as a relative spectrum, which is in the interval [-1 1]. It is defined by a window shape, a bandwith, and a phase shift.

Filter objects are immutable.

`f = Filter(windowAmp, bandwidth, phaseshift, windowFreq)` Return a new Filter object. If any input is omitted the following defaults are used: windowAmp = [1 1]: rectangular window. bandwidth = 1: full bandwidth. phaseshift = 0: spectrum centered. windowFreq = linspace(-1, 1, length(windowAmp)): equally spaced window amplitudes.

6.1 Properties

`windowAmp` Amplitudes of window shape

`windowFreq` Frequenies of window shape, from -1 to 1.

`bandwidth` Bandwidth of signal, relative to nyquist. From 0 to 1.

`phaseshift` Phase shift of center of spectrum.

6.2 `shape`

Return the shape of the full spectrum.

`[a,f] = filter.shape()` Return the amplitudes and frequencies of the spectrum in full resolution.

`[a,f] = filter.shape(f_in)` Return the spectumr interpolated to the given frequencies `f_in`.

If called on filter array, return a and f as cell arrays.

6.3 design

Design a FIR filter with the frequency response according to self.

6.4 apply

Apply one spectrum to another, which in fact is multiplication.

6.5 overlap

Return the spectrum that must be applied to get the overlapping spectrum.

6.6 rewindow

Return the spectrum that must be applied to get the given window.

6.7 oversample

Return the spectrum of the oversampled signal.

6.8 shift

Change phaseshift.

6.9 noisebandwidth

Calculate the equivalent noise bandwidth of the filter.

6.10 mainlobewidth

Calculate the width of the mainlobe in fft bins (pixels). The width is defined as up to -3 db (Cumming p.35)

6.11 sideloberatio

Calculate the integrated sidelobe ratio in dB. The width of the main lobe is defined as null-to-null.

6.12 filter

Apply the filter to an image.

6.13 raisedCosine

Raised cosine or generalized Hamming window.

6.14 impulseresponse

Return the impulse response.

irf = filter.impulseresponse(maxdist, oversample) Return the impulse response up to `maxdist` samples, oversampled with the given value.

6.15 simulateSidelobes

simulates the sidelobe positions and values of an impulse response with the defined spectrum

Chapter 7

Class: `Poly3D`

3D polynomial, with shift and scale for numerical stability.

7.1 Properties

`xshift` Shift in 1st dimension

`xscale` Scale of 1st dimension

`yshift` Shift in 2nd dimension

`yscale` Scale of 2nd dimension

`zshift` Shift in 3rd dimension

`zscale` Scale of 3rd dimension

`poly` Array with polynomial values

7.2 `Poly3D`

Create a Poly3D object.

`p = Poly3D( xshift, xscale, yshift, yscale, zshift, zscale, poly )` Create a Poly3D object with given parameters.

7.3 `evaluate`

Evaluate the polynomial at the given values.

`val = poly3d.evaluate(x, y, z)` Evaluate the polynomial at the given values. If poly3d is an array of Poly3D objects all polynomials are evaluated and the results concatenated. It is allowed to give a single n-by-3 array as input.

Chapter 8

Class: `Raster`

The `Raster` class defines a 2D rectangular window with equidistant divisions in both dimensions. The dimensions are called `line` and `pixel`, according to their names in radar sciences.

`r = Raster(firstLine:stepLine:lastLine, firstPixel:stepPixel:lastPixel)`
Returns an inclusive window (lastLine and lastPixel are included).

`r = Raster.ex(firstLine:stepLine:endLine, firstPixel:stepPixel:endPixel)`
Returns an exclusive window (endLine and endPixel are excluded).

stepLine and stepPixel default to 1 when they are omitted.

8.1 Properties

`firstLine` First line included in the raster

`firstPixel` First pixel included in the raster

`stepLine` Stepsize in line direction

`stepPixel` Stepsize in pixel direction

`nLines` Number of raster elements in line direction

`nPixels` Number of raster elements in line direction

`lastLine` Last line included in the raster

`lastPixel` Last pixel included in the raster

8.2 `isInside`

`sel = raster.isInside(coords)` Returns true for coordinates inside the span of the raster. `coords` should be a n-by-2 array [line,pixel]

8.3 contains

b = raster1.contains(raster2) Returns true if raster2 is fully within the span of raster1.

8.4 index

[I,J] = raster.index(lines,pixels) Returns the indices of the raster cells containing the coordinates (lines,pixels).

I = raster.index(lines,pixels) Returns linear indices.

see also: locate

8.5 locate

[line, pixel] = raster.locate(I) Returns the location of the linear indices I.

[line, pixel] = raster.locate(I,J) Returns the location of the indices (I,J).

see also: index

8.6 oversample

new = raster.oversample(ovs) Returns a copy of the raster oversampled ovs times. If ovs has two elements ovs(1) is used in line direction, and ovs(2) in pixel direction.

8.7 resample

new = raster.resample(nLines, nPixels) Returns a raster with the same span, but containing nLines line elements and nPixels pixel elements.

8.8 bbox

new = raster.bbox(lines, pixels) Returns a raster containing all (lines,pixels) coordinates, while keeping the stepsize.

8.9 setSteps

new = raster.setSteps(steps) Returns raster with same span, but with stepsize equal to steps. If steps has two elements steps(1) is used in line direction, and steps(2) in pixel direction. If steps is a raster the stepsizes of that raster are used.

8.10 addBorder

new = raster.addBorder(border) Returns a raster with a border of width border added to all sides. If border has two elements border(1) is used in line direction, and border(2) in pixel direction.

8.11 union

new = union([raster1, raster2, ...]) Return a raster spanning all given rasters. Only works for rasters with same stepsize and compatible start points.

8.12 intersection

Returns intersection of rasters

newraster = intersection([raster1, raster2, ...]) Return Raster containing only points that are in all given rasters.

8.13 repr

s = raster.repr() Return a string representation of the object.

8.14 makeTiles

tiles = raster.makeTiles(target_elements) Returns an array of Raster objects spanning the same area as raster. target_elements is the approximate number of elements in each Raster in both directions, and will be rounded to make it fit. If target_elements has two elements target_elements(1) is used in line direction, and target_elements(2) in pixel direction.

tiles = raster.makeTiles(target_elements, roundfact) The same, but round the element count to multiples of roundfact. If roundfact has two elements roundfact(1) is used in line direction, and roundfact(2) in pixel direction.

Chapter 9

Class: `RasterData`

Class for 2D data with coordinate information.

9.1 Properties

`raster` Coordinates and spacing of the data. Raster object.

`data` Data array.

`descr` Description string.

`orientation` Orientation, 2 element array [x_reverse, y_reverse].

`scale` Scale to convert to [m] or other convenient unit.

9.2 `interp`

Return the values at the given coordinates.

`val = imc.value(lines, pixels)` Return the values at the given coordinates. The values are interpolated using a cubic kernel if they are not exact on data points.

`val = imc.value(raster)` Return the values at the raster coordinates. The values are interpolated using a cubic kernel if they are not exact on data points.

9.3 `value`

Return the values at the given coordinates.

`val = im.value(lines, pixels)` Return the values at the given coordinates. The values are interpolated using a cubic kernel if they are not exact on data points.

9.4 cropTo

Crop the data to the given raster.

`im = im.cropTo(raster)` Crop the data to raster, where raster is a `Raster` object.

9.5 mean

Return mean of a stack.

9.6 std

Return std of a stack.

9.7 mean_abs

Return mean of the absolute values of a stack.

9.8 std_abs

Return std of the absolute values of a stack.

9.9 show

Show image or stack of image using `Image` class.

9.10 select_polygon

select a polygon with self as background

Chapter 10

Class: `hdf5prop`

Class for transparent file data access. Matlab class to create and access HDF5 datasets transparently as Matlab variable. Data can be accessed and written with subscript referencing and assignment methods, just like a matlab variable, only size must be explicitly set or changed.

`prop = hdf5prop(file, dataset)` Creates a `hdf5prop` object for the given dataset in the given HDF5 file.

`prop = hdf5prop(file, dataset, mode)` Specify the access mode, mode = `r` : readonly access (default) mode = `rw`: readwrite access

`prop = hdf5prop(file, dataset, parameter, value, ...)` Creates a `hdf5prop` object for a new dataset with given parameters.

parameter `size` - size of the dataset. Necessary. `chunk_size` - size of the chunks of the dataset. `compression` - 0-9 compression level. `type` - type of the dataset to create. This can be a string such as `double`, in which case the datatype maps to `H5T_NATIVE_DOUBLE`, or it can be a derived HDF5 datatype. `max_size` - the maximum size of the dataset to create. `fill` - the fill value.

10.1 Properties

`file`

`dataset`

`mode`

`propsize`

`dataset_id`

10.2 `close`

Close file if open.

10.3 `set_extent`

Allocate space

`hdf5prop.set_extent(sz)` Extend allocated space to size sz.

Chapter 11

Class: `dependentprop`

Property which is dynamically calculated. Use with empty () to get full evaluated array. Without () to pass the unevaluated property. Indexing works as with normal arrays, only linear or boolean indexing for multidimensional arrays is not allowed.

11.1 Properties

`parent` Object of which the property is dependent.

`func` Function to call on parent to calculate property data. The function should take care of indexing.

`sizefunc` Function to call on parent to calculate property size. Should return array with size in each dimension.

11.2 `dependentprop`

Create a `dependentprop` object.

`prop = dependentprop(parent, func, sizefunc)` Creates a `dependentprop` object with given parameters.