Land Deformation Monitoring Using Synthetic Aperture Radar Interferometry

Yin Tiangang

Department of Electrical and Computer Engineering

National University of Singapore

A thesis submitted for the degree of

Master of Engineering

June 2011

Abstract

Most areas of Southeast Asia are located at the junction of four of the world's major plates, namely the Eurasian, Australian, Philippines and Pacific plates. The many interactions occurring at the edges of these plates result in a hazard-active environment with frequent ground deformation. Tsunamis, earthquakes, and volcanic eruptions kill a lot of people every year in Indonesia. Therefore, the study of ground deformation of Southeast Asia has attracted great attention in recent years.

Synthetic aperture radar interferometry (InSAR) and differential interferometry (DInSAR) techniques have been successfully employed to construct accurate elevation and monitor terrain deformation. This technique makes use of active radar transmitters and receivers to monitor ground deformation. The measurement of phase differences at the same ground surface pixel is the main signal processing method. Since the C and X bands have poor signals over vegetation areas, the utilization of L-band SAR data overcomes the challenge of the low coherence over the rainforest areas of Southeast Asia.

In this dissertation, the general SAR processing technique is first introduced. Secondly, ground deformations related to seismic and volcanic processes are discussed in several areas of Southeast Asia (the 2009 Papua and 2009 Haiti earthquakes, and the Merapi and Luci volcanos). Following a discussion of the difficulties of INSAR in Southeast Asia, a new method for baseline correction, is presented. This method introduces the new idea that relative satellite position can be estimated from the interferometry results. By applying this method, orbital inaccuracy is calibrated iteratively, and the standard deviation substantially decreases within a few iterations. The method has good potential in platform position correction and the accuracy improvement of deformation monitoring in Southeast Asia.

Acknowledgements

I would like to acknowledge my former group leader Dr. Emmanuel Christophe for his excellent guidience to this project, especially the programming in C++ and the integration with Python, hope he will have a wonderful life in Google.

I would like to express my deep gratitude to Professor Ong Sim Heng and Doctor Liew Soo Chin, my supervisors, for their guidance, support and supervision. I learned quite a lot on image processing techniques from Professor Ong and remote sensing knowledge from Dr. Liew.

I also want to thank my group mates Mr Chia Aik Song and Miss Charlotte Gauchet. We had a very good time working together to explore SAR interferometry.

I would lie to thank Mr Kwoh Leong Keong, our research center director, and Mr Mak Choong Weng, our ground station director for their support in software purchasing and data ordering.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The shape of the Earth changes over time. The changes due to external sources such as gravity, and internal sources such as the energy transfer by heat convection from the subsurface. There are periodic and nonperiodic changes. The Earth's tide is an example of a periodic change, whereas land surface deformation is an example of a nonperiodic change. The nonperiodic changes come about suddenly and cannot be predicted. Land surface deformation can be related to seismology-tectonic processes such as lanslides, earthquakes, and volcano eruptions. Most of these processes are associated with continental plate movements caused by mantle convection, which can be explained by the theory of plate tectonics [6].

By monitoring the displacement continuously through precise positioning and mapping, the rate and the direction of the movement can be determined. Some methods and tools are developed to observe ground deformation by monitoring the movement of the objects on the Earth's surface. However, limited techniques

and environmental specificities are the main constraints in this research. The Global Positioning System (GPS) is a space-based technique that can monitor ground deformation, but it is difficult and expensive to set up a wide range of ground control points that cover every part of a country. Inactive satellite imaging is able to monitor ground deformation in wide areas by building up 3D optical model, but it only works in the day time without cloud coverage.

The Interferometric Synthetic Aperture Radar (InSAR) is an active observation method that complements the limitation of the direct observation method as mentioned. As it was developed based on remote sensing techniques, InSAR relies on a sensor platform system. For ground deformation studies, the spaceborne InSAR system with the sensor mounted in a space satellite is the most favorable approach.

InSAR is useful to estimate deformation phase to support the study of land deformation. Most of the studies in the past were conducted in high latitude regions with temperate climates. Since interferometry requires good coherence between images, doing InSAR in low latitude areas is challenging since the land cover changes rapidly due to the tropical climate. The atmosphere above tropical regions usually contains water vapor that affects the phase of the microwaves. In addition, the existence of many islands is a limitation for terrain information extraction. Besides these external problems, there are also the internal problems with the satellite. For example, the lack of accurate platform positioning results in imperfect phase removal of the Earth's surface.

Therefore in this dissertation, the problems of monitoring natural hazards on ground deformation in Southeast Asia are discussed, and new methods to solve the problems are presented. Based on the result described here, several paper has been published by CRISP SAR group ([7] [8] [9] [10].

This dissertation includes seven chapters.

The introduction and the objectives are included in Chapter 1. An overview of the radar system, data format, and applications are described in Chapter 2. This chapter focuses on the selection of L band system to overcome the problem of backscatter quality over vegetation areas in Southeast Asia. The data formats of SAR (CEOS) of different levels are listed. Furthermore, the applications of SAR are introduced, based on the data processed over Southeast Asia, including ship detection, polarimetry and interferometry.

The research on SAR interferometry can be divided into two parts:

- Chapter 3 and Chapter 4 introduce the fundamental concepts and methods of interferometry and differential interferometry. MATLAB scripts are used to explain the basics of interferometry, and PYTHON scripts for integrating the software running under LINUX system. Some deformation results over Southeast Asia are analyzed and the problems encountered are discussed in detail. The main problems for interferometric processing are baseline inaccuracy and atmospheric effects. Correcting the baseline is difficult in Southeast Asia because of limited land cover, so the creative part of this

dissertation is on orbit determination approaches.

- Chapter 5 compares the source codes and algorithms of the available softwares. Chapter 6 introduces a new method to solve the crucial baseline problem. This new method extends the baseline concepts to the relative platform positions. Therefore, if multi-pass scenes are available, the platform positions can be calibrated globally to prevent the loss of information.

Finally, Chapter 7 summarizes the research and presents conclusions.

Chapter 2

Introduction to synthetic aperture radar (SAR)

This chapter gives an introduction to the technical part of SAR. Firstly, a brief description of the SAR system will be presented. Secondly, the ALOS PALSAR system, which is the main data used in this research, is introduced. Lastly, the processing steps of the raw data are listed in Appendix A. The applications of SAR are discussed in Appendix B.

2.1 Synthetic aperture radar (SAR)

As a remote sensing technique, the platform needs to be mounted on a carrier, so the concepts of airborne (carried by airplane) and spaceborne (carried by satellite) are dealt with separately. Considering a remote radar imaging system

in a spaceborne situation, the spacial resolution has the following relationship with the size of the aperture (antenna) from the Rayleigh criterion:

$$\Delta l = 1.220 \frac{f\lambda}{D} \tag{2.1}$$

where f is the distance from the satellite platform to the target on the ground (normally several hundreds of km), λ is the wavelength (in cm range), and D is the antenna size. With the conventional concept of beam-scanning, the antenna size needs to be around thousands of meters in order to achieve an acceptable resolution of several meters. This criterion cannot be satisfied with current technology, which led to the recent development of synthetic aperture radar (SAR). SAR is a form of imaging radar that uses the motion of the aircraft/satellite and Doppler frequency shift to electronically synthesize a large antenna so as to obtain high resolution. It uses the relative motion between an antenna and its target region to provide distinctive long-term coherent-signal variations that are exploited to obtain finer spatial resolutions. As at 2010, airborne systems (on aircraft) can provide a resolution down to about 10 cm, and about 1 m with spaceborne systems.

Since SAR is an active system, it contains a wide distribution of wavelengths in radio frequencies (Table 2.1). These bands have excellent atmospheric transmission. Figure 2.1 shows that the atmospheric transmittance is almost 1 in all these bands, because the relatively long wavelength has good penetration prop-

erty. Therefore, the weather condition has almost no influence on the amplitude of the signal transmission, and the SAR observation has advantages in all these conditions.

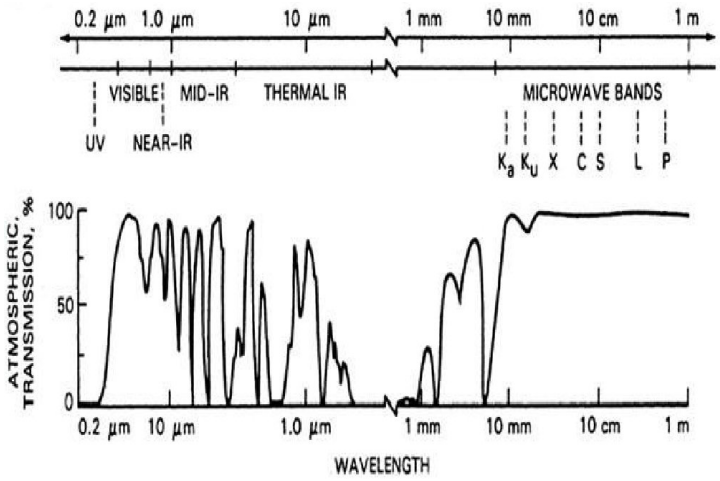


Figure 2.1: Atmospheric penetration ability of the different wavelength

Table 2.1: Radar bands and wavelength

P-Band	$30 - 100$ cm
L-Band	$15 - 30$ cm
S-Band	$7.5 - 15$ cm
C-Band	$3.75 - 7.5$ cm
X-Band	$2.4 - 3.75$ cm
K_u-Band	$1.67 - 2.4$ cm
K-Band	$1.1 - 1.67$ cm
K_a-Band	$0.75 - 1.1$ cm

Spaceborne SAR works by transmitting coherent broadband microwave radio signals from the satellite platform, receiving the signals reflected from the terrain, storing and processing the returns to synthesize a large aperture, and focusing the data to form an image of the terrain. Figure 2.2 shows a simple geometry of the SAR imaging system. The basic configuration is based on side-look ge-

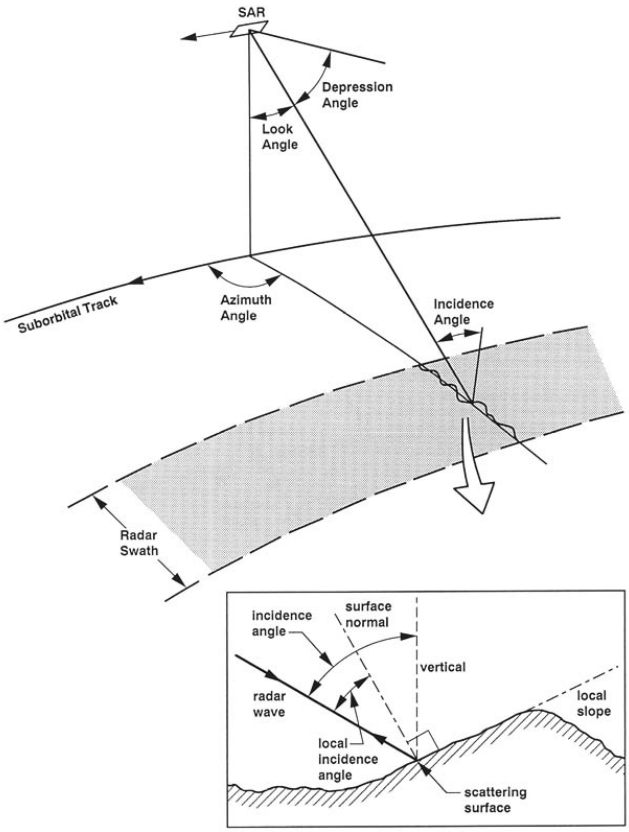


Figure 2.2: SAR imaging geometry [1]

ometry, where the satellite is traveling in a nearly horizontal direction (azimuth direction), and the radar wave transmission direction (slant range direction) is almost perpendicular to the azimuth direction. The swath width refers to the strip of the Earth's surface from which data are collected by a satellite. There are two more angles that need to be differentiated. The look angle (off-nadir angle) refers to the angle generated by the ordered lines connecting the three points: the center of the Earth, the satellite position, and the target. The incident angle uses the same points in a different order: satellite position, target, and reverse of the direction to the center of the Earth. Under the assumption of a flat earth, these

two angles are the same, but in an accurate orbital interferometric system they need to be estimated separately. Further details will be discussed in Chapter 5.

Table 2.2: Current and future spaceborne SAR systems

Name	Launched	Ended	Country	Band	Polarization
Seasat	1978	1978	USA	L-band	Single (HH)
ERS-1	1991	2000	Europe	C-band	Single (VV)
JERS-1	1992	1998	Japan	L-band	Single (HH)
ERS 2	1995		Europe	C-band	Single (VV)
Radarsat-1	1995		Canada	C-band	Single (HH)
Space Shuttle SRTM	2000	2000	USA	X-band	Single (VV)
Envisat ASAR	2002		Europe	C-band	Single, Dual (Altenating)
RISAT	2006		India	C-band	Single, Dual, Quad
ALOS	2006		Japan	L-band	Single, Dual, Quad
Cosmo/Skymed (2+4x)	2006		Italy	X-band	Single, Dual
SAR-Lupe	2006		Germany	X-band	Unknown
Radarsat-2	2007		Canada	C-band	Single, Dual, Quad
TerraSAR-X	2008		Germany	X-band	Single, Dual, Quad
TecSAR	2008		Israel	X-band	Unknown
TanDEM-X	2009		Germany	X-band	Single, Dual, Quad
Kompsat-5	2009		South Korea	X-band	Single
HJ-1-C	2009		China	S-band	Single (HH or VV)
Smotr	2010		Russia	X-band	Unknown
Sentinel-1	2011		Europe	C-band	Single, Dual, Quad
SAOCOM-1	2012		Argentina	L-band	Single, Dual, Quad
MapSAR	2012		Brazil + Germany	L-band	Single, Dual, Quad
ALOS-2	2012		Japan	L-band	Single, Dual, Quad

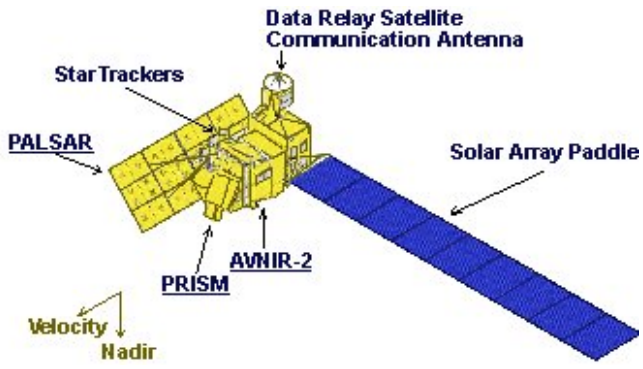
Many countries have set up their satellites with spaceborne SAR. This technique is also widely used for Moon, Mars and Venus terrain generation, water detection, and mineral detection. Table 2.2 lists current and future spaceborne SAR systems. The SAR signal is always measured from the polarization in the H (horizontal) and the V (vertical) directions. HH means the signal is transmitted in H polarization and received in V polarization. A similar concept is applied for VV, HV and VH. There are three modes of polarization: single, dual and quad. Single has the highest resolution but least polarimetry information, and quad has the lowest resolution but contains all the four polarimetries. In this

dissertaion, ALOS PALSAR of single and dual polarization will be utilized as the main observation choices, with some ERS and TerraSAR-X as the supportive materials.

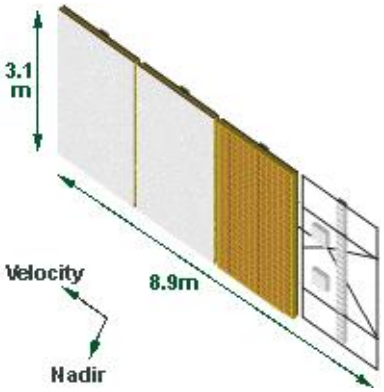
2.2 ALOS PALSAR system

The advanced land observing satellite (ALOS) was launched by the Japan Aerospace Exploration Agency (JAXA) in January 2006. The ALOS has three remote-sensing instruments: the panchromatic remote-sensing instrument for stereo mapping (PRISM) for digital elevation mapping, the advanced visible and near infrared radiometer type 2 (AVNIR-2) for precise land coverage observation, and the phased array type L-band synthetic aperture radar (PALSAR) (Figure 2.3 (a)). PRISM and AVNIR-2 are inactive optical sensors, which can only work with the existence of wave radiation from the Earth's surface (day time), at resolutions of 2.5 m and 10 m respectively. PALSAR is an active microwave sensor using L-band frequency (23.6 cm wavelength), working for day-and-night and all-weather land observations. The L-band has the advantage of the lowest sensitivity over the vegetation [11], and therefore becomes the most suitable sensor for obtaining the information over highly vegetated areas.

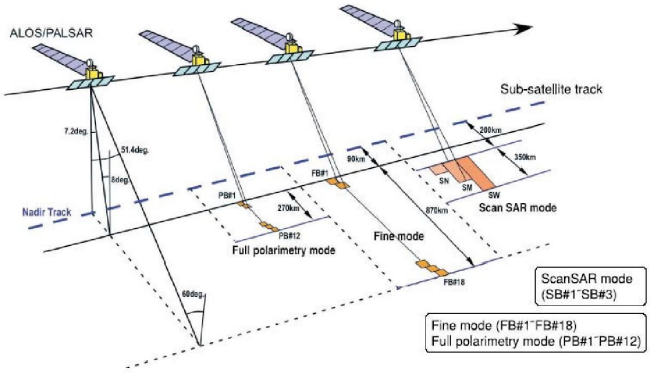
The orbit of ALOS is sun-synchronous, with a repeat cycle of 46 days over the same area. The spacecraft has a mass of 4 tons, and works at altitude of 691.65 km at the equator. The attitude determination accuracy is 2.0×10^{-4} degree



(a)



(b)



(c)

PALSAR Characteristic

Mode	Fine		ScanSAR	Polarimetric (Experimental mode)*1
Center Frequency	1270 MHz(L-band)			
Chirp Bandwidth	28MHz	14MHz	14MHz,28MHz	14MHz
Polarization	HH or VV	HH+HV or VV+VH	HH or VV	HH+HV+VH+VV
Incident angle	8 to 60deg.	8 to 60deg.	18 to 43deg.	8 to 30deg.
Range Resolution	7 to 44m	14 to 88m	100m (multi look)	24 to 89m
Observation Swath	40 to 70km	40 to 70km	250 to 350km	20 to 65km
Bit Length	5 bits	5 bits	5 bits	3 or 5bits
Data rate	240Mbps	240Mbps	120Mbps,240Mbps	240Mbps
NE sigma zero *2	< -23dB (Swath Width 70km) < -25dB (Swath Width 60km)		< -25dB	< -29dB
S/A *2,*3	> 16dB (Swath Width 70km) > 21dB (Swath Width 60km)		> 21dB	> 19dB
Radiometric accuracy	scene: 1dB / orbit: 1.5 dB			

(d)

Figure 2.3: PALSAR devices and modes. (a) ALOS devices configuration; (b) PALSAR antenna; (c) PALSAR observation modes; (d) Characteristics of observation modes (published in [2]).

with ground control points, and position determination accuracy is 1 m. There is a solid-state inboard data recorder of 90 Gbytes, and data is transferred to ground station either at a rate of 240 Mbps (data relay), or at 120 Mbps (direct transmission). The other important instrument installed in ALOS is the attitude and orbit control subsystem (AOCS) that acquires information on satellite attitude and location. By utilizing this device, high accuracy in satellite position and pointing location on the Earth's surface can be achieved.

Figure 2.3 (b) shows the PALSAR antenna device, which has a rectangle shape with a length of 8.9 m and a width of 3.1 m. The length direction is kept the same as the direction of the projecting satellite. Since the Earth takes the shape of an oblate spheroid, the look angle ranges from 9.9° to 50.8°, with corresponding incident angle from 7.9° to 60°. 80 transmission and receptions modules on four segments are used to process the single, duet, and quad polarimetry signals.

PALSAR has the following four kinds of data as described in Figure 2.3 (c) and (d):

- High resolution mode

  This mode is normally used for SAR pattern detection and interferometric processing. It is most commonly used in regular operation, with the look angle changing from 9.9° to 50.8°. It can be divided in two different types: fine beam single (FBS) polarization (HH or VV) and fine beam dual (FBD) polarization (HH + HV or VV + VH). The maximum ground resolution of

FBS is about 10 m × 10 m, whereas FBD has 20 m × 20 m resolution.

- ScanSAR mode

  The ScanSAR mode enables a adjustable look angle which is 3 to 5 times higher than fine beam model. It means that this mode can cover a wide area, from 250 km to 350 km swath width, but the resolution is inferior to high resolution mode which is approximately 100 m × 100 m.

- Direct downlink

  The direct downlink mode, which is also known as direct transmission (DT) mode, is employed to accommodate real time data transmission of single polarization. This observation mode is similar to high resolution single polarization mode but has a lower ground resolution of approximately 20 m × 10 m.

- Polarimetric mode

  This mode is used for polarimetry processing and classification. The polarimetry observation mode enables PALSAR to simultaneously receive horizontal and vertical polarization for each polarized transmission, with the look angle changing from 9.7° to 26.2°. This observation mode has 30 m × 10 m ground resolution for a 30 km swath width.

There are five levels of PALSAR standard product, called Level 1.0, Level 1.1, Level 1.5, Level 4.1 and Level 4.2. This classification is based on the processing level and observation mode. In this dissertation, the main focus is on complex

phase interferometric processing. Level 1.5, Level 4.1, and Level 4.2 are mainly amplitude processing without phase information. Therefore, the processing from Raw (Level 1.0) to single look complex (SLC) (Level 1.1) will be illustrated.

- Level 1.0 is also called the raw data. Normally Level 1.0 is just unprocessed signal data with radiometric and geometric correction coefficients. The product is not yet subjected to the recovery process of SAR. From the raw data, we cannot recognize any pattern or phase information of SAR before a series of processing steps. The data type is in 8-bit unsigned integer and is available in separate files for each polarization.

- Level 1.1 is a single look complex data (SLC), which basically requires the matched filtering of the raw data in range and in azimuth with corresponding reference functions. SLC is equally spaced on slant range compressed in range and azimuth directions. The data, basically, is complex valued with amplitude and phase information. Level 1.1 is SAR recovery processing of Level 1.09 data. The data type is in IEEE 32-bit floating point and available in separate files for each polarization.

The detailed processing steps from raw (Level 1.0) to SLC (Level 1.1) are shown in Appendix A. Since the amplitude and phase information are well described using SLC data, there are mainly three types of application. Ship detection, polarimetry and interferometry are very popular research topics in Southeast Asia, because of the ocean surface, the vegetation and the frequent natural haz-

ards that are present. For these applications, the examples processed by CRISP are shown in Appendix B.

Chapter 3

SAR interferometry processing

SAR interferometry or InSAR was developed to derive the topographic map for an area or the height for one particular point on the Earth's surface. In this chapter, the basic method of generating interferograms will be introduced with a MATLAB scripting approach. Secondly, a brief introduction to the GAMMA software we use is presented. Lastly, the steps to build a topographic height map from the interferogram will be looked in detail.

3.1 Interferogram generation from MATLAB scripting

Generally, interferogram is obtained by phase subtraction of two SAR SLCs (cross-multiplication of the two complex numbers). The phase subtraction result will provide 3D information at the the corresponding target. The following

calculation is performed on coregistered images in the form of the complex quantity $I(m,i)$ of the generated interferogram:

$$I(m,i) = \frac{\sum\limits_{cell} M(m,i) \cdot S^{*}(m,i)}{(\sum\limits_{cell} |M(m,i)|^2)^{1/2} \cdot (\sum\limits_{cell} |S(m,i)|^2)^{1/2}} \tag{3.1}$$

where M (for master) and S (for slave) represent the two radar images whose pixels are indexed by m and i, while cell is the analysis window, that is, the set of adjacent pixels included in neighborhood averaging. The phase image of $I(m,i)$ or the interferogram itself will be saved in a data file, and color is used to represent the phase cycle. The MATLAB code used to generate an interferogram with two CEOS Level 1.1 images is given in Appendix F.

3.1.1 SAR image generation

The SLC data of ALOS can be directly ordered or generated from raw data processing as mentioned above. The data format can be found from the JAXA website [3]. The binary image file and leader file are processed and specific parameters are displayed, including the number of samples per data group (column number) and the number of records (row number -1). The specific flag of ascending or descending is checked to make sure both platforms are moving in the same direction. The orbit state vectors are extracted and saved. The binary data arrangement follows the pattern $M(m,i)_{real}, M(m,i)_{imag}, M(m+1,i)_{real}, M(m+1,i)_{imag}...$, so the data can be saved in complex matrices such that $I(m,i) = M(m,i)_{real} +$

$j \times M(m,i)_{imag}$. Since the image is relatively large for MATLAB, which can only use 2 GB memory in a 32-bit PC, an overview image will be generated first with one pixel from each 15×15 image area.

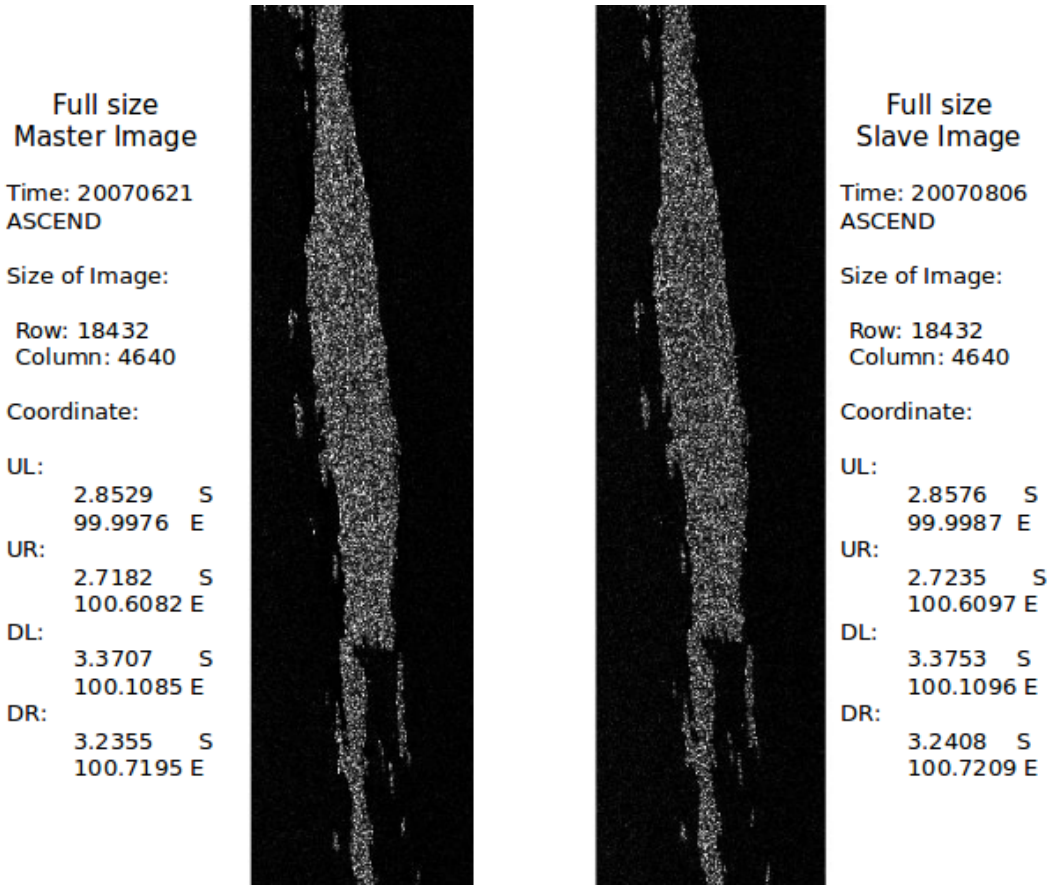


Figure 3.1: Overview of master and slave image over Singapore

Figure 3.1 shows the overview of the master and the slave images of Pagai Island (Indonesia) generated using *AlosBasic.m*. The temporal baseline is less than three months, so a good coherence map can be expected.

A certain area of the master image needs to be cropped from the full scene because of the memory limitation of MATLAB. This process is done with *AlosM.m*. The four image coordinates of a certain area will be selected. The ascending and

descending property of passes is important because they have upside-down image coordinates. After running this script, this area will be saved as a matrix and displayed in the MATLAB image viewer.

3.1.2 Registration

Each SAR pixel is a combination of amplitude and phase. The amplitude which shows obvious features and patterns can be used to register two SAR images. Later, a simple calculation on the phase difference over the corresponding points will show a fine interferogram. The registration part is very important. Coherence is a property of waves that enables stationary (i.e, temporally and spatially constant) interference. In registered master and slave images, if the pixels have a 1/8 pixel mis-registration, the coherence of interferogram will drop by 1/2. Therefore, the accuracy is not good enough to be detectable by the naked eye to select feature control points, and much time will be wasted. To achieve the best accuracy automatically, the method contains two steps to accomplish the registration.

- Coarse Registration:

  The sensor model will be used to find the corresponding points between images. It helps to find the corresponding geocoordiate from a specific image pixel, and vice versa. The general process is:

$$ImagePixel(M) \rightarrow Geocoordinate \rightarrow ImagePixel(S)$$

In the traditional method, for a fixed image parameter from the leader file, the sensor model [12] is based on five variables: p, l, h, Φ, Λ, where p and l are the pixel indices, h is the height of the location with geocoordinate latitude Φ and longitude Λ. This model is relatively complicated, and an easier way is found from the ALOS PALSAR data format description. Figure 3.2 from the file of Level 1.1 data format shows that the $8th$ order polynomial model contains 25 coefficients.

It can be seen that this model does not require height input, so a comparison can be made between the physical model (with height) and the polynomial model (without height) (Figure 3.2). Several observation points are selected near Merapi Volcano, Indonesia (Figure 3.2 (a)), with the height information from the Shuttle Radar Topography Mission (SRTM). Figure 3.3 (b) shows the comparison result. With terrain information, the difference can be as large as hundreds of pixels, which is proportional to the terrain height. On the other hand, the offset is less than 5 pixels if terrain information is set to zero. In conclusion, the polynomial model is a simulation of the coordinates without height information, but we can still use it for registration between two SAR images because of the compensatory processing between reverse translations. The translations are done using $PixToCoor.m$ and $CoorToPix.m$. Because of the differences in look angle, the error is equal to the baseline divided by platform altitude, about 1/1000 of the offset (less than 1 pixel).

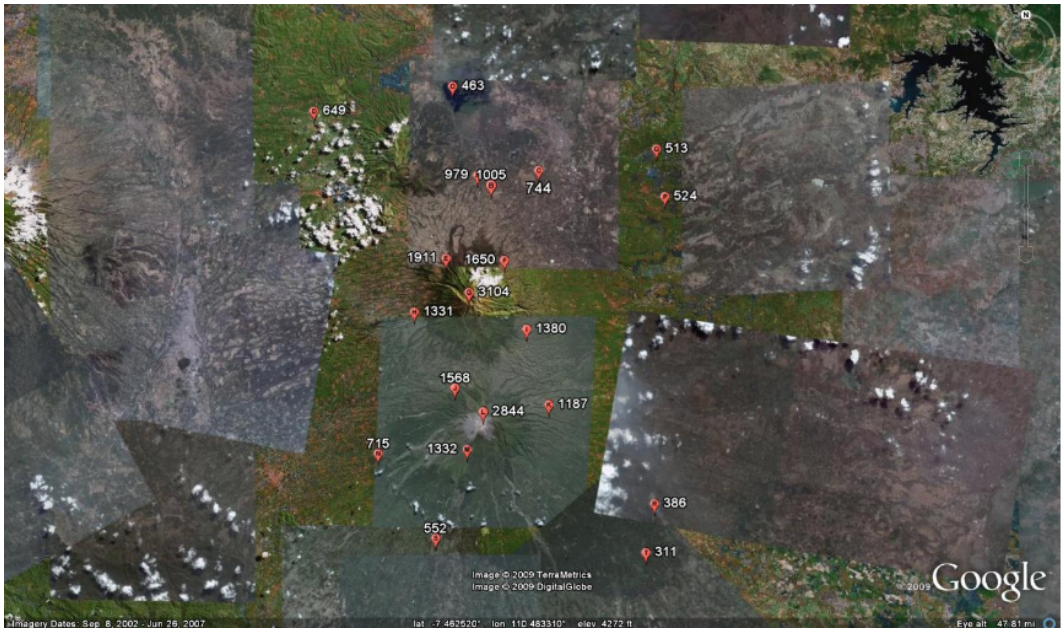
Bytes	Format	Description	Contents
2065-3064	**50E20.10**	Coefficients of the 8th polynomial expression to convert from latitude (Φ) and longitude (Λ) to pixel (p) and line (l), say (p, l) where: $p = c_0\Lambda^4\Phi^4 + c_1\Lambda^3\Phi^4 + c_2\Lambda^2\Phi^4 + c_3\Lambda\Phi^4 + c_4\Phi^4$ $+ c_5\Lambda^4\Phi^3 + c_6\Lambda^3\Phi^3 + c_7\Lambda^2\Phi^3 + c_8\Lambda\Phi^3 + c_9\Phi^3$ $+ c_{10}\Lambda^4\Phi^2 + c_{11}\Lambda^3\Phi^2 + c_{12}\Lambda^2\Phi^2 + c_{13}\Lambda\Phi^2 + c_{14}\Phi^2$ $+ c_{15}\Lambda^4\Phi + c_{16}\Lambda^3\Phi + c_{17}\Lambda^2\Phi + c_{18}\Lambda\Phi + c_{19}\Phi$ $+ c_{20}\Lambda^4 + c_{21}\Lambda^3 + c_{22}\Lambda^2 + c_{23}\Lambda + c_{24}$ $l = d_0\Lambda^4\Phi^4 + d_1\Lambda^3\Phi^4 + d_2\Lambda^2\Phi^4 + d_3\Lambda\Phi^4 + d_4\Phi^4$ $+ d_5\Lambda^4\Phi^3 + d_6\Lambda^3\Phi^3 + d_7\Lambda^2\Phi^3 + d_8\Lambda\Phi^3 + d_9\Phi^3$ $+ d_{10}\Lambda^4\Phi^2 + d_{11}\Lambda^3\Phi^2 + d_{12}\Lambda^2\Phi^2 + d_{13}\Lambda\Phi^2 + d_{14}\Phi^2$ $+ d_{15}\Lambda^4\Phi + d_{16}\Lambda^3\Phi + d_{17}\Lambda^2\Phi + d_{18}\Lambda\Phi + d_{19}\Phi$ $+ d_{20}\Lambda^4 + d_{21}\Lambda^3 + d_{22}\Lambda^2 + d_{23}\Lambda + d_{24}$ (The order of storing: c_0, c_1, c_2, ..., c_{24} & d_0, d_1, d_2, ..., d_{24})	(Φ, Λ) refered in the left are substituted by the following expressions as Φ=ϕ-Φo (degrees), Λ=λ−Λo (degrees), where (ϕ, λ) is an arbitrary position on the image. For the expressions, the position defined as (p, l)=(0, 0) corresponds to the central point of the pixel at the upper left corner.
3065-3084	**E20.10**	Origin Latitude (Φo) (degrees)	
3085-3104	**E20.10**	Origin Longitude (Λo) (degrees)	
3105-5000	**CH**	blanks	

(a)

Bytes	Format	Description	Contents
1025-2024	**50E20.10**	Coefficients of the 8th polynomial expression to convert from pixel (P) and line (L) to latitude (ϕ) and longitude (λ), say (ϕ, λ) where: $\phi = a_0L^4P^4 + a_1L^3P^4 + a_2L^2P^4 + a_3LP^4 + a_4P^4$ $+ a_5L^4P^3 + a_6L^3P^3 + a_7L^2P^3 + a_8LP^3 + a_9P^3$ $+ a_{10}L^4P^2 + a_{11}L^3P^2 + a_{12}L^2P^2 + a_{13}LP^2 + a_{14}P^2$ $+ a_{15}L^4P + a_{16}L^3P + a_{17}L^2P + a_{18}LP + a_{19}P$ $+ a_{20}L^4 + a_{21}L^3 + a_{22}L^2 + a_{23}L + a_{24}$ $\lambda = b_0L^4P^4 + b_1L^3P^4 + b_2L^2P^4 + b_3LP^4 + b_4P^4$ $+ b_5L^4P^3 + b_6L^3P^3 + b_7L^2P^3 + b_8LP^3 + b_9P^3$ $+ b_{10}L^4P^2 + b_{11}L^3P^2 + b_{12}L^2P^2 + b_{13}LP^2 + b_{14}P^2$ $+ b_{15}L^4P + b_{16}L^3P + b_{17}L^2P + b_{18}LP + b_{19}P$ $+ b_{20}L^4 + b_{21}L^3 + b_{22}L^2 + b_{23}L + b_{24}$ (The order of storing: a_0, a_1, a_2, ..., a_{24} & b_0, b_1, b_2, ..., b_{24})	(P, L) refered in the left are substituted by the following expressions as P = p - Po, L = l - Lo, where (p, l) is an arbitrary coordinate address on the image. For the expressions above, the position defined as (p, l)=(0, 0) corresponds to the central point of the pixel at the upper left corner and (ϕ, λ) is measured in "degrees".
2025-2044	**E20.10**	Origin Pixel (Po)	
2045-2064	**E20.10**	Origin Line (Lo)	

(b)

Figure 3.2: Polynomial model of SAR [3]. (a) From SAR pixel to geocoordinate (number indicating the elevation in meters from SRTM; (b) From geocoordinate to SAR pixel.

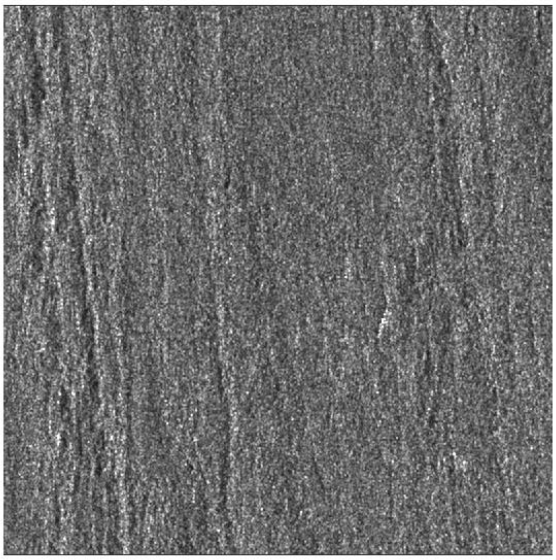


(a) Selection of point candidates

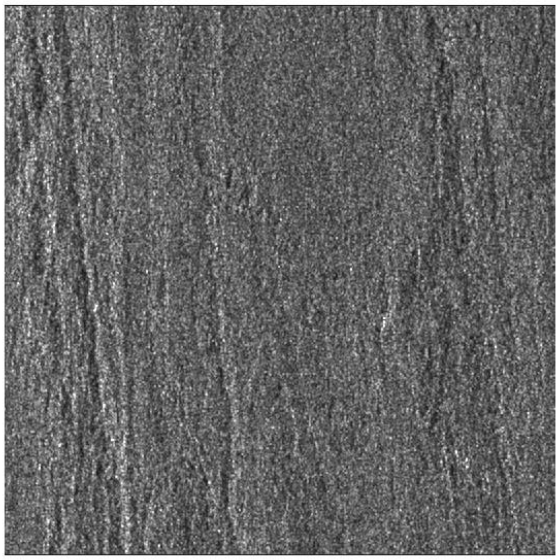
Label	Long,Lat(degree)	SRTM Height (m)	Pix (Polynomial)	Pix (Gamma with height input)		Pix (Gamma without height input)		Difference (With Height)		Difference (Without Height)	
A	110.44208669, -7.36591283476	979	14771 2227	14766	2145	14767	2227	82	5	4	0
B	110.452301163, -7.37485049808	1005	14389 2286	14383	2202	14385	2286	84	6	4	0
C	110.488051816, -7.36208240762	744	14552 2565	14547	2503	14547	2565	62	5	5	0
D	110.318236213, -7.31739409103	649	17364 1419	17359	1364	17359	1419	55	5	5	0
E	110.419104127, -7.428476478	1911	12817 1962	12811	1803	12812	1962	159	6	5	0
F	110.462515635, -7.42975328704	1650	12443 2272	12438	2135	12439	2272	137	5	4	0
G	110.436979454, -7.45401265891	3104	11812 2050	11805	1791	11807	2050	259	5	5	0

(b) Table of comparison between the two models

Figure 3.3: Comparison between geometric model with height information and polynomial model without height information, in Merapi.



(a)



(b)

Figure 3.4: The master and slave image after Lee filtering. (a) Master image; (b) Slave image.

Continue from the MATLAB processing above, *AlosS.m* is used to cut the slave image. The four corner pixels of the selected master image is translated to the corners in the slave image. The resultant rectangle on the slave image may not have the same size as the master image. Many control points are translated using the polynomial model, and bicubic transformation of the image is performed based on these points through *OrbitRegis.m*. In the last step, the slave image is resized to be the same as the master image. The resultant images are less than 20 pixels off in azimuth, and less than 2 pixels off in range.

- Fine registration:

  The Lee filter is applied to both master and slave images before fine registration to reduce the cross noise (Figure 3.4). To get sub-pixel registration, both images will be oversampled by 8 times larger [13] at a restricted area (30×30). By calculating the cross-correlation, the corresponding pixels with the maximum cross-correlation value will be selected as a registered pair.

Thus by using coarse and fine registration, the slave image is well registered with the master image to 1/8 sub-pixel accuracy. Bicubic transformation will be performed again at this point, where both master and slave images are ready for interferogram generation.

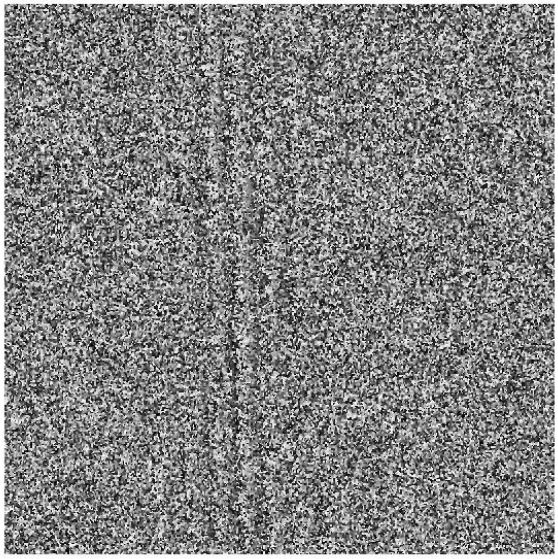
3.1.3 Interferogram

By using Eq. (3.1), phase differences can be calculated at this point. Figure 3.5 shows the interferogram before and after fine registration, where it can be observed that the alternating fringe number greatly increases with fine registration. Better coherence is obviously obtained in phase domain. However, the fringe does not seem to represent the terrain information. A linear phase trend is observed because the flat earth effect has not been removed from the image. The interferogram at this stage contains several phase contributions. These contributions will be discussed in detail in Chapter 4.

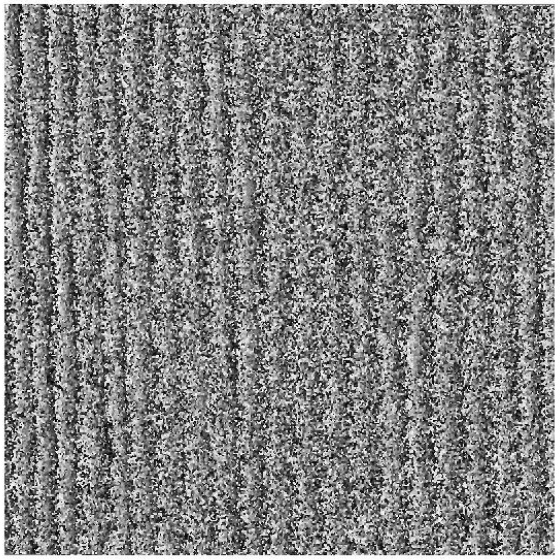
3.2 GAMMA software

The previous processing is based on a MATLAB script. From this stage, GAMMA software is used for extracting terrain information from the interferogram [14]. The GAMMA SAR and Interferometry Software is a commercial software that contains a collection of programs for the processing of SAR, interferometric SAR (InSAR) and differential interferometric SAR (DInSAR) data for airborne and spaceborne SAR systems. The software is arranged in packages, each dealing with a specific aspect of the processing. The processing used in this dissertation includes:

- MSP: SAR data processing



(a)



(b)

Figure 3.5: The interferogram before and after fine registration (phase domain). (a) Before fine registration; (b) After fine registration.

- ISP: Interferometric SAR processing
- DIFF&GEO: Differential interferometric SAR processing and terrain geocoding

GAMMA runs on any Unix or Linux system. After compiling the source codes of the software, Python is used to integrate all the command lines. The Python coding to generate interferogram is shown in Appendix E. *gamma.py* (containing more than 2500 lines) is the library file to be called by the main file. All the functions in this file can be used for all the data in same level. Therefore, in the main script file, only the file name and directory need to be specified. With this approach, the processing steps are clearer.

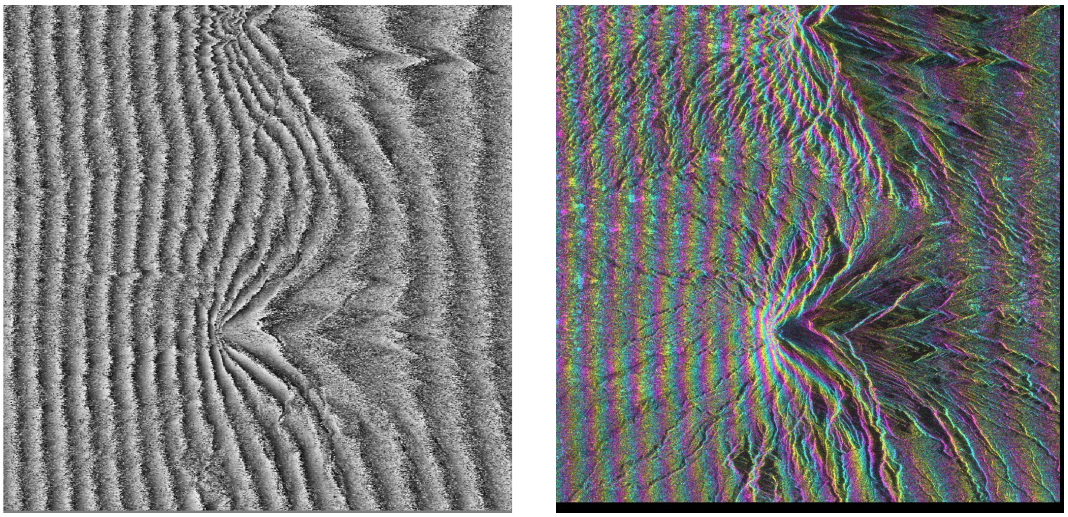
3.3 From interferogram to terrain height

Figure 3.6 shows the interferograms generated by MATLAB and GAMMA respectively on Merapi Volcano. The fringes are almost the same in the phase domain (GAMMA's result has amplitude image as the intensity and color as the phase).

For one interferogram, the following expression is the most accurate model:

$$\Phi = \Phi_{curv} + \Phi_{elev} + \Phi_{base} + \Phi_{atm} + \Phi_{dif} + \Phi_0 \tag{3.2}$$

where Φ_{elev} is the phase contribution of the expected elevation, Φ_{curv} is due



(a) (b)

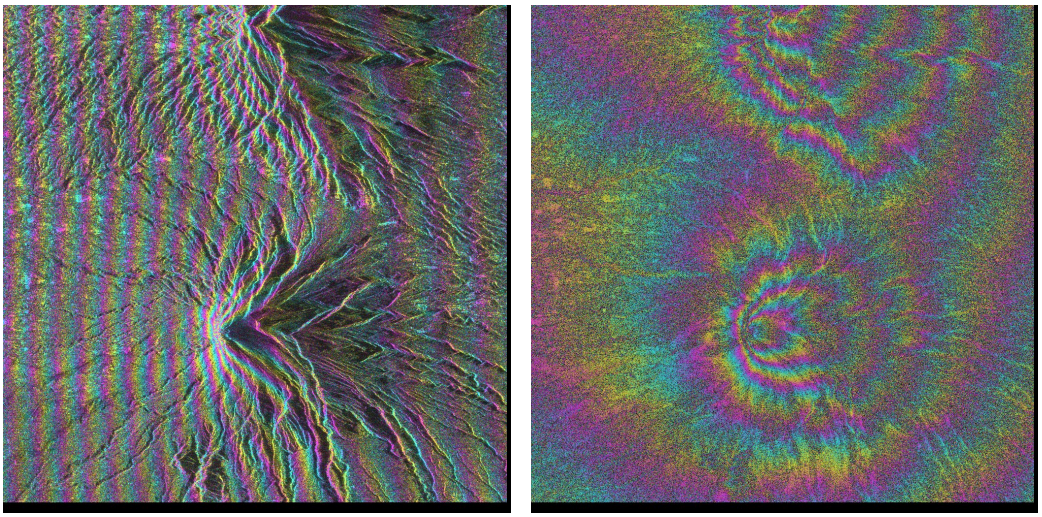
Figure 3.6: The interferogram generated using 20090425(M) and 20080610(S). (a) Interferogram using MATLAB script; (b) Interferogram using GAMMA software.

to the Earth's curved surface, Φ_{base} is the linear phase trend of the flat Earth surface, Φ_{atm} is the atmospheric contribution to the phase, Φ_{def} is the phase of deformation which is not included in Φ_{elev}, and Φ_0 is a constant.

3.3.1 Baseline estimation and interferogram flattening

Among all these terms in Eq. (3.2), the assumption can be made that no deformation occurs ($\Phi_{dif} = 0$) and Φ_{elev} is the desired value.

Φ_{atm} depends on the weather, with two contributions. The first contribution that rarely occurs is the ionospheric effect, which will result in a nonlinear phase trend over the entire interferogram. The other is the residual phase, caused by gathering of water vapor (cloud) [15], which will result in additional phase in a specific area. Normally, both these effects can be recognized clearly from the



(a) Interferogram before flattening (b) Interferogram after flattening

Figure 3.7: The interferogram flattening

interferogram, but are not easy to remove. Therefore they are considered as part of the final interferogram and will cause error in the final elevation model.

In contrast, Φ_{base} can be easily calculated once the baseline is accurately measured. The distance between the two satellite positions is known as the baseline (B), which can be decomposed into the parallel baseline (B_{para}) and the perpendicular baseline (B_{perp}). The parallel baseline is the component along the radar's line of sight while the perpendicular baseline is the component which is perpendicular to the line of sight. The initial baseline estimation is obtained from the orbital interpolation of platform position over the interferogram. This process is done by interpolating the orbit state vectors mentioned in Chapter 2.

Flattening is the process of subtracting Φ_{base} from Φ. The linear fringes on the interferogram are supposed to be removed by flattening. Figure 3.7 shows the result. After removing the linear phase trend, only the elevation phase on

Merapi Volcano is left (atmospheric phase is small and negligible). However, if the baseline has an error, a portion of the linear phase trend will still remain on the interferogram.

Another method for baseline estimation is called the Fast Fourier Transform(FFT) method. Based on the known elevation, a relatively flat area is selected and processed with FFT spectrum calculation. The FFT centroid is eliminated by using a specific baseline to flatten the image. Thus the resultant interferogram has no phase trend at the selected flat area. The detailed process of baseline estimation and flattening will be discussed in Chapter 5.

3.3.2 Filtering and phase unwrapping

The residue refers to a point in the interferogram where the sum of the phase differences between pixels around a closed path is not zero. The number of residues can be effectively reduced by an adaptive filter with the filtering function based on the local fringe spectrum [16]. Figure 3.8 shows the filtered interferogram compared to Figure 3.6 (entire scene). It is obvious that phase noise has been effectively removed and fringes are lost.

The phase unwrapping, which converts the interferogram into a terrain map, is the most crucial step in SAR interferometry. The interferometric phase is wrapped modulo 2π. An integer multiple of 2π has to be added to recover the absolute phase difference, so-called unwrapping. The methods used in this step are the minimum cost flow (MCF) technique and the triangular irregular network

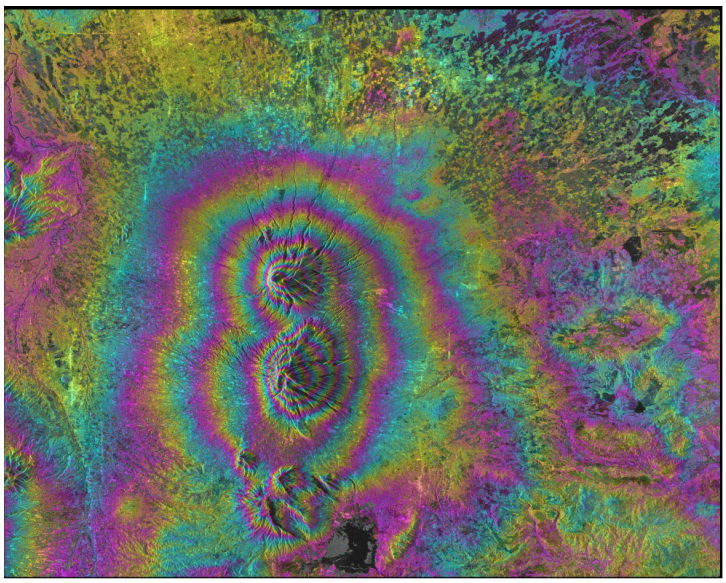


Figure 3.8: Filtered interferogram: 20080425 – 20080610 Merapi

(TIN) technique. The application of MCF techniques to phase unwrapping and thereby achieve a global optimization was first presented in [17]. The possibility for improving the unwrapping by generalizing the network topology to be a triangulation network was proposed in [18]. This technique is a global optimization technique to the phase unwrapping problem.

Figure 3.9 shows the 3D surface construction of unwrapped phase by using MATLAB display tools. The center of image is taken as the relative zero phase as reference.

3.3.3 GCP baseline refinement and computation of height

Although the terrain phase is successfully constructed, the phase map needs to be converted to height before we obtain the digital elevation model (DEM). By doing that, several control points (GCP)[19] with height information need to be

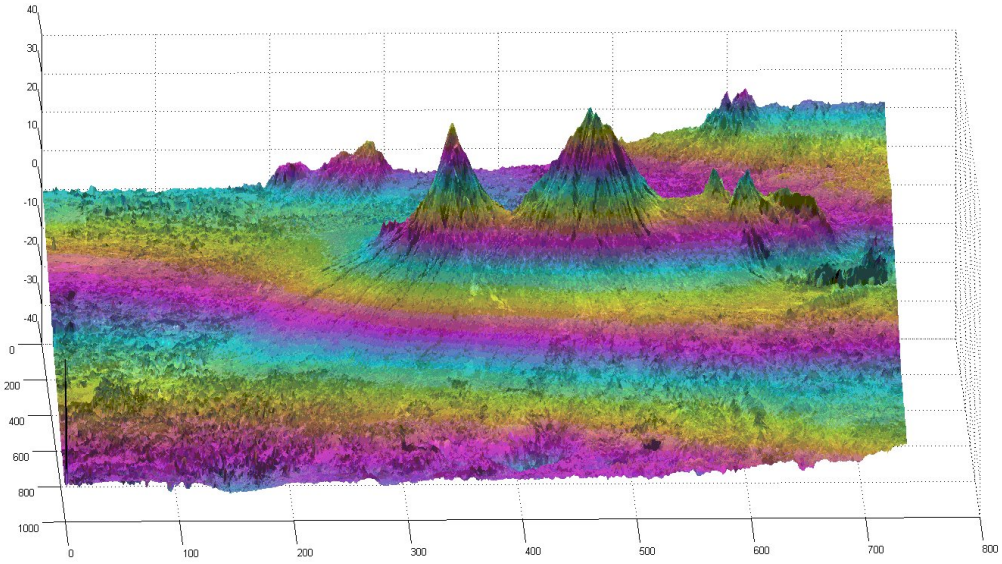


Figure 3.9: Unwrapped phase: 20080425 – 20080610 Merapi

selected to relate the three components: phase, accurate baseline, and height. These three components have the following relationship:

$$\Phi_{elev} = \frac{4\pi B_{perp}}{\lambda R \sin\theta} h + \Phi_C \tag{3.3}$$

where Φ_{elev} is the same component from Eq. (3.2), λ represents the wavelength, R the slant range distance to the point, θ the local incidence angle, and Φ_C the constant added to phase. From Eq. (3.3), using the phase and height of GCPs, a least square regression model can be constructed on the precise perpendicular baseline B_{perp} (zeroth order constant represents Φ_C):

$$B_{perp} = \frac{\lambda R \sin\theta}{4\pi h}(\Phi - \Phi_C) \tag{3.4}$$

The least squares regression of Eq. (3.4) is due to the existence of atmospheric phase screen (part of Φ_{atm} from Eq. (3.2)). The solution typically converges in 3 to 8 iterations. Once B_{perp} and Φ_C are known, Eq. (3.3) is used to calculate h from Φ over the whole image. At this point, the DEM can be successfully generated.

Chapter 4

Volcano and earthquake observation in Southeast Asia

For the interferometric phase, there are several components in Eq. (3.2). The phase contribution of deformation has the following expression:

$$\Phi_{dif} = \Phi - (\Phi_{curv} + \Phi_{elev} + \Phi_{base} + \Phi_{atm} + \Phi_0) \tag{4.1}$$

In Eq. (4.1), all the other contributions can be simulated based on an available elevation model, except for Φ_{atm}. Therefore, the deformation phase can be measured once the interferometric phase after deformation and elevation phase before deformation are available. This is called differential interferometry (DINSAR). In this chapter, DINSAR is presented with the algorithm to monitor ground deformation. Several examples of volcano and earthquake activities in southeast

Asia are monitored. Some of the results have been published in IGARSS 2010, Hawaii [7].

4.1 Differential interferometry (DINSAR)

The basic idea of DINSAR is to subtract the elevation phase from the interferometric phase. Therefore, an elevation model is needed, either from an available DEM database, or another interferogram. There are three ways to generate a differential interferogram:

- 2-pass DINSAR:

  A DEM is required to perform a 2-pass DINSAR. Two scenes are used to generate interferogram after the deformation happens. Before that, the DEM data is taken as the reference. The quality of DEM has a crucial effect on the DINSAR result. Therefore, the DEM with higher accuracy will generate the better differential interferogram.

  The SRTM data, which flew in February 2000, provided scientists with digital elevation data on a limited region (between ±60 degree latitude). The mission uses C and X-band INSAR data. The digital topographic map produces 30 m × 30 m spatial sampling digital elevation with 16 m absolute vertical height accuracy. SRTM has provided good support for baseline correction, 2-pass DINSAR, and phase unwrapping over the past 10 years. However, the tests show that the accuracy is worse (60m) than

the value mentioned [20].

SRTM shows the terrain information in the year 2000. Therefore, if there is one specific hazard to be measured, the precondition of using SRTM is that no significant deformation occurred after that year.

Much better DEMs can be expected from the TerraSAR-X and TanDEM-X satellites. The two satellites act as an interferometric pair. They are scheduled provide a DEM of the complete Earth in 2014. 2.5 years after launch, the expected initial DEM product will provide an accuracy better than 30m depending on geometrical baseline. The relative height accuracy will be approximately 2m (four years after launch), and the resolution on the ground will be better than 12m [21]. The new DEM will allow the generation of new results in terrain deformation.

- 3-pass DINSAR:

  3-pass differential interferometry is based on three SAR images to derive two interferograms based on the same reference (same master image). One of the pairs with a short acquisition time interval and a large baseline before the deformation is used to estimate the topographic phase (topography pair). The other pair has to contain one scene before and another scene after the deformation (deformation pair). Therefore, two scenes before deformation and one scene after that are chosen. The shorter the time interval, the better the coherence. This method is good at monitoring deformation of

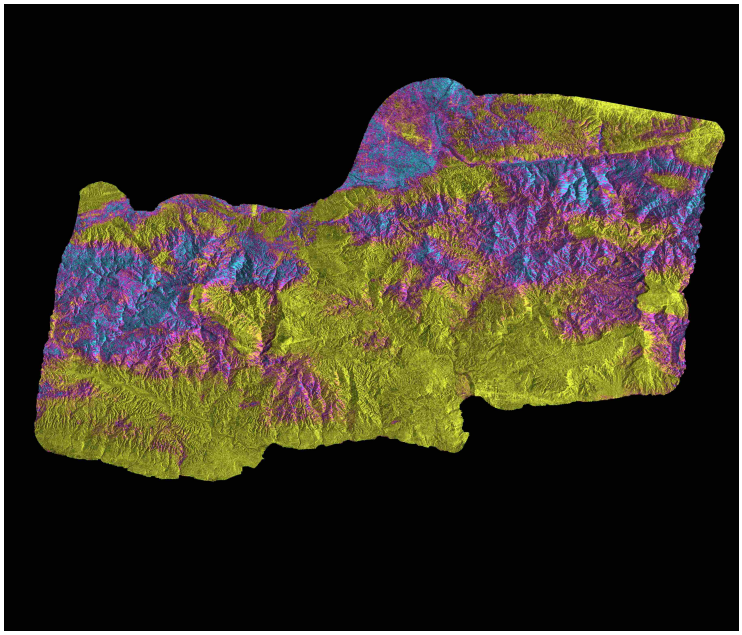
short temporal baseline. Another advantage of the method is that no DEM is required.

- 4-pass DINSAR:

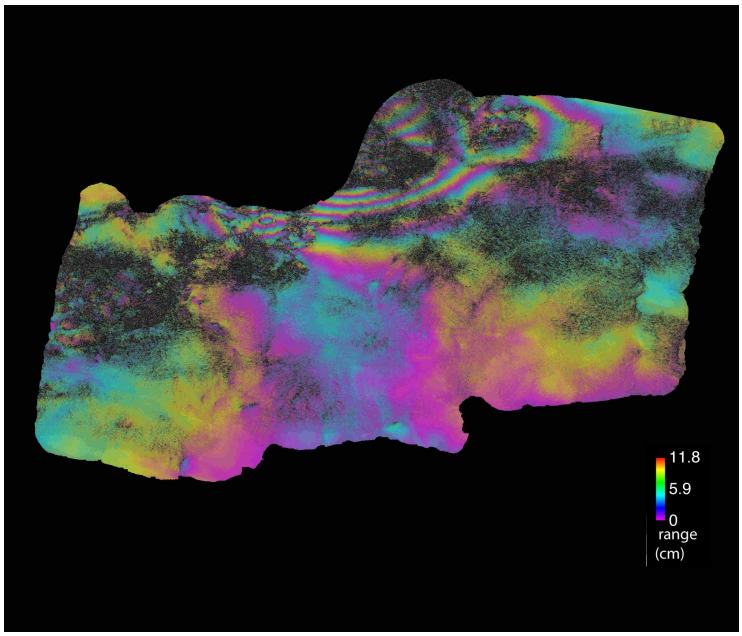
  Similar to the 3-pass DINSAR, two scenes before the deformation and two scenes after that are required. 4-pass DINSAR has the advantage of a less stringent coherence requirement since it does not need the temporal baseline to be short in the middle of the four scenes.

In 2-pass processing, Φ_{elev} is simulated based on the available baseline and look angle information. The external DEM, which is in the geo-referenced map coordinate system, needs to be transformed to the SAR geometric coordinate system. By utilizing the same orbital parameter, the simulated phase will show the same pattern as the interferometric phase that contains the topographic fringes. The simulated phase is subtracted from the interferometric phase to leave mainly Φ_{elev} and Φ_{atm}. The remaining parts are trivial, including phase unwrapping and height conversion, as discussed in Chapter 3.

Figure 4.1 is an example of a 2-pass differential interferogram of the famous Haiti earthquake on 12 January 2010. The black area of the differential interferogram stands for the area with low coherence. The coherence of the differential interferogram is the same as the original interferogram. The colorful "fringes" show contours of the ground deformation caused by the earthquake. Each contour represents 11.8 cm (half of the wavelength) of ground motion. For earthquakes,



(a)



(b)

Figure 4.1: Haiti earthquake: 20100125-2009030. (a) Coherence; (b) Differential interferogram.

the fringes are supposed to spread from the epicenter of the deformation in circular ring. The official epicenter location from GPS detection is at $18.457N$ $72.533W$ (Figure 4.2). Compared with Figure 4.1, the differential interferogram detects almost the same position of the epicenter. Therefore, the differential interferogram gives results that are equivalent to a "natural" GPS network in monitoring sliding areas.

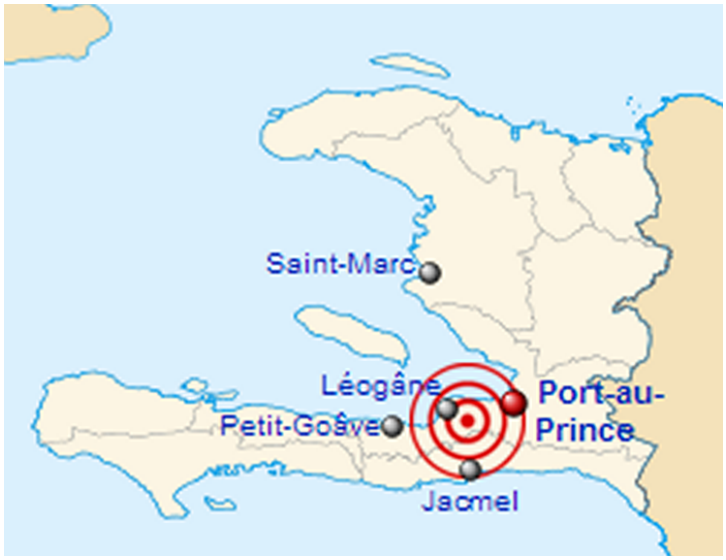


Figure 4.2: Epicenter: Haiti earthquake 12 Jan 2010. [4]

If there is no deformation over the entire scene, the differential interferogram can be used for baseline refinement. Theoretically, the calculated phase is equal to the simulated phase, so the differential interferogram has no linear phase fringe. The presence of a linear fringe means that the estimated baseline has some errors. To estimate the baseline residue, the FFT method introduced before is used to eliminate the linear fringes from the resultant differential interferogram. Finally, this baseline residue will be added to the initial estimated baseline.

This baseline refinement method works well for the deformations of relatively small area over the entire image (volcano area). However, in the differential interferogram of Haiti above, the baseline cannot be refined. This is because the deformation happened everywhere across the land, and the linear phase term does not represent only orbital inaccuracy. The refinement will also remove the deformation fringes. The master and slave images used in Haiti earthquake have provided accurate orbit information, but it does not mean that the orbit is accurate for all other paths.

Orbit inaccuracy can be considered as the most difficult part for deformation monitoring in Southeast Asia. The land-covered area is very limited, due to the presence of many small islands. The epicenter of the earthquake is in the sea area, which causes the deformation over the whole island.

4.2 Volcano monitoring of Lusi

The Lusi volcano erupted on May 29th, 2006. Drilling [22] or an earthquake [23] may have resulted in the Sidoarjo mud flow in the Porong subdistrict of East Java province. The mud covered about 4.40 km^2, and inundated four villages, homes, roads, rice fields, and factories, displacing about 24,000 people and killing 14. As the volcano is in the middle of a suburb of Sidoarjo, more than 30,000 people had to move elsewhere. The mud comes from a pocket of pressurized hot water about 2500 meters below the surface. When the water rises, it mixes with sediments to

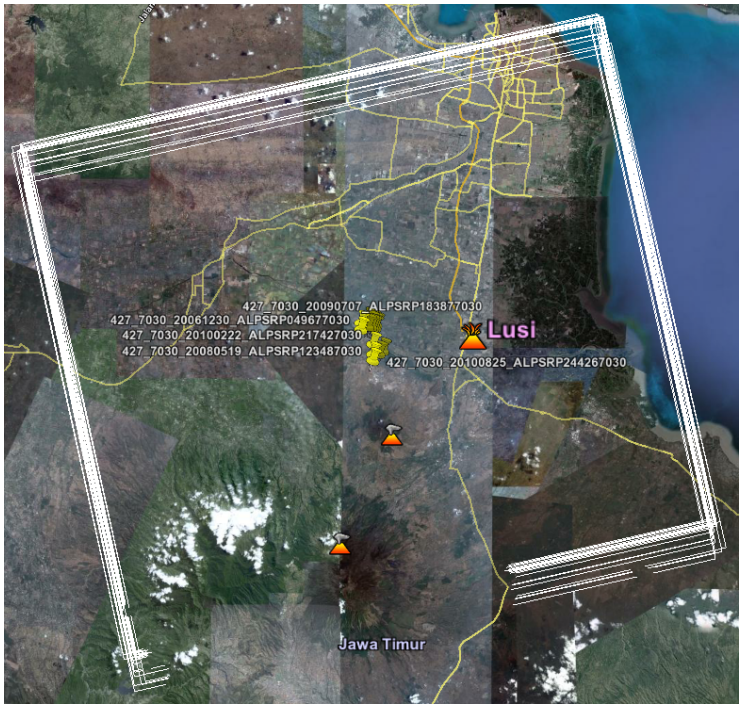
form the viscous mud. The detailed satellite images of the evolution are shown in Appendix C.

2-pass differential interferogram is applied with SRTM as reference DEM. The chosen frame is shown on Figure 4.3, with the location of interest marked. The area is very small (less than 10.0 km^2) comparing with the whole image. Therefore, baseline refinement can be applied to the differential interferogram.

Figure 4.4 shows the result of differential interferogram. The areas without deformation (also without phase change) are in GREEN. It can be observed that the phase cycle at the edge of volcano is good and coherent, but the phase at the crater is a bit messy due to the presence of smoke and vapor. From the image of the Haiti earthquake (Figure 4.1), the sea area is completely dark with no coherence; likewise, mud with water and vapor can absorb signal phase and the coherence will be poor. This analysis can be validated with the photo taken near the crater (Figure 4.5). The smoke and vapor diffusing from the crater results in a drop in coherence. The purple phase on the left of Figure 4.4 is the cloudy area. The shape of the differential phase is the same as that of the cloud [24].

The baseline and the ionosphere effect can be observed in some of the other interferograms (Figure 4.6). One observation about the baseline effect is that only some interferograms have this problem. There is no real connection between a long period of time between master and slave and the baseline effect.

The ionospheric effect is independent of the satellite and the ground motion. This is why it is difficult to take them into account in the process. In Figure 4.6,



(a)

(b)

Figure 4.3: Lusi volcano satellite frame. (a) Chosen frame over Lusi mud volcano; (b) SAR amplitude image.

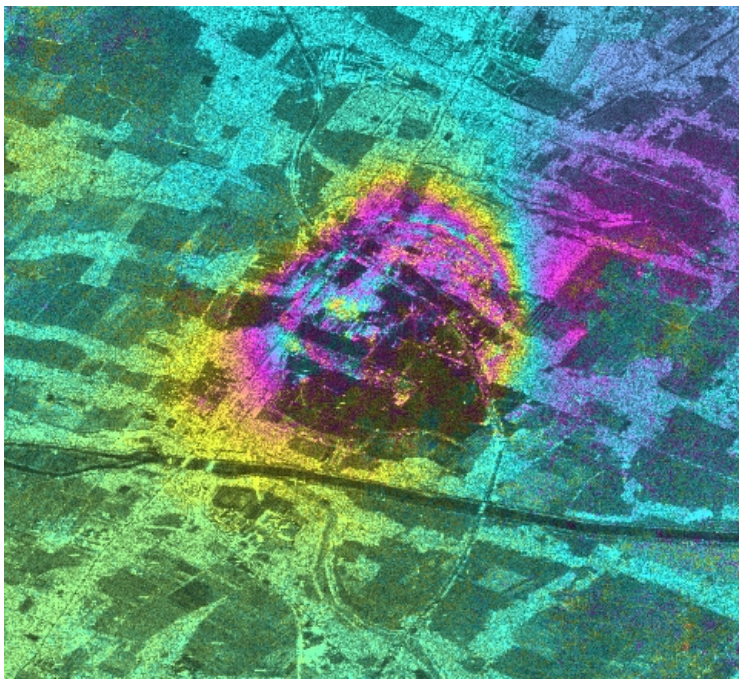


Figure 4.4: Differential interferogram after the Lusi eruption. M:20061004, S:20060519

the difference is obvious that the ionosphere phase is always nonlinear comparing with the linear phase trend of the baseline effect. The other method to distinguish ionosphere effect is time series image comparison. Assume there is a time series of SAR images which are labeled 1, 2, 3... n, $n+1$... The expression $D_{n,n+1}$ represents the interferogram generated by image n as master with image $n+1$ as slave. To know if the problem seen on one interferogram $D_{n,n+1}$ is really due to atmospheric effects in one of the image, we check the interferograms $D_{n-1,n+1}$ and $D_{n,D_{n+2}}$. If we do not see any effect on $D_{n-1,n+1}$ or $D_{n,n+2}$, it means that there was atmospheric issues on image n or image $n+1$.

Therefore, in conclusion, the DINSAR technique can easily and accurately monitor volcano eruption activities. The baseline effect can be recognized and

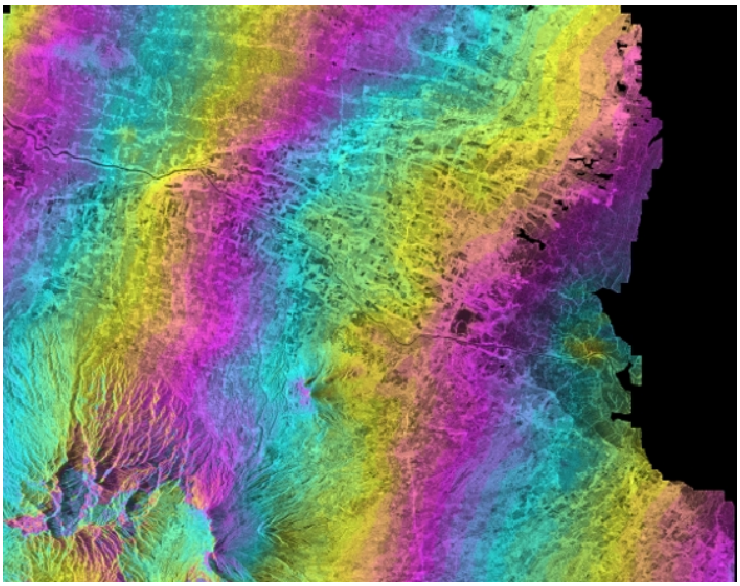
Figure 4.5: Photo taken near the crater

corrected to show the actual result. However, earthquake activities are different. The details are discussed in the next section.

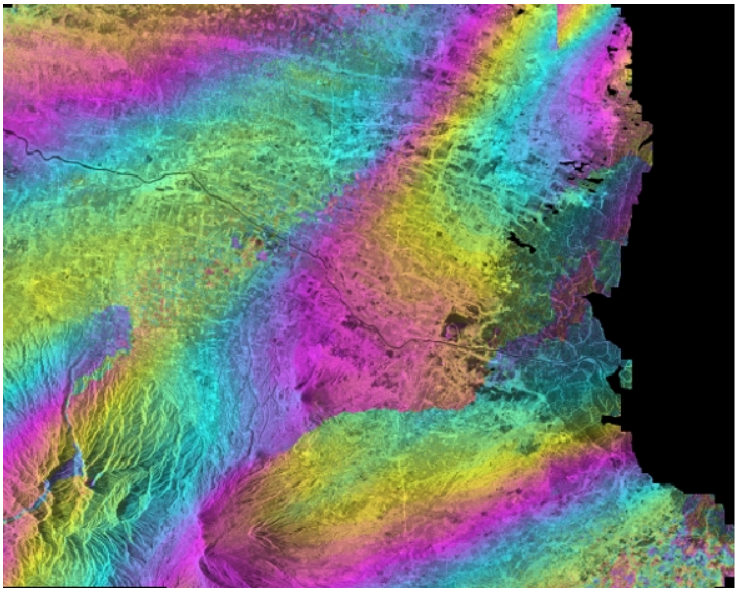
4.3 2009 earthquake of Padang, Sumatra

On September 30, 2009, a major earthquake of magnitude 7.6 occurred near the west coast of Sumatra close to the city of Padang. On October 1, a significant aftershock (magnitude 6.6) occurred 270 km away. The casualties were estimated at 1200. This earthquake came at a time when seismic activity in the region was particularly high [25].

Most areas of Sumatra is covered by dense vegetation, and the town is mountainous with steep slopes (the altitude reaches 3200m less than 50km from the sea). Furthermore, access to this region is difficult, which makes spaceborne interferometry more suitable than airborne. As there are currently no tandem missions for spaceborne interferometry, the temporal baselines are usually impor-



(a)



(b)

Figure 4.6: Baseline effect and ionosphere effect. (a) Baseline effect of interferogram 20080519 – 20080704; (b) Ionosphere effect of interferogram 20081119 – 20090104.

tant, about 30-50 days depending on the orbit cycle. With such long temporal baselines, the L-band ALOS PALSAR is more promising as the coherence usually remains sufficiently high even in heavily forested areas.

The available scenes of the interested area are shown in Figure 4.7, and the scene information is provided in Table 4.1.

The coherence between the different images is strongly dependent on the temporal baseline. As most of the areas are vegetation, the coherence is quite low even between consecutive orbits. Fig. 4.8 shows the evolution of the coherence according to the temporal baseline. These values correspond to the average coherence over a 4.8 km by 6 km area common to the two satellite path. The low coherence at three orbits (138 days) is due to the relatively large spatial baseline (about 1 km). Within this region, some isolated points have a higher coherence.

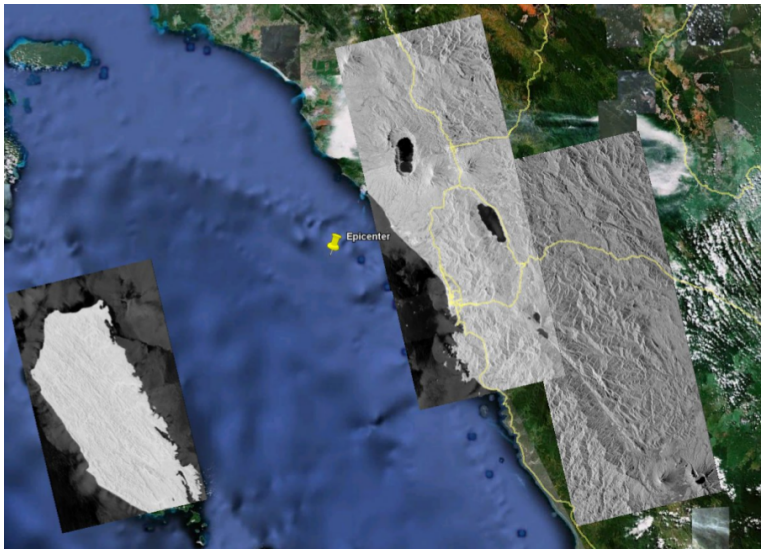


Figure 4.7: Location of the scenes used around the earthquake epicenter.

Table 4.1: Available passes

Path	Frame	Date
446	7150	2009-02-20, 2009-07-08, 2009-10-08
446	7160	2009-02-20, 2009-07-08, 2009-10-08
446	7170	2009-02-20, 2009-07-08, 2009-10-08
447	7160	2009-03-09, 2009-09-09, 2009-10-25
447	7170	2009-03-09, 2009-09-09, 2009-10-25
447	7180	2009-03-09, 2009-09-09, 2009-10-25
450	7150	2009-07-30, 2009-09-14, 2009-10-30
450	7160	2009-07-30, 2009-09-14, 2009-10-30

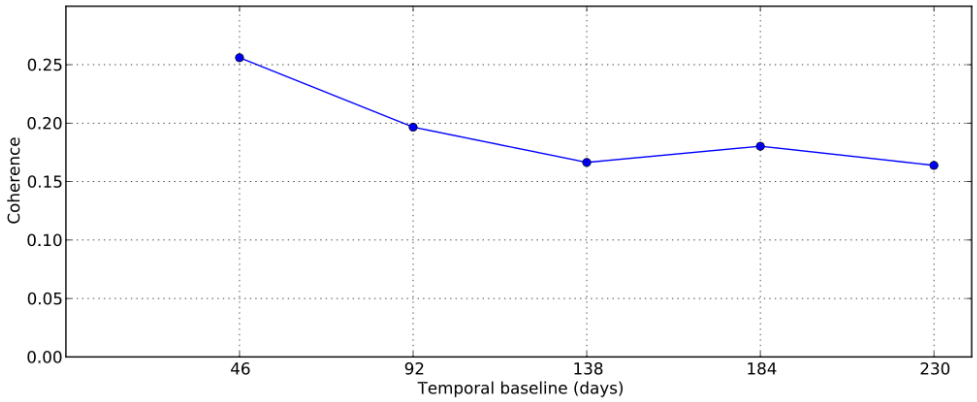


Figure 4.8: Average coherence over a 4.8 km by 6 km area for different temporal baselines.

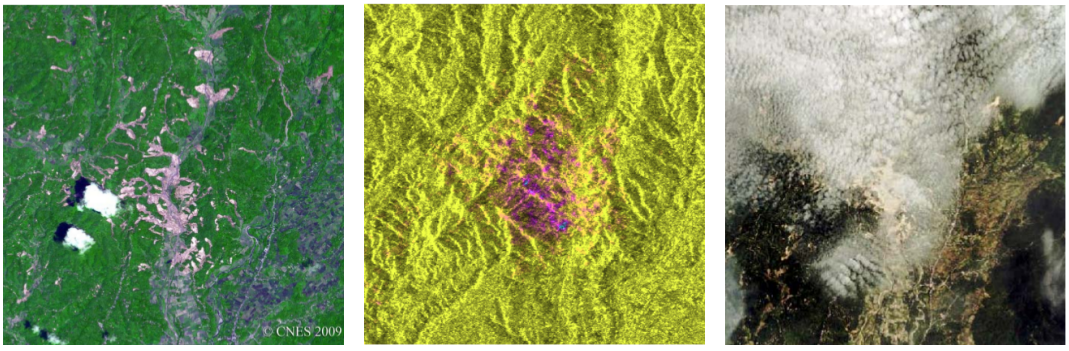
4.3.1 Landslide detection using coherence map

One of the important consequences of the earthquake is the major landslides that occurred in the area. These landslides are easy to see on optical images from SPOT5.

The advantage of SAR systems is that they are able to acquire the image in spite of bad weather conditions. However, it is difficult to identify a landslide from the SAR data (e.g. SLC) directly. For this purpose, a coherence map, which indicates the change in the scattering properties of each pixel, can be generated

as a by-product of interferometric processing (Figure 4.9).

Our study area is located north of the city of Padang, in the regency of Padang Pariaman, where the most severe landslides caused by the earthquake have occurred. The coherence is computed from the interferogram and filtered with an adaptive function based on the local interferogram fringes. Between two consecutive orbits (46 days), coherence usually remains high even for vegetated areas (in L-band). For example, many of the vegetated areas in Fig. 4.9(b) have a coherence above 0.8 (in yellow). In contrast, the part where the landslide occurred has a coherence below 0.2 (in blue).



(a) SPOT5 (b) PALSAR coherence (c) Ikonos

Figure 4.9: Images (a) and (b) show roughly the same area with a SPOT5 image where the landslides are obvious, and a color composite of the multilook PALSAR image of the same area and the coherence computed between two images before and after the earthquake. Areas of low coherence appear in blue and indicate the landslides. (c) shows the same area by Ikonos: note the cloud cover affacts the image quality.

4.3.2 The DINSAR result

The 3-pass differential interferogram is generated in this area using the passes given in Table 4.1. Acquisition over Padang does not present such an ideal configuration. The perpendicular baseline between the images taken before and after the earthquake is very small, which is less than 2 m at near range. This short baseline amplifies the errors in the interferogram due to an imperfect knowledge of the orbit.

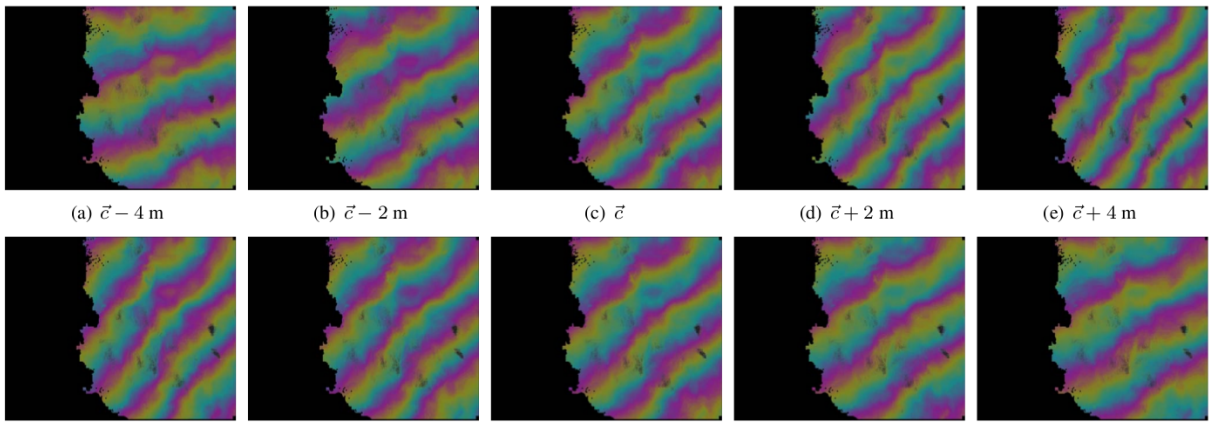
The TCN coordinate system is used to represent the baseline:

$$\hat{n} = \frac{-\vec{P}}{\mid P \mid} \qquad \hat{c} = \frac{\hat{n} \times \vec{V}}{\mid \hat{n} \times \vec{V} \mid} \qquad \hat{t} = \hat{c} \times \hat{n} \tag{4.2}$$

where $\vec{P}$ is the platform position vector with respect to the Earth center, and $\vec{V}$ is the direction of velocity vector.

Does the baseline need to be corrected? The 2 passes to generate the base interferogram before the earthquake happens can be corrected using SRTM [26], but the interferogram after the earthquake requires great attention since the deformation fringe may vanish. Figure 4.10 (c) shows the result of differential interferogram without baseline correction. The linear phase trend indicates the existance of the baseline problem. The other images in Figure 4.10 show the results after changing the $\hat{c}$ and $\hat{n}$ components of the baseline, and none of them provides a good estimation.

Figure 4.11 is the result after baseline correction. All the fringes have van-



(a) $\vec{c} - 4$ m (b) $\vec{c} - 2$ m (c) $\vec{c}$ (d) $\vec{c} + 2$ m (e) $\vec{c} + 4$ m

(f) $\vec{n} - 4$ m (g) $\vec{n} - 2$ m (h) $\vec{n}$ (i) $\vec{n} + 2$ m (j) $\vec{n} + 4$ m

Figure 4.10: Impact of the baseline accuracy on the fringe pattern. Original baseline is 141.548 m for the cross component ($\vec{c}$) and 216.412 m for the normal component ($\vec{n}$). Variations of 2% significantly impact the pattern.

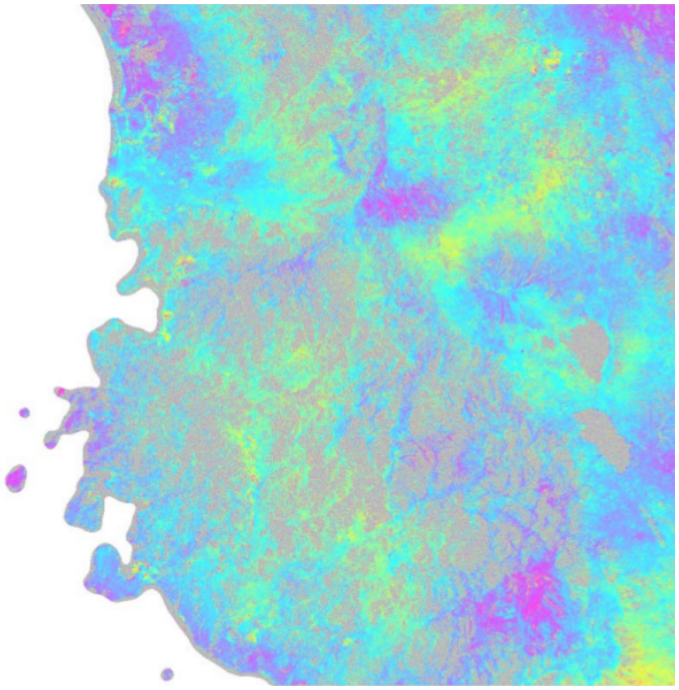


Figure 4.11: Result over the city of Padang after the earthquakes of September 30 and October 1, 2009 after baseline correction. One cycle represents a motion in the line of sight of 11.8 cm.

ished. The accuracy of this result is estimated to be worse than the ground motion measured by GPS and can be considered as inconclusive.

In conclusion, L-band interferometry is a useful tool to help in the study of earthquakes and the assessment of the resultant damage in tropical regions. The differential interferogram is particularly sensitive to baseline accuracy, even in difficult conditions. If coherence is good enough, the large number of points and the stability of the orbit of spaceborne sensors allow for a better estimation, and good results can generally be obtained.

Chapter 5

Baseline approaches: source code and software review

As discussed in Chapter 3 and 4, the contribution of baseline accuracy is significant for INSAR and DINSAR processing. The degree to which the surface topography contributes the interferometric phase depends on the interferometric baseline. The Japan Aerospace Exploration Agency (JAXA) claimed that the ALOS orbit accuracy is 6 cm [27]. However, the result of ALOS PALSAR interferometry does not give a very accurate estimate of the baseline. If baseline estimation is inaccurate, a linear phase trend ($\frac{4\pi}{\lambda} \times \Delta\phi$) will remain in the interferometric phase (Figure 4.10). The baseline error can be up to several meters, which is unacceptable for earthquake monitoring. Therefore, the problem can be due to the software or the orbit data.

In this chapter, existing software codes and algorithms are compared to find

the best approach for ALOS orbit interpretation.

Currently, there are several open source software applications available as following:

- GMTSAR [28]

  An InSAR processing system based on generic mapping tools(GMT) - open source GNU General Public License

- ROIPAC [29]

  ROIPAC is a repeat orbit interferometry package produced by Jet Propulsion Laboratory (NASA) and Caltech. ROIPAC is UNIX based, and can be freely downloaded from the Open Channel Foundation.

- WINSAR [30]

  WINSAR is a consortium of universities and research laboratories established by a group of practicing scientists and engineers to facilitate collaboration in, and advancement of, earth science research using radar remote sensing. It is the preprocessor for GMTSAR and ROIPAC.

- GAMMA

  Commercial software suite consisting of different modules covering SAR data processing, SAR interferometry, differential SAR interferometry, and interferometric point target analysis (IPTA), runs on Solaris, Linux, Mac OS X, and Windows.

Table 5.1: Comparison of software: Functions

Software	Level 1.0 parameter reader	Level 1.1 parameter reader	Interferogram generation	Baseline estimation	Flattening
WINSAR	Yes	No	No	Yes	No
GMTSAR	From WINSAR	No	Yes	From WINSAR	Yes
ROIPAC	From WINSAR	No	Yes	Yes	Yes
GAMMA	Yes	Yes	Yes	Yes	Yes

Table 5.2: Comparison of software: Usability

Software	Usability
WINSAR	Written in C, easy to install and study, good documentation and explanation.
GMTSAR	Written in C, hard to install (some error in the code), but the code is readable and understandable.
ROIPAC	Hard to install and hard to use, written in C, Fortran, and integrated with Pearl, almost no documentation, but some algorithms can be found in [31].
GAMMA	Written in C, easy to install and study, good documentation and explanation.

Table 5.1 and Table 5.2 show the comparison of the software packages. GMTSAR cannot be installed correctly and the the flattening processing is almost the same as Gamma, and WINSAR does not have interferometric processing. Therefore, the comparisons will be mainly on the algorithms of ROIPAC and GAMMA. To know which step is wrong, the main method is to correct the GAMMA source code using algorithm from ROIPAC, so the processing can be compared step by step. The comparisons will be made from the first step of data processing, since every single parameter can have some effect on the estimated baseline result, and furtherly on the flattened interferogram. The following sections will compare the software approaches in 3 aspects: the time estimation, the baseline estimation and the flattenning.

5.1 Comparison of data starting time estimation

ROIPAC uses WINSAR as the preprocessor, which includes four programs:

- *ALOS_pre_process*

  Takes the raw ALOS PALSAR data and aligns the data in the near range. In addition it produces a parameter files in the SIOSAR format containing the essential information needed to focus the data as Single Look Complex (SLC) images.

- *ALOS_baseline*

  Takes two parameter files of an interferometric pair and calculates the approximate shift parameters needed to align the two images as well as the accurate interferometric baseline at the beginning and end of the frame.

- *ALOS_merge*

  Appends two raw image files and eliminates duplicate lines. In addition it makes a new PRM file representing the new longer frame.

- *ALOS_fbd2fbs*

  Converts a raw image file in FBD mode (14 MHz) to an FBS mode (28 MHz) by Fourier transformation of each row of the image file (one echo) and padding the spectrum with zeros in the wave number domain. A new

parameter file is created to reflect the new data spacing and chirp parameters.

The RAW data processing are similar to the method introduced in Chapter 2, except that the WINSAR preprocessor added $\frac{16384-9216}{2\times prf}$ at the beginning of the clock start time, where prf is the pulse repetition frequency. This number 16384 is found from the PALSAR data format, with the description: "The number of azimuth samples, FFT processing unit 2 in a scene with a length of 9216 with an output of 16384 units of the length of treatment calculated." Therefore, this time interval is added after the Azimuth compression, the first line of the first segment would be (16384 - 9216)/2 lines after the first line of the FFT window. By using Level 1.1 SLC data, this problem does not need to be considered. GAMMA and WINSAR have different scene starting times, and also different starting points on the image. By checking the specific time at the same line, both of them are correct.

5.2 Comparison of the baseline estimation

Table 5.3 shows the workflow of ROIPAC, which are integrated by a Perl script.

Consider a target imaged by two antennae:

$$\vec{T} = \vec{P_1} + \rho_1 \vec{l_1} = \vec{P_2} + \rho_2 \vec{l_2} \tag{5.1}$$

Table 5.3: The workflow of ROIPAC data processing

Processing Step	Code Element
Process Control	$process.pl$
SAR raw data conditioning	$make_raw.pl$
SAR Image formation	roi
SAR Image registration	$ampcor$
Interferogram formation	$resamp\_roi$
Baseline determination	$haseest$
Interferogram flattening	$cecpxcc, rilooks$
Correlation determination	$Makecc, icu$
Interferogram filtering	icu
Phase unwrapping and absolute phase determination	$Icu, baseline$
Deformation determination	$diffnsim$
Topography determination from unwrapped phase	$inverse3d$

where $\vec{T}$ is the target position vector, $\vec{P_i}$ is the antenna i center position vector, ρ_i is the range of antenna i to the target, $\vec{l_i}$ is the unit look vector from antenna i to the target, and the subscripts $i = 1, 2$ represent the master and slave images, respectively.

From Eq. (5.1), the interferometric baseline $\vec{B}$ can be written as:

$$\vec{B} = \vec{P_2} - \vec{P_1} = \rho_1\vec{l_1} - \rho_2\vec{l_2} \tag{5.2}$$

To find the correct baseline, ROIPAC and GAMMA use different approaches. With the starting time recorded with the master image, the baseline estimation will be carried out at the top, middle and bottom lines of the image. At each point, interpolation will be carried out using the state vectors. GAMMA will interpolate platform position $(\vec{P_x}, \vec{P_y}, \vec{P_z})$ and velocity $(\vec{V_x}, \vec{V_y}, \vec{V_z})$ separately with an eighth order polynomial. ROIPAC uses only platform position data, and velocity (gradients of position) will be automatically generated by Hermite inter-

polation. To find the corresponding position on the slave path, GAMMA finds out the position path of the slave image, calculate the displacement vector from a point on the slave path to the master platform position, convert it to the TCN coordinate (Eq. (4.2)), and then find the point with zero value of the dot product (the displacement vector with the $\hat{t}$ component of the velocity), iteratively. On the other hand, ROIPAC locates a range offset ($\pm$1000 pixels) on the slave image, and calculates the absolute displacement to find the smallest value. Each time an offset of $\frac{1}{prf}$ seconds is added before interpolation.

The resultant baseline is different by less than 5cm in the $\vec{C}$ and $\vec{N}$ directions, but by more than 3m in the $\vec{T}$ direction. However, the $\vec{T}$ component does not have much effect on the flattening, so very little difference can be expected in the flattening result.

5.3 Comparison of flattening

The representation of the look vector has different expressions in ROIPAC and GAMMA. With reference to Figure 5.1, we have:

ROIPAC:

$$\vec{l_1} = \begin{bmatrix} \sin\theta_1 \sin\beta \\ \sin\theta_1 \cos\beta \\ -\cos\theta_1 \end{bmatrix} \tag{5.3}$$

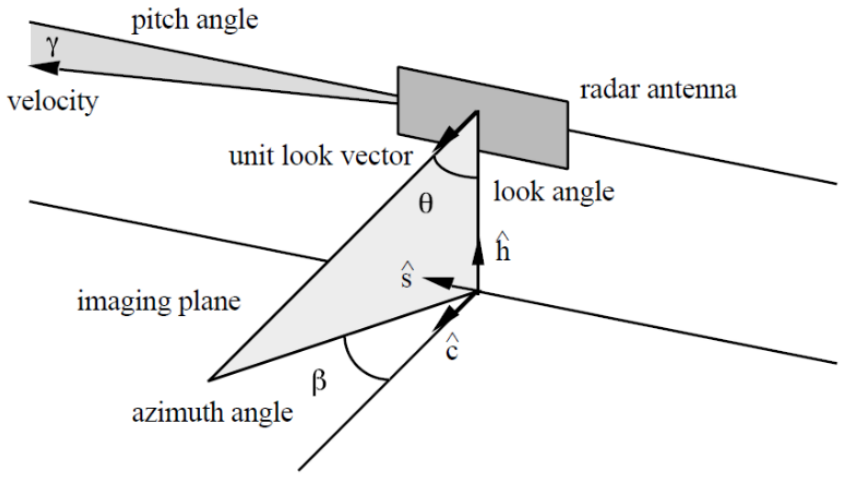


Figure 5.1: Look angle model in ROIPAC

$$\vec{l_2} = \begin{bmatrix} \cos\theta_2 \sin\gamma \cos\beta + \sin\theta_2 \sin\beta \\ -\cos\theta_2 \sin\gamma \sin\beta + \sin\theta_2 \cos\beta \\ -\cos\theta_2 \cos\gamma \end{bmatrix} \tag{5.4}$$

GAMMA:

$$\theta_1 = \arccos(\vec{lv}_{tcn}.n) \tag{5.5}$$

Usually, The squint angle (azimuth) angle β (Figure 5.1), which represents the offset between transmission and the normal of the plane of antenna, is much smaller than 1°, and can be neglected. In Eq. (5.3) and Eq. (5.4), the pitch angle γ and squint angle β are taken into consideration. Both the two angles are very small. Therefore, GAMMA just neglects them and simply uses the arccosine of the $\vec{n}$ component of the look vector $(acos(\vec{lv}_{tcn}.n))$ to represent the look angle, where $\vec{lv}_{tcn}$ is the TCN vector derived from Eq. (4.2).

To flatten the interferogram, the phase from the Earth's surface is calculated and subtracted on every point of each interferogram:

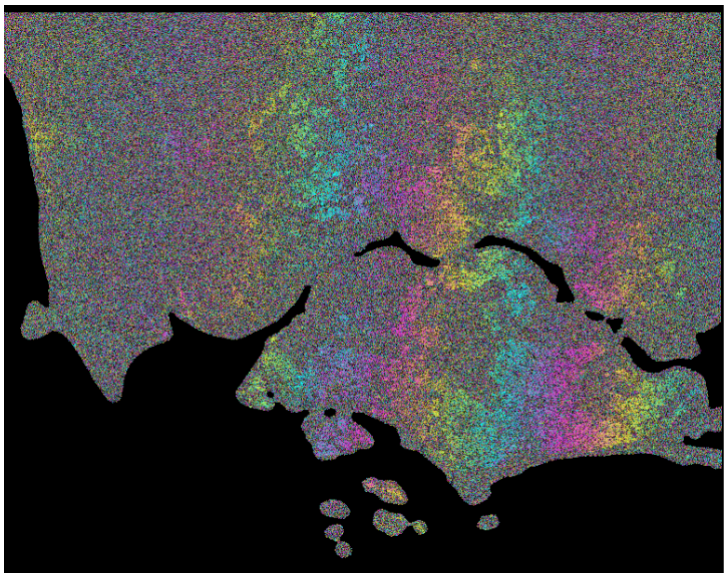
$$\begin{aligned}\phi &= \frac{2\pi p}{\lambda}(\rho_2 - \rho_1) \\ &= \frac{2\pi p}{\lambda}(|\vec{l_2}| - |\vec{l_1}|) \\ &= \frac{2\pi p}{\lambda}\rho_1((1 - \frac{2\hat{\vec{l_1}} \cdot \vec{B}}{\rho_1} + (\frac{B}{\rho_1})^2)^{1/2} - 1)\end{aligned} \tag{5.6}$$

ROIPAC makes use of this expression without any approximation (detailed processing steps are shown in Appendix D), but GAMMA uses a simplified method:

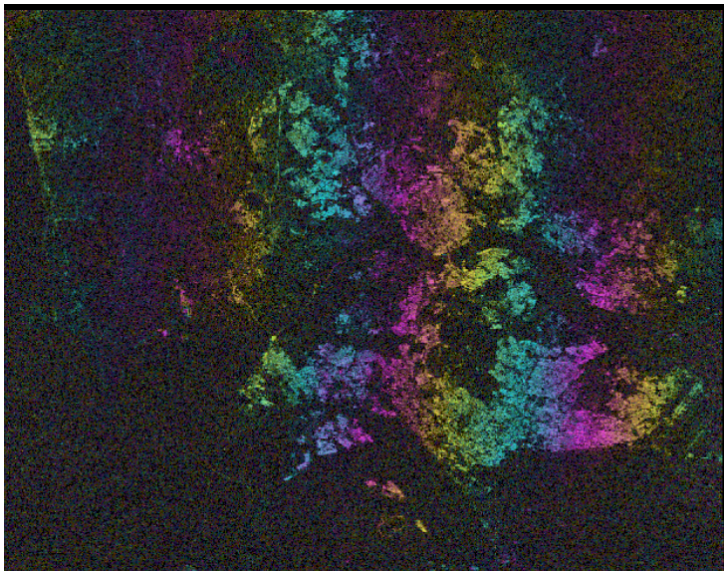
$$\phi \approx -\frac{2\pi p}{\lambda}\hat{\vec{l_1}} \cdot \vec{B} = -\frac{2\pi p}{\lambda}B_{para} \tag{5.7}$$

The result after flattening is compared by using a image with a baseline problem over Singapore (Figure 5.2). They look similar, and the baseline problem has not been solved.

The baseline estimation and flattening method of GAMMA and ROIPAC are compared. We tried to implement the methods in GAMMA, and the result shows that the estimated baselines are almost the same, with none of their method on flattening giving good results. ROIPAC is very accurate without any approximation. GAMMA uses a lot of approximations in baseline estimation and flattening, but the results are almost the same. Therefore, the problem is probably not due to software, but the inaccurate orbit. The accuracy report on [27] may have some problems. There are also two other possibilities:



(a) Result using GAMMA



(b) Result using ROIPAC

Figure 5.2: Comparison of the interferogram result by using GAMMA and ROIPAC algorithms. The master and slave images cover the Singapore area. (M:20090928, S:20090628)

- There are problems with the other parts of the processing steps, like registration, interferogram generation, line and column shif, but not on baseline and flattening.

- ROIPAC is automatic with the precondition of the existence of the DEM file, so that it might hide the baseline correction part from the users.

To solve this problem, a new method is proposed in the next chapter.

Chapter 6

Iterative calibration of relative platform position: A new method for baseline correction

By using PALSAR data and precise orbit information, a new approach for baseline correction is proposed based on a model of platform position from several acquisitions. Some of the results have been published in IGARSS 2010, Hawaii [8].

Baseline precision contributes significantly to the accuracy of SAR interferometry processing. The precise estimation on baseline is required for most applications. Usually, the initial estimation is using orbital information, and extracted from the platform position vectors of a pair of SAR passes. Therefore, based on spaceborne properties, all the platform positions can be built under a coor-

dinate system of several acquisitions at corresponding points. The error of the perpendicular baseline can be effectively reduced by using Ground Control Points (GCPs) [32] or a reference low resolution DEM [33].

Three aspects of baseline estimation have to be considered before. First, the source of the baseline error is the inaccurate estimation of platform position, which can happen on any of the interferometric pairs. A very common phenomenon in 2-pass differential interferogram is that a linear phase trend always appears on the interferogram with the same pass (Fig. 6.4) because of inaccurate platform position estimation of the single pass, even if all the other passes are accurate. Second, a majority of the methods for baseline calibration use least squares fitting based on the known height information, with the assumption of no significant ground deformation. Lastly, the correction is only done on the relative distance between the two platforms, but not the individual position.

Therefore, the concept of the baseline can be extended. In a situation of multiple available SAR images, more information on platform position is interpreted from the data. Without a global constraint over every pair of images, the geometry of platform positions will not be representative of a realistic situation (see Fig. 6.1).

In Fig. 6.1, relative positions are reconstructed using corrected baselines. If three positions are fixed (red solid arrow), then the other three baselines will not end at a single fourth point. This problem will become very complicated in 3D when more passes are used.

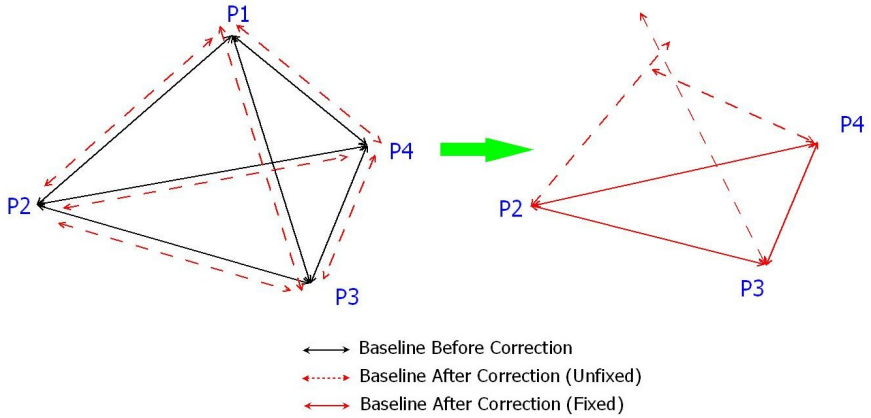


Figure 6.1: 2D illustration of the problem between 4 passes. $P1$, $P2$, $P3$ and $P4$ represent the relative platform positions of passes. 6 baselines (4 sides and 2 diagonals) are displayed (black arrow). After the correction of baselines independently without constraint, the possible inaccurate reference DEM (or GCPs) and presence of APS affect the corrected baselines (red dashed arrow).

In this chapter, an iterative optimization method is provided for baseline calibration, under the constraint from relative platform positions over several acquisitions. A reference DEM is needed (DInSAR) to calculate platform position displacement. The linear phase trend of 2-pass differential interferogram decreases during each iteration. The unique result from this method with respect to traditional techniques is the detection and quantitative position calibration of any pass with inaccurate orbit information (offset error up to 10 m) along the direction of the perpendicular baseline. The advantage of this method is that the baseline calibration is global.

6.1 Repeat-pass interferometry

The repeat-pass interferometry is the precondition to apply this method. The Repeat-Pass (Multi-Temporal) application of SAR has been of great interest over recent years. The phase information of several image data files with different temporal and spacial baselines are superimposed. These SAR scenes have the same path number and frame number. Table 6.1 shows an example of repeat passes. From several recent publications, the research topics could include:

- High-accuracy digital elevation model (DEM) construction by interferogram stacking [32].
- Separation of phase contributions of atmospheric effects (clouds, ionosphere) [34].
- Terrain deformation rate estimation by using persistent scatterer technique (PSINSAR) [35] [36]. An accuracy of a few mm/year can be achieved [37]. Further applications are proposed on hazard prediction from sub-surface strain estimation.

6.2 Algorithm

Modeling of the platform position requires a coherent coordinate system. Consider $K + 1$ SAR images of the same area with no topographic change and DEM is also available. Corresponding points can be found by precise registration of

images using cross-correlation. The ideal system can be validated by TCN coordinates from [38]. Their unit vectors ($\hat{t}$, $\hat{c}$, and $\hat{n}$) are defined by Eq. 4.2. We assume that all the platforms can have the same direction of $\vec{V}$. The difference of velocity vector can be observed from the baseline changing rate; it is always very small or can be easily corrected by a rotation matrix depending on changing rate vector.

The error of TCN axis angle obtained by taking from image i to image j as reference (under the above assumption) is:

$$\Delta\theta = \arctan \frac{\sqrt{\mid \vec{B}_{ij} \cdot \hat{c} \mid^2 + \mid \vec{B}_{ij} \cdot \hat{t} \mid^2}}{A_i + R} \tag{6.1}$$

where $\vec{B}_{ij}$ is the baseline vector between master image i and slave image j, A_i is the platform altitude of image i (691.65 km for ALOS) and R is the radius of the earth (6378.1 km). Usually the baseline component along $\hat{t}$ is small and its error contribution can be neglected. Therefore, for a baseline of 1 km along $\hat{c}$, the axis error is 0.0081° and the baseline error is $\vec{B}_{ij} \cdot \hat{c} \times \tan \Delta\theta \simeq 14$ cm for this system. In the following, the same TCN coordinate system will be considered at the corresponding point for all passes.

The starting points of the iteration come from the initial orbit-estimated baseline. The combination of the $K+1$ image is generated (totally $K(K+1)/2$ interferograms). Under the previous assumptions, the following results can be implied

(in TCN coordinates):

$$\vec{B}_{ji} = -\vec{B}_{ij} \qquad \dot{\vec{B}}_{ji} = -\dot{\vec{B}}_{ij} \tag{6.2}$$

The iteration is processed with both baseline vector $\vec{B}_{ij}$ and baseline azimuth changing rate $\dot{\vec{B}}_{ij}$.

The iteration of $\vec{B}_{ij}$ is completed as follows:

1. Step 1:

   Taking image i ($i = 1$ at beginning of each iteration) as the master image, generate 2-pass differential interferograms (reference DEM available) by taking the other K images as slave images. The baseline error can be calculated by using two methods. One method is to eliminate the frequency centroid of Fast Fourier Transform (FFT) over a specific area. The other method is to use GCPs and unwrapped phase, but it may introduce unwrapping error [39]. Average the result and estimate the standard deviation: $\Delta\vec{P}_i^{(n)} = \frac{1}{K} \times \sum_{j \neq i} \Delta\vec{B}_{ij}$, where $\Delta\vec{P}_i^{(n)}$ represents the displacement of platform position respect to image i, and n is the current iteration number.

2. Step 2:

   Update all the baseline vectors using the platform displacement.

$$\vec{B}_{ij} = \vec{B}_{ij} + \Delta\vec{P}_i^{(n)} \tag{6.3}$$

A weight coefficient $\frac{1}{n}$ can be added before $\Delta\vec{P}_i^{(n)}$ to slow down the convergence and make sure it is not trapped by local minimum. Results will be compared later using test data.

3. Step 3:

   Update the reversed baseline $\vec{B}_{ji}$ by using Eq. (6.2), take $i = i + 1$ and go back to Step 1, until all of the images have been taken once as master image (until $i = K + 1$).

4. Step 4:

   Complete iteration n. Calculate the total displacement of all platform (absolute value) at iteration n: $\Delta\vec{P}^{(n)} = \sum_{i=1}^{K+1} \mid \Delta\vec{P}_i^{(n)} \mid$. Take $n = n + 1$ and go back to Step 1 for another iteration.

   For further iterations, $n > 1$, a simpler method can be implied by direct baseline error estimation by subtracting the displacement from the previous baseline error, without calculating the differential interferogram. Then the problem is simplified to find the optimized multiple vertex locations with known side and diagonals similar to the problem in Fig. 6.1.

The algorithm for $\dot{\vec{B}}_{ij}$ is the same as $\vec{B}_{ij}$ by simply replacing $\vec{B}_{ij}$ with $\dot{\vec{B}}_{ij}$.

Traditionally, the linear phase trends are removed by the baseline correction method. However, some of them may not result from inaccurate orbits. Atmospheric phase screen can also obviously affect the interferogram over a specific

area of the image [35]. Actually, this algorithm provides an optimized relative position to minimize the global linear phase trend. The main purpose is to average all the incoherence over the interferograms to extract reliable information of platform position, and improve overall quality.

6.3 Validation using data over Singapore

The area is centered on the island of Singapore in South East Asia. The area is rather flat except for one mountain to the northwest of the image. Most of Singapore is urban while the part of Malaysia (north of the image) is mostly palm tree plantations. The data are 8 passes of PALSAR over the same area in interferometric conditions between December 2006 and September 2009 (Table 6.1). Only HH polarization is used for processing here. SRTM is used as the reference DEM, and GAMMA is used for interferogram processing. FFT is the method for baseline calibration (GCP method also gives a similar result but takes a long time for phase unwrapping). The Python script for automating the iteration is shown in Appendix G.

10 iterations are processed on the passes. The displacements of each platform position are clearly shown in Fig. 6.2(a). In the figure, only cross-track ($\hat{c}$) and normal ($\hat{n}$) coordinates are illustrated ($\hat{t}$ component is usually small and not clearly observed in 3D, even can be set to 0). The position of 20081226 before iterations is taken as the reference origin of the coordinates (we are only interested

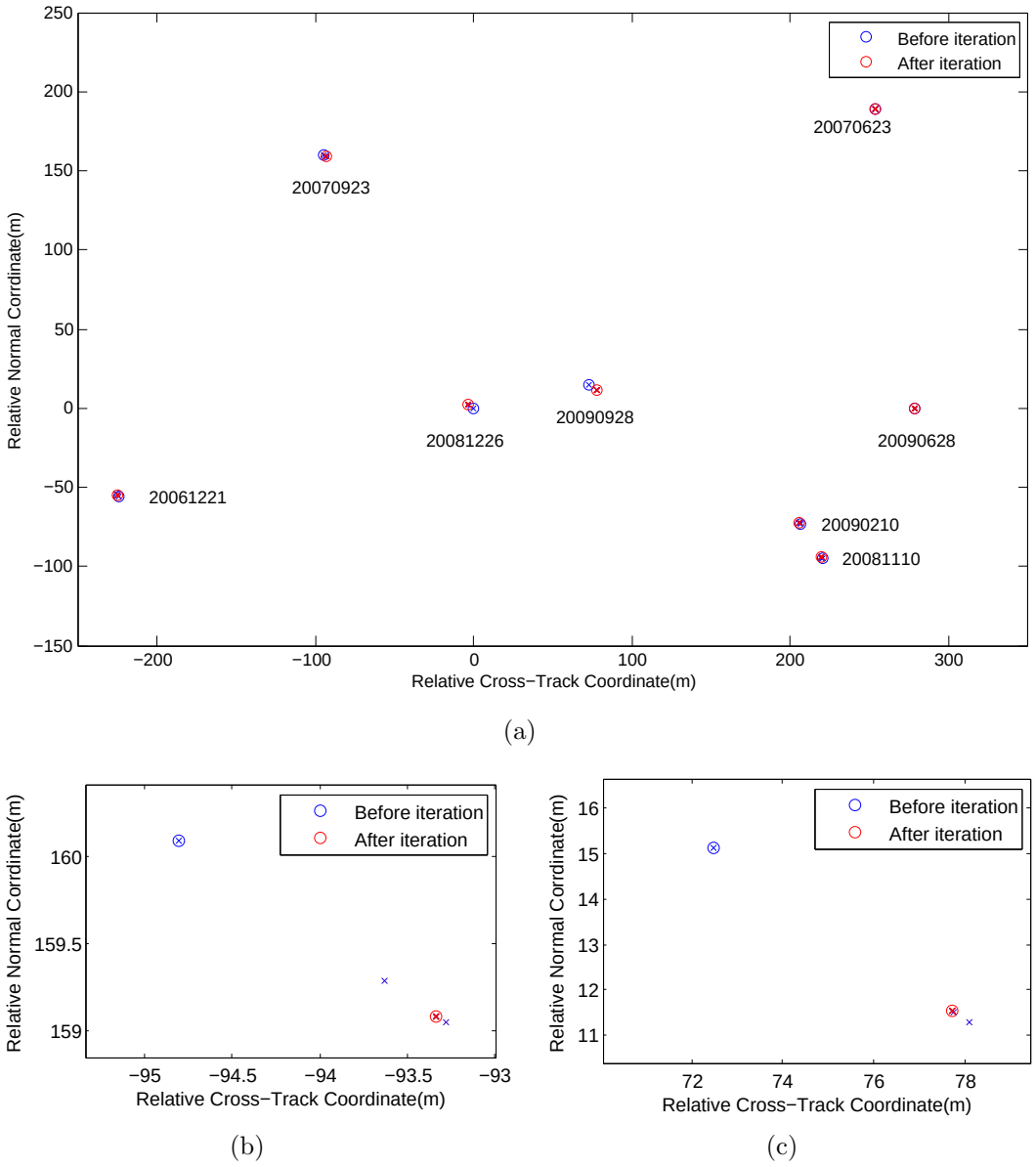
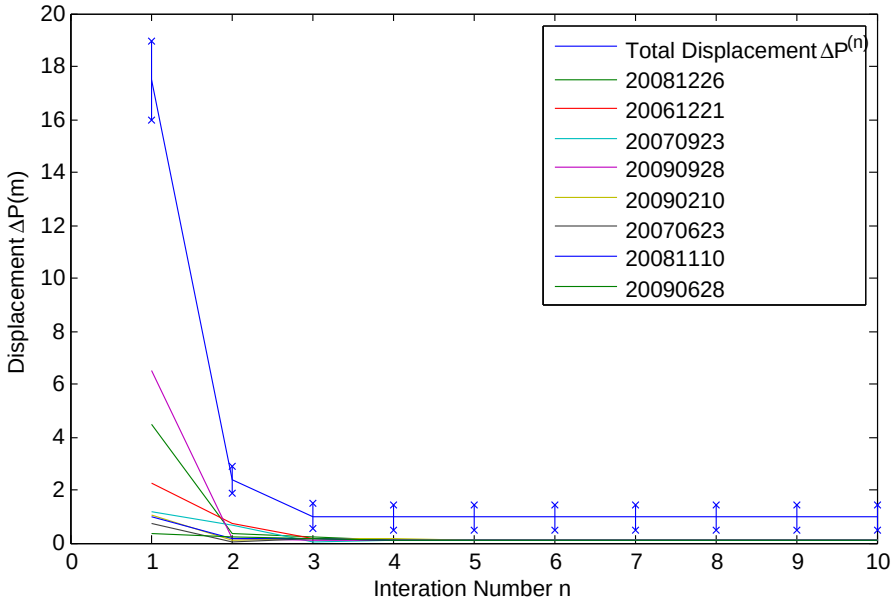
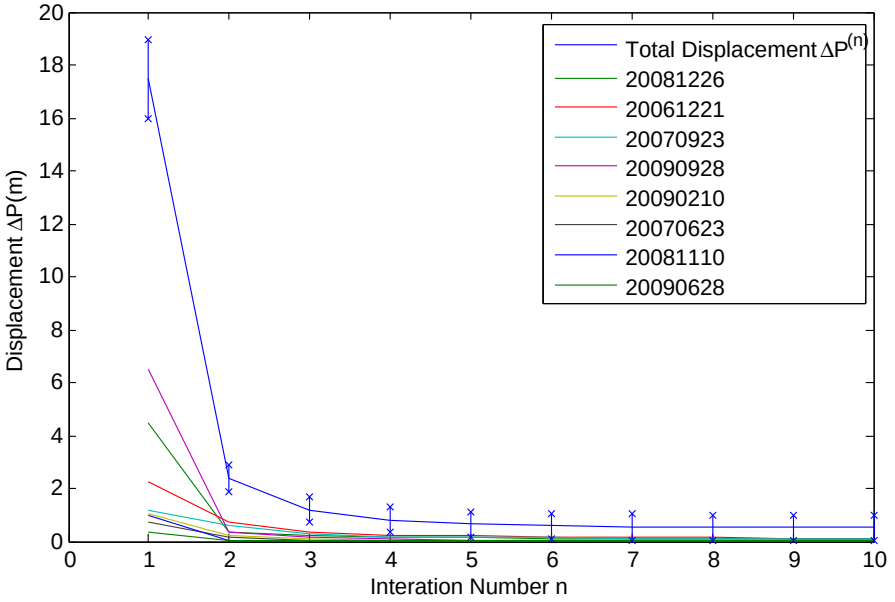


Figure 6.2: Relative position iteration of Singapore passes and zoom-in passes (20070923 and 20090928). Blue and red ∘ represent the position before and after all iterations respectively. × represents the position of each iteration. (a) Global relative position iteration; (b) Iteration for 20070923; (c) Iteration for 20090928.



(a) Iteration without weight coefficient



(b) Iteration with weight coefficient $\frac{1}{n}$ multiplied on displacement

Figure 6.3: Plot of the displacement for each pass $\Delta\vec{P}_i^{(n)}$ and the total displacement $\Delta\vec{P}^{(n)}$ during the n^{th} iteration. The total standard deviation is indicated together with $\Delta\vec{P}^{(n)}$

Table 6.1: Data sets over Singapore

Path	Frame	Date
486	0010	20061221
486	0010	20070623
486	0010	20070923
486	0010	20081110
486	0010	20081226
486	0010	20090210
486	0010	20090628
486	0010	20090928

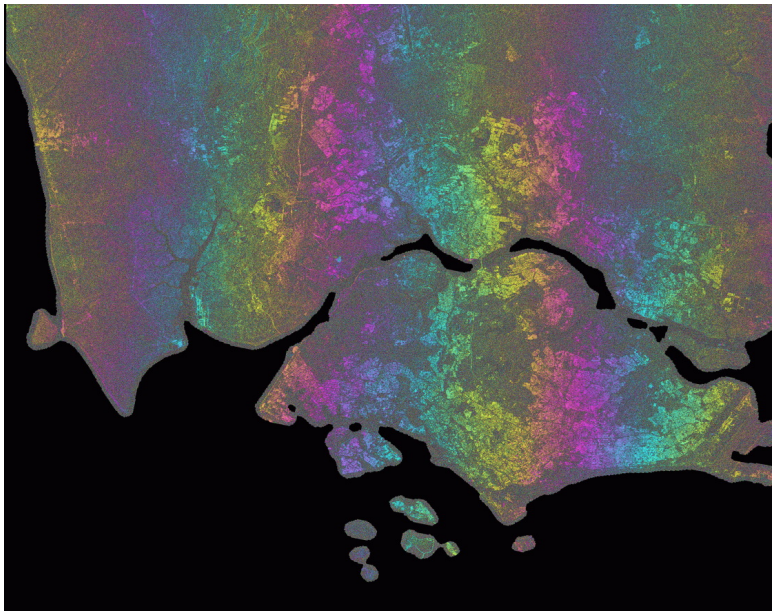
in relative position). The 8 positions are translated back every time after Step 4 by minimizing global displacement. The maximum $\hat{c}$ component of baseline is about 300 m. From Eq. (6.1), the estimated system error is 4 cm. Two obvious calibrated passes can be detected (20081226 and 20090928). Details are shown for selected small calibrated passes (20070923)(Fig. 6.2(b)) and large calibrated passes (20090928)(Fig. 6.2(c)). The displacements are 1.78 m and 6.35 m respectively. The displacement follows almost a straight line along the direction of the perpendicular baseline in the TCN coordinate system.

Fig. 6.3(a) shows the plots of displacement during the iterations. The total displacement $\Delta\vec{P}^{(n)}$ is an indication of the amount of the linear phase trend that is left during the n^{th} iteration. The convergence does not go to zero (1m in Fig. 6.3(a)), which verifies that some of the linear phase trend cannot be removed under the given constraints (SRTM error and APS). The small converged value of $\Delta\vec{P}^{(n)}$ indicate an accurate result of relative platform position calibration. Furthermore, the standard deviation values support the argument in Fig. 6.1. It

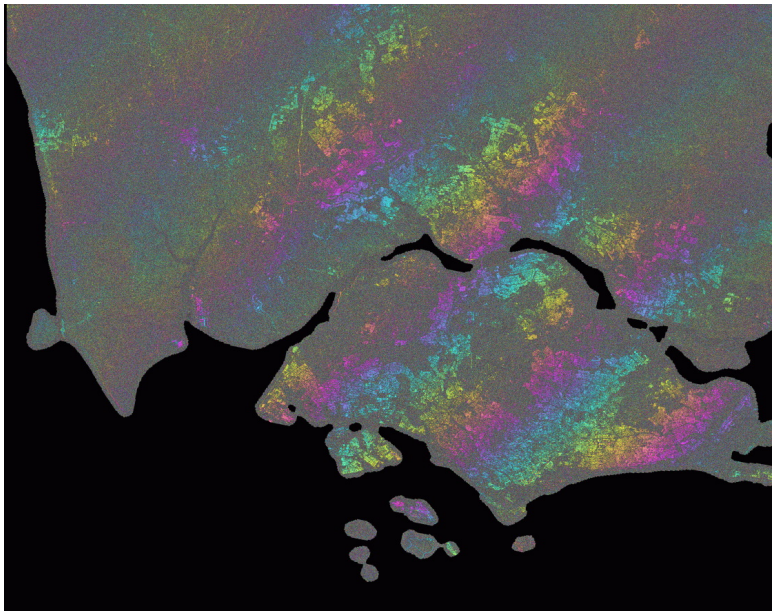
shows that the displacements provided by the other K passes are different with about 0.5 m standard deviation.

For a comparison, Fig. 6.3(b) shows the result with weight coefficient $\frac{1}{n}$ multiplied before $\Delta\vec{P}_i^{(n)}$ in Eq. (6.3). Obviously the convergence is slower, but results in a smaller value (about 0.5m), which is better. We can conclude the convergence speed can neither be too slow nor too fast. However, the speed does not depend on the coefficient.

We observe that 5 iterations are enough for the platform positions to converge. Therefore, processing time can be further reduced, which depends on the number of passes used. Fig. 6.4 and Fig. 6.5 show the 2-pass differential interferogram of the most inaccurate pass 20090928 with some other passes before and after iteration. After processing, the new estimated baseline improves the quality of the interferogram. The PALSAR passes which give inaccurate platform position are successfully detected and calibrated using this algorithm.

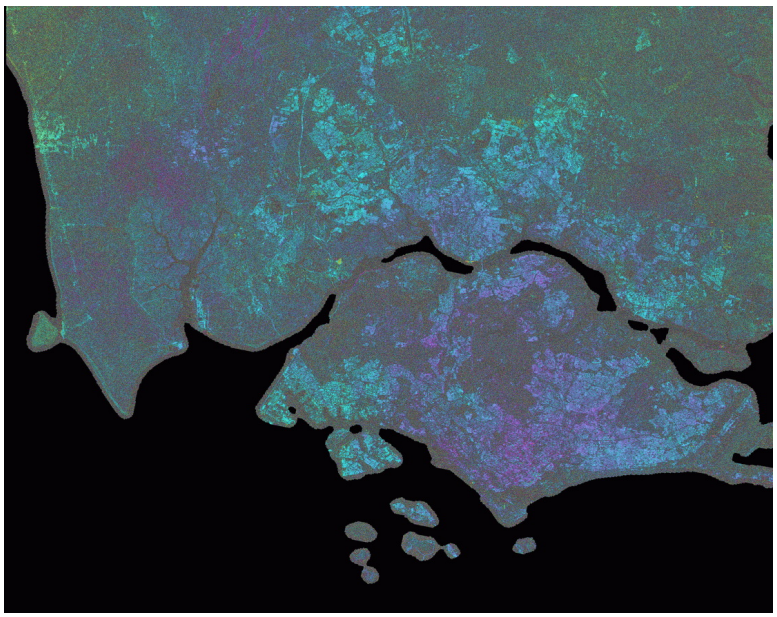


(a) M: 20090928 S: 20090628

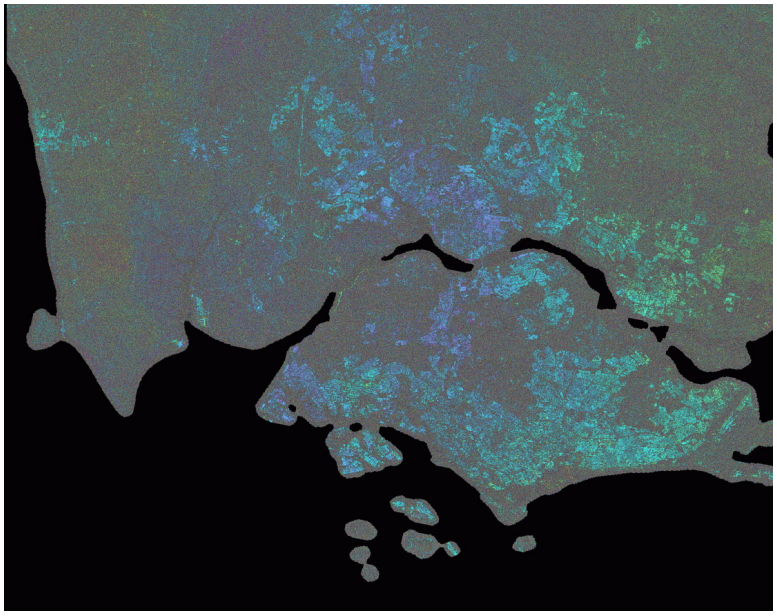


(b) M: 20090928 S: 20081110

Figure 6.4: 2-pass DInSAR before baseline correction.



(a) M: 20090928 S: 20090628



(b) M: 20090928 S: 20081110

Figure 6.5: 2-pass DInSAR after baseline correction.

Chapter 7

Conclusion

The advantage of spaceborne systems is the fast and efficient observation over large areas. This dissertation has systematically described the application of the spaceborne SAR interferometry technique over hazard-active areas of Southeast Asia. A series of geoscience and image processing techniques are used, and we have achieved the following goals:

1. The systematic SAR processing platform is built in NUS CRISP (the first place that can handle INSAR in Singapore). The Python script can handle the interferometric processing automatically once new data comes.

2. We have presented an iterative optimization of baseline with the constraint of relative platform position. Different from the past idea that interferogram is calibrated from satellite position, this method constructs a reverse model that satellite position can be relatively calibrated from multiple in-

terferograms. We believe there will be many applications that will benefit from this method, such as satellite orbit calibration and DEM generation by interferogram stacking, and baseline correction for deformation monitoring.

3. The platform we have developed facilitates further research on PSINSAR. The subsidence of Singapore reclaimed land will be investigated on the deformation rate at an accuracy of $mm/year$. Furthermore, this platform inspired us to develop a software for SAR processing. The probable release will be based on the Orfeo Tool Box (OTB) [40].

Appendix A

Processing raw(Level 1.0) to SLC(Level 1.1)

Although the two directions, i.e. range and azimuth, are not independent from each other, processing a SAR image consists in separate actions on the range spectrum and on the azimuth spectrum. Figure 1 illustrates with a simplified block diagram the core of SAR processing using a Range/Doppler algorithm. The processing steps include data analysis, parameter estimation, calculation of the derivation of range and azimuth spectrum, and their compression respectively.

A.1 Level 1.0 data format

SAR raw data is delivered in the Committee on Earth Observation Satellites (CEOS) format. The CEOS format SAR raw data nominally consists of a volume directory file, SAR leader file, raw data file, and a null volume file. The volume

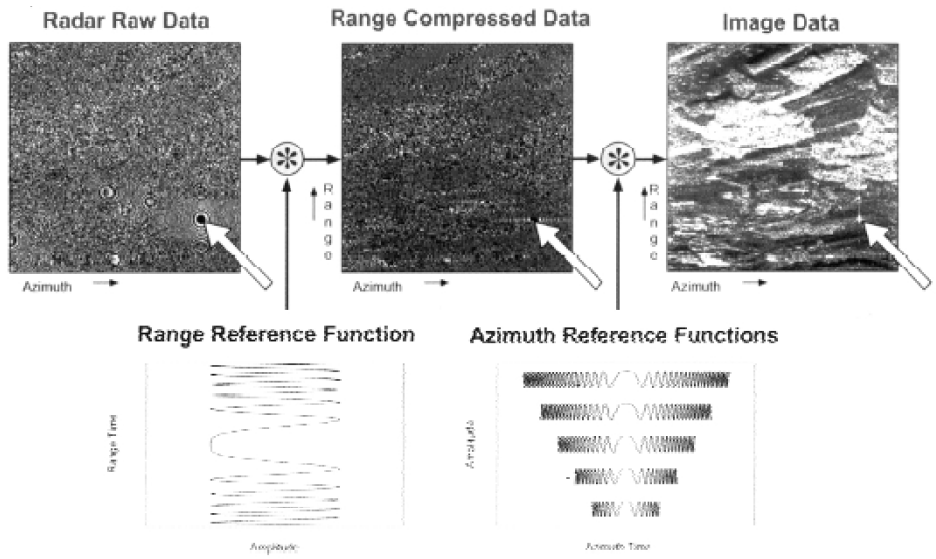


Figure 1: Simplified scheme of a Range/Doppler algorithm ([5]).

directory file describes the arrangement of the data on the storage media. The SAR leader file provides pertinent information about the specific SAR data set: raw data file size, spacecraft height and velocity, scene center latitude, longitude and time of acquisition, etc. The raw data file include a header record which is in equal size as one data line containing data size information, plus the SAR raw data nominally stored one line per record. Each record consists of a prefix, raw data, and a suffix.

A.2 Pre-processing: missing line and state vectors

As part of the preprocessing of the data at the archival facility, bad data lines may be identified and removed. Missing lines in the raw data are a problem

for interferometry applications where two images must be registered to the sub-pixel level. To check for missing lines, the line counter may be extracted from the prefix information of each raw data record. Missing lines are identified by non-consecutive line counter values. To fix the raw data for a missing line, the previous data line simply is duplicated as a placeholder for the missing line.

Orbit data is important for baseline estimation, especially for georeference registration and baseline estimation in interferometry. The original orbit data in leader file has a record of 28 sets of vectors within every 60 seconds. Each set has a position coordinate and a velocity coordinate in a reference system (X, Y, Z). The time between the start and the end of a SAR image is about 8-10 seconds. Therefore 11 continuous sets of vectors around the center time of SAR image will be selected as state vectors. If the position and velocity of a specific time is needed, a polynomial interpolation will show the result.

A.3 Range spectrum estimation and doppler frequency estimation

A chirp is a signal in which the frequency increases ("up-chirp") or decreases ("down-chirp") with time. A common technique for many radar systems (usually also found in SAR systems) is to "chirp" the signal. In a chirped radar, this longer pulse also has a frequency shift during the pulse (hence the chirp or frequency shift). When the "chirped" signal is returned, it must be correlated with the

pulse sent.

The range chirp, noise, and properties of scattering object of a dataset across range can be observed from the range spectrum. The spectrum can be retrieved by applying a Fast Fourier Transform (FFT) to the raw dataset. The range spectrum is useful for estimating the signal-to-noise ratio, SNR, of the final image. Typically, the spectrum extends over about 80 percent of the digitized bandwidth. The SNR estimate is obtained by comparing the average level chirp bandwidth to the level in the noise region. This estimate is then used for radiometric compensation of the antenna pattern gain used for calibration of the SAR image, since the antenna gain correction applies only to the signal and not the noise fraction of the SAR image.

The frequency of received signals is shifting because of the Doppler Effect on relative motion between satellite and the target. From the azimuth wave spectrum of raw data, we can eliminate this effect by calculating the Doppler information. Ideally, the Doppler frequency of a target at the closest point, namely Doppler centroid, is zero. However, there is a non-zero Doppler condition because of the existence of squint angle. Earth rotation makes the effective body fixed spacecraft velocity different from the inertial velocity. Assuming that the radar antenna is oriented along the spacecraft inertial velocity, it results in an effective yaw angle for the antenna.

Typically, the Doppler effect is modeled as a linear function of slow time s:

$$f = f_{DC} + f_R \times s \tag{1}$$

where f is the instantaneous Doppler frequency, f_DC is the Doppler centroid, and f_R is the Doppler rate. The Doppler centroid and Doppler rate either calculated from orbit and attitude data or estimated in an automated fashion from the data itself.

There exist a number of algorithms to estimate the Doppler centroid frequency, for example multi-look cross correlation (MLCC) algorithm and multi-look beat frequency algorithm (MLBF) algorithm by Wong et al. (1996).

A.4 Range compression and azimuth prefilter

Range compression compresses all the energy distributed over the chirp duration into an narrow time window in the range chirp bandwidth. In this step, a filtering to match the transmitted pulse with the recorded data is applied, and the full range resolution is recovered. The resolution in slant range direction in SAR system depends on the transmitted pulse width (τ) :

$$\Delta R = \frac{c \times \tau}{2} = \frac{c}{2 \times \Delta f} \quad ; \quad \Delta x = \frac{c \times \tau}{2 \times \sin(\theta)} \tag{2}$$

where ΔR is resolution of slant range, f is chirp frequency changes, x is ground resolution, and θ is the incident angle.

From the equation (2), we know that smaller τ gives higher resolution, but requires more power. To deal this problem, a technique called chirp modulation is developed to compress the transmitted pulse width. Chirp modulation, or linear frequency modulation for digital communication was patented by Sidney Darlington in 1954 with significant later work performed by Winkler in 1962. This type of modulation employs sinusoidal waveforms whose instantaneous frequency increases or decreases linearly over time. In SAR, these waveforms are commonly referred to as linear chirps or simply chirps.

Prior to range compression it is possible to decimate the data in azimuth by pre-filtering around the Doppler centroid. This might be desirable if a quick-look survey product is desired. This step can also be performed after range compression.

A.5 Azimuth compression

Azimuth compression is the last important step in SAR data processing. This step is to focus the data in azimuth by considering the phase shift of the target as it moves through the aperture. Similar to the range compression, the azimuth compression can be carried out efficiently in the Doppler frequency domain. This is done as a complex multiplication of the azimuth reference function Doppler

spectrum with the range-compressed, range- migrated data. Finally, the result is transformed back to the time domain.

The resolution in azimuth direction depends on the physical antenna size:

$$\Delta A_{SAR} = \frac{L}{2} \tag{3}$$

where L is the antenna length.

A.6 Resultant SLC

The resolution is improved after the compression in both directions. In the final data file, each pixel is represented by a complex number with real part and imaginary part. In other words, the amplitude and the phase information can be extracted with this expression: $y = A \times e^{i\phi}$. The amplitude, has the information of electromagnetic backscattering, and the phase is a measurement of wrapped distance of a full wavelength between sensor and each pixel area.

Appendix B

Applications of SLC data over Southeast Asia

B.1 Ship Detection

Ship detection is an application using the amplitude of SAR images over the ocean area. Figure 2 is an example of ship detection using ERS image. The pink area, which has been recognized as ships, are successfully detected using a computer program. Since ship bodies are always built using metal material, a high reflected amplitude can be observed on the body. The easiest method has the following steps. Firstly, the land area is excluded from the image used for detection. Then, by using connected component analysis, highly reflected areas can be located. Finally, ship features are examined, including the shape of ship, and the speed detection using the water wave after the ship, etc... By using this

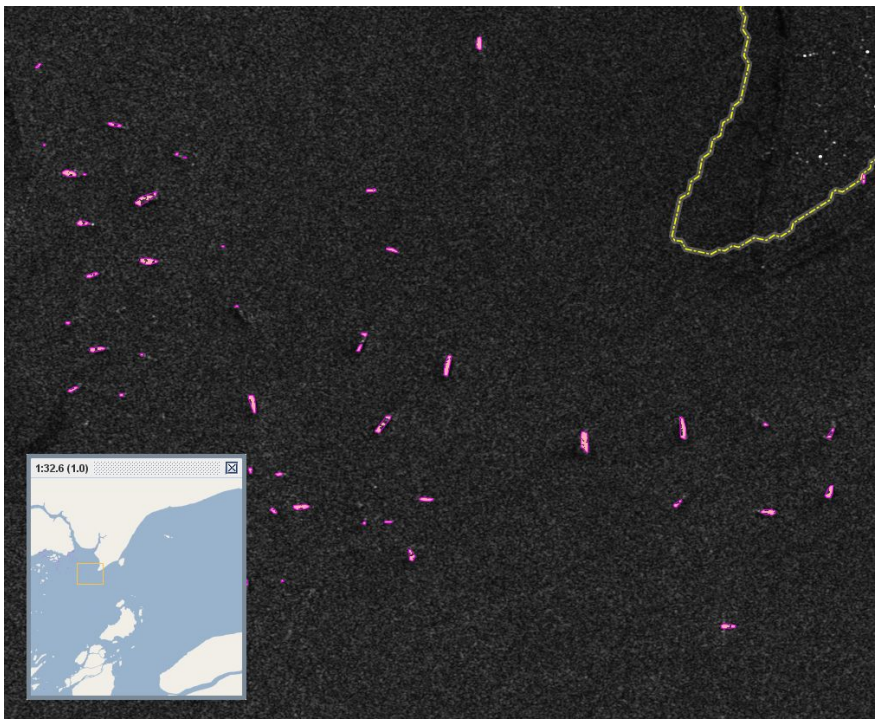
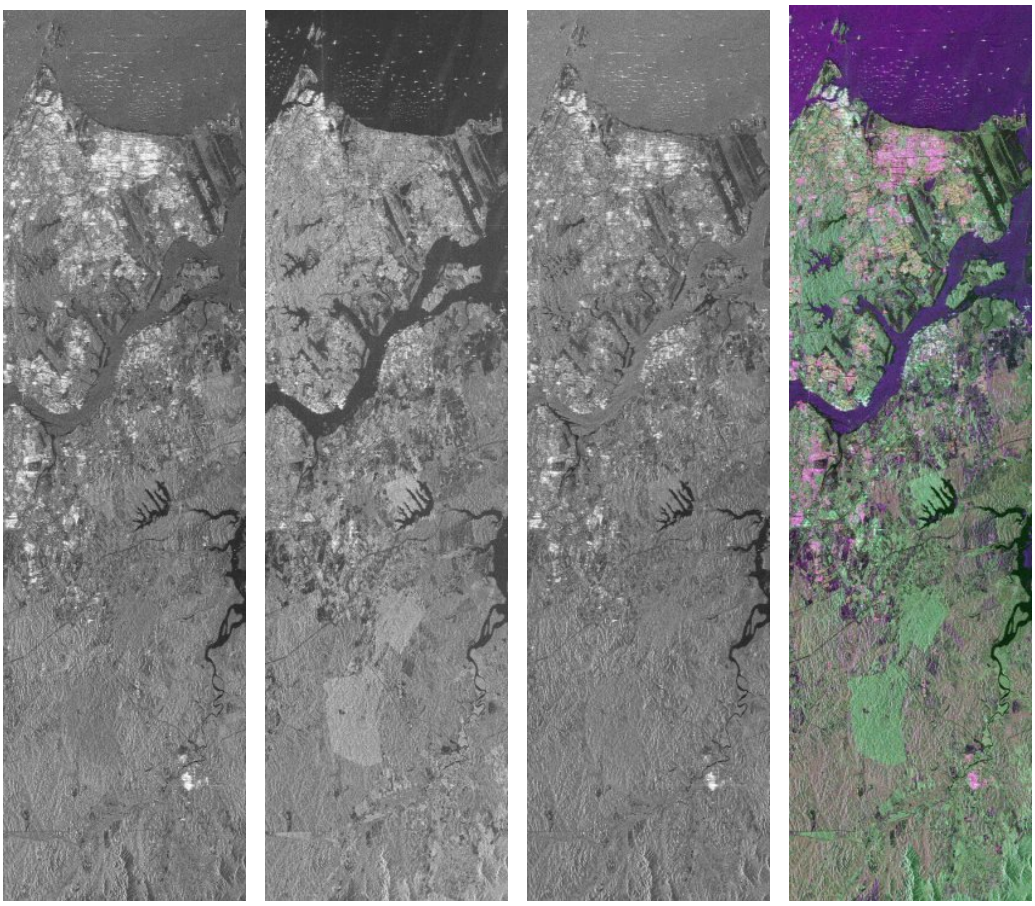


Figure 2: Ship detection using ERS amplitude Data

method, each ship will has a unique signature from SAR image, and SAR from the repeat-pass satellite or synchronous satellite will facilitate the tracking of the ship.

B.2 SAR polarimetry

As mentioned above, in polarimetry mode of ALOS PALSAR, a fully polarimetry data is used for classification. In a full polarimetry data, four image files are presented: HH, HV, VH and VV. Since different materials have the different polarimetry properties, which represent the ability to receive a certain polarization but reflect with another polarization. Therefore, the combination of these four



(a) Red: HH (b) Green: VH (c) Blue: VV (d) Polarimetry

Figure 3: Polarimetric PALSAR scene over part of Singapore and Johor Bahru (20090421)

band is like an optical image with color combination of Red, Green, and Blue (RGB).

Figure 3 shows a bands combination of polarimetry data (Red: HH, Green: VH, Blue:VV). PINK as city area, which has a very high reflectivity in HH, which we can observe easily the construction part of Singapore. GREEN as the forest area, meaning that the vegetation has a cross-polarimetry property when reflecting the radar signals, and the village which are neither pure construction

nor forest can be distinguished from both of them. DARK blue area as the reclamation land and airport. WHITE area is the harbor, which equally reflected band in HH, VH and VV.

B.3 Interferometry

The interferometry processing is the main content of this dissertation. Details are given in Chapter 3 and 4.

Appendix C

Evolution of Lusi Mud volcano

The following images are optical images taken from satellite, illustrating the evolution of the area of Lusi volcano by the satellite IKONOS.

Before and after the eruption that occurred the 29 May 2006. 3 months after, the quantity of mud is already huge. Later, the mud can be more or less dry, hence the different colors of the mud. The small road around the crater has disappeared during the year 2009.

(a) 20051006

(b) 20060829

(c) 20070105

(d) 20070605

(e) 20070807

(f) 20080105

(g) 20080505

(h) 20081011

(i) 20090214

(j) 20090626

Appendix D

ROIPAC flattening algorithm and steps

1. **compute** ρ_1, s_1, h**.**

   $\rho_1 = \rho_0 + (i-1)\Delta r$

   $s_1 = (j_a - 1)\Delta a$

   $h = h_0 + \dot{h}s_1 + \ddot{h}s_1^2$

   where ρ_0 is the range to the first range sample, Δr is the range sample spacing, Δa is the azimuth line spacing, h is the spacecraft height above the reference surface.

2. **compute** ρ_2 **(**$\Delta\rho_{ref} = 0$ **first time processing).**

   $\rho_2 = \rho_1 + \Delta\rho_0 + \Delta\rho_{ref}$

3. **compute** $\theta_{1ref}, \theta_{2ref}$**, approximate** b_c**,** b_s**.**

$cos\theta_{1ref} = \frac{\rho_1^2+(r+h)^2-r^2}{2\rho_1(r+h)}$

$cos\theta_{2ref} = \frac{\rho_2^2+(r+h)^2-r^2}{2\rho_2(r+h)}$

$b_c \approx b_{c0} + \dot{b}_c s_1$

$b_s = \Delta s_T = b_c tan\beta + \dot{b}_c \rho_2 sin\theta_2 - \dot{b}_h \rho_2 cos\theta_2$

where r is the radius of the earth (assumes spherical earth),

4. **compute b_c, b_h, b, $\hat{l}_{1ref} \cdot \vec{b}$, approximate $\Delta\rho_{ref}$.**

   $b_c = b_{c0} + \dot{b}_c(s_1 + \Delta s_T)$

   $b_h = b_{h0} + \dot{b}_h(s_1 + \Delta s_T)$

   $b = (b_s^2 + b_c^2 + b_h^2)^{1/2}$

   $\hat{l}_{1ref} \cdot \vec{b} = b_s sin\theta_{1ref} sin\beta + b_c sin\theta_{1ref} cos\beta - b_h cos\theta_{1ref}$

   $\Delta\rho_{ref} = \rho_1((1 - 2(\hat{l}_{1ref} \cdot \vec{b})\rho_1^{-1} + b^2\rho_1^{-2})^{1/2} - 1)$

5. **repeat steps $2-4$ for updated $\Delta\rho_{ref}$.**

6. **compute phase ϕ_{ref}.**

   $\phi_{ref} = \frac{4\pi}{\lambda}\Delta\rho_{ref}$

   where ϕ_{ref} is interferometric phase due to the reference surface.

7. **conjugate multiplication (subtract the phase ϕ_{ref}).**

   $C_{\phi_{ref}} = cos\phi_{ref} + jsin\phi_{ref}$

   $C_{\phi_{flat}} = C_\phi C^*_{\phi ref}$

where C_ϕ is the complex interferogram, and C_{flat} is the complex flattened interferogram.

Appendix E:

Python code to integrate

GAMMA software

A example of Python code to integrate Gamma software. The interferogram over Merapi Volcano are generated after running the code (*processMerapi.py*). The functions are called from *python.py*. Since *python.py* has more than 2000 lines, only some of the functions are shown:

processMerapi.py:

```
# -*- coding: utf-8 -*-

from gamma import *

dirResults = "Merapi"

rawimagedir="/home/christop/data2/Palsar"
slcdir = "/home/christop/data2/"+dirResults+"/slc"
interfdir = "/home/christop/data2/"+dirResults+"/interf"
demdir = "/home/christop/data2/"+dirResults+"/dem"
diffdir = "/home/christop/data2/"+dirResults+"/diff"
srtmdir = "/home/christop/data/SRTM"

images =[
("20070908","431_7030_20070908_FBD_11","ALPSRP086437030"),
("20071209","431_7030_20071209_FBS_11","ALPSRP099857030"),
("20080910","431_7030_20080910_FBD_11","ALPSRP140117030"),
("20070608","431_7030_20070608_FBD_11","ALPSRP073017030"),
("20080310","431_7030_20080310_FBS_11","ALPSRP113277030"),
("20080425","431_7030_20080425_FBD_11","ALPSRP119987030"),
("20080610","431_7030_20080610_FBD_11","ALPSRP126697030"),
("20071024","431_7030_20071024_FBS_11","ALPSRP093147030"),
```

```
("20080726","431_7030_20080726_FBD_11","ALPSRP133407030"),
("20090613","431_7030_20090613_FBD_11","ALPSRP180377030"),
("20080124","431_7030_20080124_FBS_11","ALPSRP106567030")
    ]

#images =[("20080425","A0900998-001","ALPSRP119987030"),
#           ("20080610","A0900998-002","ALPSRP126697030"),
#           ("20080726","A0900998-003","ALPSRP133407030"),
#           ("20080910","A0900998-004","ALPSRP140117030"),
#           ("20070608","A0903125-001","ALPSRP073017030"),
#           ("20070908","A0903125-002","ALPSRP086437030"),
#           ("20090613","A0903125-003","ALPSRP180377030"),
#           ("20071024","A0903097-001","ALPSRP093147030"),
#           ("20071209","A0903097-002","ALPSRP099857030"),
#           ("20080124","A0903097-003","ALPSRP106567030"),
#           ("20080310","A0903097-004","ALPSRP113277030")]


g = Gamma(rawimagedir=rawimagedir, slcdir=slcdir, interfdir=interfdir, demdir=
    demdir, diffdir=diffdir, srtmdir=srtmdir)

for image in images:
  #print "*****************************************************"
  #print "******* SLC/MLI/DEM generation: "+image[0]
  #print "*****************************************************"
  g.generateSLC(image)
  g.generateOversampled(image[0])
  g.generateMLI(image[0])
  g.generateDEM(image[0])

interfList = []
for imageMaster in images:
  for imageSlave in images:
    if (imageMaster[0] < imageSlave[0]):
      interfList.append((imageMaster[0], imageSlave[0]))

for master, slave in interfList:
  g.generateInterferogram(master, slave)
  g.refineBaseline(master, slave)
  g.generate2passDiff(master, slave)
  #g.generatePerpendicularBaseline(master, slave) #optional
  #g.generatePerpendicularBaselineRough(master, slave)
  #g.flattenInterferogram(master, slave)
  #g.filterInterferogram(master, slave)
  #g.unwrapInterferogramMcf(master, slave)
  #g.unwrapInterferogramTree(master, slave)
  #g.unflattenInterferogram(master, slave)
```

gamma.py:

```
# -*- coding: utf-8 -*-
import os, sys, math
import re, time, pylab, pickle
import numpy as np
from time import gmtime, strftime
from multiprocessing import Process
from graphProcess import *
from array import array
import matplotlib.pyplot as plt
import datetime
from matplotlib.dates import drange
from matplotlib.dates import *


## global constants ##
MRange = 2
MAzimuth = 5

CCthredMask =0.3
CCthredBase = 0.85
```

```
class Gamma:

  def __init__(self, level0imagedir = None, rawimagedir = None, slcdir = None,
      interfdir = None, demdir = None, diffdir = None, srtmdir = None,
      iterinterfdir = None, plotdir = None, imagedir = None, IPTAdir = None,
      resamplingdir = None, executeCommand=True, removeintermediatefiles = False
      ):
    self.level0imagedir = level0imagedir
    self.rawimagedir = rawimagedir
    self.slcdir = slcdir
    self.interfdir = interfdir
    self.demdir = demdir
    self.diffdir = diffdir
    self.srtmdir = srtmdir
    self.executeCommand = executeCommand
    self.iterinterfdir = iterinterfdir
    self.plotdir = plotdir
    self.imagedir = imagedir
    self.IPTAdir = IPTAdir
    self.resamplingdir = resamplingdir
    self.removeintermediatefiles = removeintermediatefiles

    self.graphproc = GraphProcess()


  def gammaCommand(self, *argv):
    """ Format the calls to Gamma
    """
    print "****************************************"
    #print "Calling gammaCommand with", len(argv), "arguments:", argv
    #print "****************************************"
    self.graphproc.addCommand(argv)
    cmd = " ".join(str(arg) for arg in argv)
    print cmd
    if (self.executeCommand):
      logfile = open('processing_history.log', 'a')
      logfile.write('# '+ strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())+'\n
          ')
      logfile.write(cmd+'\n\n')
      logfile.close()
      os.system(cmd)


  def getparameter(self, parameterFile, param):
    f = open(parameterFile)
    lines = f.readlines()
    for line in lines:
      linesplit = line.split()
      if (len(linesplit) > 1):
        if (linesplit[0] == param):
          parameter = linesplit[1]
    f.close()
    return parameter

#######################################################################
#######################     SLC PROCESSING     ########################
#######################################################################

  def generateSLC(self, image):
    """ Generate the SLC in the Gamma format
        Requires:
        - Alos level 1.1 data
    """
    imageID = image[0]
    subfolder = image[1]
    imageBaseName =  image[2]
```

```
    if (len(image) > 3):
      polar = image[3]
      workingdir = self.slcdir+"/"+imageID+image[3]
      slcParameterFile = imageID+image[3]+".slc_par"
      slcFile = imageID+image[3]+".slc"
    else:
      polar = 'HH'
      workingdir = self.slcdir+"/"+imageID
      slcParameterFile = imageID+".slc_par"
      slcFile = imageID+".slc"

    cmd = "mkdir -p "+workingdir
    os.system(cmd)
    os.chdir(workingdir)
    # Check file existence supposing Asc orbit ##
    leaderFilename = self.rawimagedir+"/"+subfolder+"/"+"LED-"+imageBaseName+"-
        H1.1__A"
    dataFilename = self.rawimagedir+"/"+subfolder+"/"+"IMG-"+polar+"-"+
        imageBaseName+"-H1.1__A"
    if not (os.path.exists(leaderFilename) and os.path.exists(dataFilename)):
      ## If not, check Desc orbit ##
      leaderFilename = self.rawimagedir+"/"+subfolder+"/"+"LED-"+imageBaseName
          +"-H1.1__D"
      dataFilename = self.rawimagedir+"/"+subfolder+"/"+"IMG-"+polar+"-"+
          imageBaseName+"-H1.1__D"
      if not (os.path.exists(leaderFilename) and os.path.exists(dataFilename)):
        leaderFilename = self.rawimagedir+"/"+subfolder+"/"+"LED-"+imageBaseName
            +"-P1.1__A"
        dataFilename = self.rawimagedir+"/"+subfolder+"/"+"IMG-"+polar+"-"+
            imageBaseName+"-P1.1__A"
        if not (os.path.exists(leaderFilename) and os.path.exists(dataFilename))
            :
          print "Error, can't find the leader and data files"


    self.gammaCommand("par_EORC_PALSAR", leaderFilename, slcParameterFile,
        dataFilename, slcFile)

  def processlevel0(self, imageID):
    subfolder = "SCENE01"
    workingdir = self.rawimagedir+"/"+imageID+"/"
    cmd = "mkdir -p "+workingdir
    os.system(cmd)
    os.chdir(workingdir)

    #####Define file name
    leaderFilename = self.level0imagedir+"/"+imageID+"/"+subfolder+"/"+"LEA_01
        .001"
    dataFilename = self.level0imagedir+"/"+imageID+"/"+subfolder+"/"+"DAT_01
        .001"

    fixedFilename = imageID +".fix"
    MSPParameterFile = "p"+imageID +".slc_par"
    DopplerAmbigFile = "dop_ambig.dat"
    AzimuthSpecFile = imageID +".azsp"
    DopplerCentroidFile = imageID +".dop"
    RangeSpecFile = imageID +".rspec"
    RangeCompressedFile = imageID +".rc"
    AzimuthFocusFile = imageID + ".autof"
    SLCFile = imageID + ".slc"
    MLIparameterFile = imageID + ".mli_par"
    MLIFile = imageID + ".mli"

    tempSLCParameterFile = imageID +"tempslc_par"

    #####Define Sensor Reading Method
    if (imageID[0:2] == "E1"):
      SensorParamemterFile= "/home/christop/software/GAMMA_SOFTWARE-20091214/MSP
          /sensors/ERS1_ESA.par"
      orbdir="/home/christop/data/ERS-Orbits/ERS-1/dgm-e04"
```

```
    AntennagainFile = "/home/christop/software/GAMMA_SOFTWARE-20091214/MSP/
        sensors/ERS1_antenna.gain"
    Fcomplexfactor = "-12.5"
    Scomplexfactor = "47.5"
  elif (imageID[0:2] == "E2"):
    SensorParamemterFile= "/home/christop/software/GAMMA_SOFTWARE-20091214/MSP
        /sensors/ERS2_ESA.par"
    orbdir="/home/christop/data/ERS-Orbits/ERS-2/dgm-e04"
    AntennagainFile = "/home/christop/software/GAMMA_SOFTWARE-20091214/MSP/
        sensors/ERS2_antenna.gain"
    Fcomplexfactor = "-2.8"
    Scomplexfactor = "57.2"
  else:
    print "Error, not a ERS1 or ERS2 sensor"

  if not (os.path.exists(leaderFilename) and os.path.exists(dataFilename)):
    print "Error, can't find the leader and data files"

  ####Generate MSP Processing parameter file
  cmd = "echo \"%(imageID)s\n\n\n\n\n\n\n\n\n\n\" > ERS_proc_CRISP.in" % {'
      imageID':imageID}
  os.system(cmd)
  self.gammaCommand("ERS_proc_CRISP", leaderFilename,  MSPParameterFile, "<
      ERS_proc_CRISP.in")

  ####Conditioning of Raw Data
  self.gammaCommand("ERS_fix", "ESA/ESRIN", SensorParamemterFile,
      MSPParameterFile, "1", dataFilename, fixedFilename)

  ####Manipulation of orbits
  self.gammaCommand("DELFT_proc2", MSPParameterFile,  orbdir)

  #####Determine the Doppler ambiguity
  self.gammaCommand("dop_ambig", SensorParamemterFile, MSPParameterFile,
      fixedFilename, "2", "-", DopplerAmbigFile)

  #####Determine the flactional Doppler Centroid
  self.gammaCommand("azsp_IQ", SensorParamemterFile, MSPParameterFile,
      dataFilename, AzimuthSpecFile)

  #####Estimate Doppler Centroid across the swath
  self.gammaCommand("doppler_2d", SensorParamemterFile, MSPParameterFile,
      fixedFilename, DopplerCentroidFile)

  #####Estimate Range power spectrum
  self.gammaCommand("rspec_IQ", SensorParamemterFile, MSPParameterFile,
      dataFilename, RangeSpecFile)

  #####Range Compression
  self.gammaCommand("pre_rc", SensorParamemterFile, MSPParameterFile,
      dataFilename, RangeCompressedFile)

  #####Autofocus
  self.gammaCommand("autof", SensorParamemterFile, MSPParameterFile,
      RangeCompressedFile, AzimuthFocusFile, "2.0")
  self.gammaCommand("autof", SensorParamemterFile, MSPParameterFile,
      RangeCompressedFile, AzimuthFocusFile, "2.0")

  #####Azimuth Compression
  cmd = "cp "+AntennagainFile+" ."
  os.system(cmd)
  self.gammaCommand("az_proc", SensorParamemterFile, MSPParameterFile,
      RangeCompressedFile, SLCFile, "4096", "1", Scomplexfactor, "-", "-")

  #####SLC Image detection and generation of Multi-look intensity imagedir
  self.gammaCommand("multi_SLC", MSPParameterFile, MLIparameterFile, SLCFile,
      MLIFile, "2", "5", "1")

  #####generateTempSLCParameterFile
```

```
        self.gammaCommand("par_MSP", SensorParamemterFile, MSPParameterFile,
            tempSLCParameterFile, "1")


    def generateInterferogram(self, imageIDmaster, imageIDslave, multilookRange=2,
        multilookAzimuth=5):
        """ To generate the interferogram master_salve.int
        """
        coupleName = imageIDmaster+"_"+imageIDslave
        workingdir = self.interfdir+'/'+coupleName
        cmd = "mkdir -p "+workingdir
        os.system(cmd)
        os.chdir(workingdir)

        offsetFile = coupleName+".off"
        cmd = "rm -f "+offsetFile #Make sure that the file does not exist already
        os.system(cmd)

        # inputs #
        masterSlcParameter = self.slcdir+"/"+imageIDmaster+"/"+imageIDmaster+".
            oslc_par"
        masterSlc = self.slcdir+"/"+imageIDmaster+"/"+imageIDmaster+".oslc"
        slaveSlcParameter = self.slcdir+"/"+imageIDslave+"/"+imageIDslave+".oslc_par
            "
        slaveSlc = self.slcdir+"/"+imageIDslave+"/"+imageIDslave+".oslc"

        width = self.findImageWidth(imageIDmaster,fileSuffix=".oslc_par")
        #Generate offsets
        self.generateOffsets(coupleName, masterSlc, masterSlcParameter, slaveSlc,
            slaveSlcParameter, offsetFile)

        #Resample slave
        slaveResampleSlcParameter = imageIDslave+".rslc_par"
        slaveResampleSlc = imageIDslave+".rslc"
        self.gammaCommand("SLC_interp", slaveSlc, masterSlcParameter,
            slaveSlcParameter, offsetFile, slaveResampleSlc,
            slaveResampleSlcParameter)

        #Generate interferogram
        interferogramFile = coupleName+".int"
        self.gammaCommand("SLC_intf", masterSlc, slaveResampleSlc,
            masterSlcParameter, slaveResampleSlcParameter, offsetFile,
            interferogramFile, multilookRange, multilookAzimuth, "-", "-", 1, 1)

        #Generate baseline
        roughBaselineFile = coupleName+".rough.base"
        baselineFile = coupleName+".base"

        cmd = "rm -f "+roughBaselineFile #Make sure that the file does not exist
            already
        os.system(cmd)
        #self.gammaCommand("base_orbit", masterSlcParameter,
            slaveResampleSlcParameter, roughBaselineFile)
        #self.gammaCommand("base_init", masterSlcParameter, slaveSlcParameter,
            offsetFile, interferogramFile, roughBaselineFile, 0, 1024, 1024)
        self.gammaCommand("base_orbit", masterSlcParameter,
            slaveResampleSlcParameter, roughBaselineFile)
        self.gammaCommand("cp", roughBaselineFile, baselineFile)

        ## remove temporary files ##
        if (self.removeintermediatefiles) :
          cmd = "rm -f coffs coffsets offs offsets snr create_offset.in "
          os.system(cmd)

    def flattenInterferogram(self, imageIDmaster, imageIDslave, imageID3=None,
        multilookRange=2, multilookAzimuth=5, rough=False):
        coupleName = imageIDmaster+"_"+imageIDslave
        workingdir = self.interfdir+'/'+coupleName
        os.chdir(workingdir)
        # inputs #
```

```
    if (imageID3==None):
      masterSlcParameter = self.slcdir+"/"+imageIDmaster+"/"+imageIDmaster+".
          oslc_par"
    else:
      ##in this case imageID3 is the global master and we deal with rscl files##
      masterSlcParameter = self.interfdir+"/"+coupleName+"/"+imageIDmaster+".
          rslc_par"
    interferogramFile = coupleName+".int"
    offsetFile = coupleName+".off"
    if (rough):
      baselineFile = coupleName+".rough.base"
    else:
      baselineFile = coupleName+".base"
    masterMli, masterMliParameter = self.getMLIname(imageIDmaster,
        multilookRange = multilookRange, multilookAzimuth= multilookAzimuth)
    masterMli = self.slcdir+"/"+imageIDmaster+"/"+masterMli
    if (imageID3==None):
      width = self.findImageWidth(imageIDmaster,fileSuffix=".oslc_par")
    else:
      width = self.findRSLCwidth(imageIDmaster,imageIDslave)
    RasWidth=str(int(width)/multilookRange)
    # output #
    interferogramFlattened = coupleName+".flt"
    self.gammaCommand("ph_slope_base", interferogramFile, masterSlcParameter,
        offsetFile, baselineFile, interferogramFlattened)
    self.gammaCommand("rasmph_pwr", interferogramFlattened, masterMli, RasWidth)

  def filterInterferogram(self, imageIDmaster, imageIDslave, imageID3=None,
      fileSuffix=".flt", multilookRange=2, multilookAzimuth=5, strong=False):
    coupleName = imageIDmaster+"_"+imageIDslave
    workingdir = self.interfdir+'/'+coupleName
    os.chdir(workingdir)
    # inputs #
    slaveResampleSlcParameter = imageIDslave+".rslc_par"
    slaveResampleSlc = imageIDslave+".rslc"
    interferogramFlattened = coupleName+fileSuffix

    slaveResampleMli, slaveResampleMliParameter = self.getResampleMLIname(
        imageIDslave, multilookRange = multilookRange, multilookAzimuth=
        multilookAzimuth)
    self.gammaCommand("multi_look", slaveResampleSlc, slaveResampleSlcParameter,
        slaveResampleMli, slaveResampleMliParameter, multilookRange,
        multilookAzimuth)
    ## if imageID3=!None, we need to resample the master MLI image (in this case
        we deal with a slave1_slave2 interferogram) and imageID3 represents the
        global master image
    if (imageID3==None) :
      masterMli, masterMliParameter = self.getMLIname(imageIDmaster,
          multilookRange = multilookRange, multilookAzimuth= multilookAzimuth)
      masterMli = self.slcdir+"/"+imageIDmaster+"/"+masterMli
      width = self.findImageWidth(imageIDmaster,fileSuffix=".oslc_par")
    else :
      masterMli, masterMliParameter = self.getResampleMLIname(imageIDmaster,
          multilookRange = multilookRange, multilookAzimuth= multilookAzimuth)
      masterResampleSlcParameter = imageIDmaster+".rslc_par"
      masterResampleSlc = imageIDmaster+".rslc"
      self.gammaCommand("multi_look", masterResampleSlc,
          masterResampleSlcParameter, masterMli, masterMliParameter,
          multilookRange, multilookAzimuth)
      width = self.findRSLCwidth(imageIDmaster,imageIDslave)

    imageWidth = width / multilookRange
    # outputs #
    coherenceFile = coupleName+".cc"
    interferogramFiltered = coupleName+fileSuffix+"_sm"
    coherenceFiltered = coupleName+".smcc"

    self.gammaCommand("cc_wave", interferogramFlattened, masterMli,
        slaveResampleMli, coherenceFile, imageWidth, 5, 5, 1)
```

```
    if (strong):
      self.gammaCommand("adf", interferogramFlattened, interferogramFiltered,
          coherenceFiltered, imageWidth, 0.7, 256, 7, 64, 0, 0, 0.500) #FIXME
          this is the nice (but long processing for the final diff interf) EDIT
          faster with the 64 blk size
#      self.gammaCommand("adf", interferogramFlattened, interferogramFiltered,
    coherenceFiltered, imageWidth, 0.7, 128, 7, 64, 0, 0, 0.500)
    else:
      self.gammaCommand("adf", interferogramFlattened, interferogramFiltered,
          coherenceFiltered, imageWidth, 0.5, 32, 7, 4, 0, 0, 0.700)

    #generate raster output
    self.gammaCommand("rasmph", interferogramFiltered, imageWidth)


  def unwrapInterferogramMcf(self, imageIDmaster, imageIDslave, imageID3=None,
      fileSuffix=".flt_sm", multilookRange=MRange, multilookAzimuth=MAzimuth,
      split=False):
    """ Unwrap the interferogram (on iron)
    """
    coupleName = imageIDmaster+"_"+imageIDslave
    workingdir = self.interfdir+'/'+coupleName
    os.chdir(workingdir)
    # input #
    interferogramFiltered = coupleName+fileSuffix
    ## if imageID3, we need to resample the master MLI image (in this case we
        deal with a slave1_slave2 interferogram) and imageID3 represents the
        global master image
    if (imageID3==None) :
      masterMli, masterMliParameter = self.getMLIname(imageIDmaster,
          multilookRange = multilookRange, multilookAzimuth= multilookAzimuth)
      masterMli = self.slcdir+"/"+imageIDmaster+"/"+masterMli
      width = self.findImageWidth(imageIDmaster,fileSuffix=".oslc_par")
    else :
      masterMli, masterMliParameter = self.getResampleMLIname(imageIDmaster,
          multilookRange = multilookRange, multilookAzimuth= multilookAzimuth)
      masterMli = self.interfdir+"/"+coupleName+"/"+masterMli
      masterMliParameter = self.interfdir+"/"+coupleName+"/"+masterMliParameter
      masterResampleSlcParameter = imageIDmaster+".rslc_par"
      masterResampleSlc = imageIDmaster+".rslc"
      self.gammaCommand("multi_look", masterResampleSlc,
          masterResampleSlcParameter, masterMli, masterMliParameter,
          multilookRange, multilookAzimuth)
      width = self.findRSLCwidth(imageIDmaster,imageIDslave)

    imageWidth = width / multilookRange
    # output #
    maskFile = coupleName+".mask.ras"
    maskThinnedFile = coupleName+".mask_thinned.ras"
    interferogramUnwrappedNonInterporlated = coupleName+fileSuffix+".mcf_unw"
    interferogramUnwrappedInterp = coupleName+fileSuffix+"_unw_interp"

    cmd = "rm -f "+ maskFile +" " + maskThinnedFile +" "+
        interferogramUnwrappedInterp#Make sure that the files d not exist
        already
    os.system(cmd)

    coherenceFiltered = coupleName+".smcc" #FIXME the smcc can correspond to
        different processes (add fileSuffix?)
    interferogramUnwrappedModel = coupleName+fileSuffix+"_unw"

    cmd = "rm -f "+ maskFile +" "+ interferogramUnwrappedModel#Make sure that
        the files do not exist already
    os.system(cmd)

    self.gammaCommand("rascc_mask", coherenceFiltered, masterMli, imageWidth, 1,
         1, 0, 1, 1, 0.3, "-", "-", "-", "-", "-", maskFile)
    #look at p 60 of the ISP_user_guide (optional thinning)
    #TODO look at this command more into details
```

```
    self.gammaCommand("rascc_mask_thinning", maskFile, coherenceFiltered,
        imageWidth, maskThinnedFile, 3, 0.3, 0.5, 0.7)

    numPatch = 1
    if (split):
      numPatch = 2

    self.gammaCommand("mcf", interferogramFiltered, coherenceFiltered,
        maskThinnedFile, interferogramUnwrappedNonInterporlated, imageWidth, 1,
        9, 18, "-", "-", numPatch, numPatch, "-", "-", "-" )
    self.gammaCommand("interp_ad", interferogramUnwrappedNonInterporlated,
        interferogramUnwrappedInterp, imageWidth, 32, 8, 16, 2, 2)

    self.gammaCommand("unw_model", interferogramFiltered,
        interferogramUnwrappedInterp, interferogramUnwrappedModel, imageWidth)

    #generate raster output
    self.gammaCommand("rasrmg", interferogramUnwrappedModel, masterMli,
        imageWidth, 1, 1, 0, 1, 1, 0.5, 1.0, 0.35, 0.0, 1)

    # remove _unw_interp we will not need any more
    if (self.removeintermediatefiles) :
      cmd = "rm -f "+interferogramUnwrappedInterp
      os.system(cmd)


  def generateElevationFromInterf(self, imageIDmaster, imageIDslave):
    coupleName = imageIDmaster+"_"+imageIDslave

    workingdir = self.interfdir+'/'+coupleName
    os.chdir(workingdir)

    masterSlcParameter = self.slcdir+"/"+imageIDmaster+"/"+imageIDmaster+".
        oslc_par"

    offsetFile = coupleName+".off"
    baselineFile = coupleName+".base"
    interferogramUnwrapped = coupleName+".flt_sm_unw_interp"
    masterMli = self.slcdir+"/"+imageIDmaster+"/"+imageIDmaster+".mli"

    heightMap = coupleName+".hgt"
    crosstrackGroundRange = coupleName+".grd"

    self.gammaCommand("hgt_map", interferogramUnwrapped, masterSlcParameter,
        offsetFile, baselineFile, heightMap, crosstrackGroundRange, "1")
    #self.gammaCommand("dishgt", heightMap, masterMli, 4640)
```

Appendix F:

Matlab code for interferogram processing of ALOS PALSAR Level 1.1 data

This chapter shows the Matlab code for interferogram processing. The follows are the instructions of using the codes:

1. These codes are running in sequence.

2. If Level 1.1 PALSAR data are available, change the data and leader file name pointers (the first lines) in *AlosBasic.m*.

3. Fill in the specific area of interest which you want to process.

AlosBasic.m:

```
fdatm = fopen('A0805750-001/IMG-HH-ALPSRP074917120-H1.1__A','rb');
fleam = fopen('A0805750-001/LED-ALPSRP074917120-H1.1__A','rb');
fdats = fopen('A0805750-002/IMG-HH-ALPSRP081627120-H1.1__A','rb');
```

```
fleas = fopen('A0805750-002/LED-ALPSRP081627120-H1.1__A','rb');

d1 = fread(fdatm, 180, 'int8');
d2 = fread(fdatm, 6, 'int8');
d3 = fread(fdatm, 6, 'int8');
d4 = fread(fdatm, 24, 'int8');
d5 = fread(fdatm, 4, 'int8');
d6 = fread(fdatm, 4, 'int8');
d7 = fread(fdatm, 4, 'int8');
d8 = fread(fdatm, 4, 'int8');


Row_M = sscanf(char(d2),'%d');
Number_Of_Record_M = Row_M+1;
Record_Length_M = sscanf(char(d3),'%d');
Number_Of_Bits_PerSample_M = sscanf(char(d5),'%d');
Number_Of_Sample_PerPixel_M = sscanf(char(d6),'%d');
Number_Of_Bytes_PerPixel_M = sscanf(char(d7),'%d');
Column_M = (Record_Length_M-412)/(Number_Of_Bits_PerSample_M/8)/
    Number_Of_Sample_PerPixel_M;
Pixel_Number_M = Row_M*Column_M;
Number_Of_SarData_Perrecord_M = Record_Length_M-12;
Total_Size_M = (Row_M+1)*Record_Length_M;

d9 = fread(fdatm, 720-232+412, 'int8');

clear d9;


Out00 = sprintf(==================Master_Image_Information==========\n','r');
Out0 = sprintf('Demo~~~Sar Data Reader','r');
Out1 = sprintf('Total Size of Data File: %d bits (%dBytes=%.2fMB)',
    Number_Of_Record_M*Record_Length_M ,Number_Of_Record_M*Record_Length_M/8,
    Number_Of_Record_M*Record_Length_M/1024/1024);
Out2 = sprintf('Number of bits per sample: %d (%d Bytes)',
    Number_Of_Bits_PerSample_M, Number_Of_Bits_PerSample_M/8);
Out3 = sprintf('Number of sample per data group (or pixels): %d',
    Number_Of_Sample_PerPixel_M);
Out4 = sprintf('Number of Bytes per data group (or pixels): %d',
    Number_Of_Bytes_PerPixel_M);
Out5 = sprintf('Number of Sar Data Record: %d',Row_M);
Out6 = sprintf('SAR Data record length: %d', Record_Length_M);
Out7 = sprintf('Size of Image: \n Row: %d',Row_M);
Out8 = sprintf(' Column: %d',Column_M);
Out9 = sprintf(' Number of Pixels: %d',Pixel_Number_M);

disp(Out00)
disp(Out0)
disp(Out1)
disp(Out2)
disp(Out3)
disp(Out4)
disp(Out5)
disp(Out6)
disp(Out7)
disp(Out8)
disp(Out9)

t=15;
r= rem(Column_M*2,t*2);
s=sprintf('%d*float32=>float16', 2);

p= fseek(fleam,2254,-1);
TDI= fread(fleam, 8,'8*uint8=>char');
char TDILM;
TDILM= sscanf(char(TDI),'%s');

Out10 = sprintf(' Time direction indicator along line direction: %s',TDILM);
disp(Out10)
```

```
clear TDI

for z=1:1:uint32((Row_M/t)-2)
    y=double(((uint32(z)-1)*uint32(t)+1)*uint32(Record_Length_M)+1132);
    p= fseek(fdatm,y,-1);
    p= fread(fdatm, double(int16(Column_M/t-0.5)*2), s,
        Number_Of_Bytes_PerPixel_M*(t-1), 'b');
    Com(z,1:int16(Column_M/t-0.5)*2)=p';
end

    if TDILM=='ASCEND'
    Com = rot90(Com',1);
end


Re =  single(Com(:,1:2:int16(Column_M/t-0.5)*2));
Com =  single(Com(:,2:2:int16(Column_M/t-0.5)*2));
Com(:,int16(Column_M/t-0.5)) = 0;
Com = Re+i*Com;
clear Re
av=abs(Com).^2;
clear Com
avs=sum(histc(av,0:200000000:max(max(av)')');
k= find(avs<max(avs)/1000);
s=size(avs);

for i= 1:(s(2)-100)
    st(i)=mean(avs(i:(i+20)));
end

k= find(st<max(st)/700);
maxx=min(k);
r1=st(1);
clear k st;
k1=200000000*maxx;

iptsetpref('ImshowAxesVisible','on')
imshow(av,[1 k1])
figure, plot(avs(1:(1.5*maxx)))
hold on
plot([maxx maxx],[0 r1],'r','Linewidth',1)
hold off

clear av
clear avs
clear Out0 Out1 Out2 Out3 Out4 Out5 Out6 Out7 Out8 Out9
clear d1 d2 d3 d4 d5 d6 d7 d8
clear i maxx
clear r r1 s t y z p
```

AlosM.m:

```
clear ComM
s1=sprintf('Upper Coordinate (%d-%d): ',1,Row_M);
upperM=input(s1);
s2=sprintf('Lower Coordinate (%d-%d): ',upperM,Row_M);
lowerM=input(s2);
s3=sprintf('Left Coordinate (%d-%d): ',1,Column_M);
leftM=input(s3);
s4=sprintf('Right Coordinate (%d-%d): ',leftM,Column_M);
rightM=input(s4);


if TDILM=='ASCEND'
    upper1=upperM;
    upperM=Row_M-lowerM;
    lowerM=Row_M-upper1;
end


y=15;
```

```
RR1=[upperM:y:(lowerM-y) lowerM]';
RR2=[leftM:y:(rightM-y) rightM]';


for xxx = 1:size(RR1,1)
for zzz = 1:size(RR2,1)
    RegisterM((xxx-1)*size(RR2,1)+zzz,:)=[RR1(xxx),RR2(zzz)];
end
end

CoorRegist=PixToCoor(RegisterM,fleam);
RegisterS=CoorToPix(CoorRegist,fleas);

figure
s=sprintf('%d*float32=>float16', 2);

for z=1:1:uint32(lowerM-upperM+1)
    y=double((upperM+z-1)*uint32(Record_Length_M)+(leftM-1)*
        Number_Of_Bytes_PerPixel_M+1132);
    p=fseek(fdatm,y,-1);
    p=fread(fdatm, double(int16(rightM-leftM+1)*2), s, 0, 'b');
    ComM(z,1:int16(rightM-leftM+1)*2)=p';
end

clear z y p

if TDILM=='ASCEND'
    ComM = rot90(ComM',1);
end


clear i

Re =  single(ComM(:,1:2:int16(rightM-leftM+1)*2));
ComM =  single(ComM(:,2:2:int16(rightM-leftM+1)*2));
ComM = Re+i*ComM;
clear Re

imshow(abs(ComM).^2,[1 k1])
clear y z xxx zzz upper1 s1 s2 s3 s4 s p RR1 RR2
```

AlosS.m:

```
fdats = fopen('A0805750-002/IMG-HH-ALPSRP081627120-H1.1__A','rb');
fleas = fopen('A0805750-002/LED-ALPSRP081627120-H1.1__A','rb');

d1 = fread(fdats, 180, 'int8');
d2 = fread(fdats, 6, 'int8');
d3 = fread(fdats, 6, 'int8');
d4 = fread(fdats, 24, 'int8');
d5 = fread(fdats, 4, 'int8');
d6 = fread(fdats, 4, 'int8');
d7 = fread(fdats, 4, 'int8');
d8 = fread(fdats, 4, 'int8');


Row_S = sscanf(char(d2),'%d');
Number_Of_Record_S = Row_S+1;
Record_Length_S = sscanf(char(d3),'%d');
Number_Of_Bits_PerSample_S = sscanf(char(d5),'%d');
Number_Of_Sample_PerPixel_S = sscanf(char(d6),'%d');
Number_Of_Bytes_PerPixel_S = sscanf(char(d7),'%d');
Column_S = (Record_Length_S-412)/(Number_Of_Bits_PerSample_S/8)/
    Number_Of_Sample_PerPixel_S;
Pixel_Number_S = Row_S*Column_S;
Number_Of_SarData_Perrecord_S = Record_Length_S-12;
Total_Size_S = (Row_S+1)*Record_Length_S;

d9 = fread(fdats, 720-232+412, 'int8');
```

```
clear d9;

Out00 = sprintf('==================Slave_Image_Information==========/n/n','r');
Out0 = sprintf('Demo~~~Sar Data Reader','r');
Out1 = sprintf('Total Size of Data File: %d bits (%dBytes=%.2fMB)',
    Number_Of_Record_S*Record_Length_S ,Number_Of_Record_S*Record_Length_S/8,
    Number_Of_Record_S*Record_Length_S/1024/1024);
Out2 = sprintf('Number of bits per sample: %d (%d Bytes)',
    Number_Of_Bits_PerSample_S, Number_Of_Bits_PerSample_S/8);
Out3 = sprintf('Number of sample per data group (or pixels): %d',
    Number_Of_Sample_PerPixel_S);
Out4 = sprintf('Number of Bytes per data group (or pixels): %d',
    Number_Of_Bytes_PerPixel_S);
Out5 = sprintf('Number of Sar Data Record: %d',Row_S);
Out6 = sprintf('SAR Data record length: %d', Record_Length_S);
Out7 = sprintf('Size of Image: \n Row: %d',Row_S);
Out8 = sprintf(' Column: %d',Column_S);
Out9 = sprintf(' Number of Pixels: %d',Pixel_Number_S);

disp(Out00)
disp(Out0)
disp(Out1)
disp(Out2)
disp(Out3)
disp(Out4)
disp(Out5)
disp(Out6)
disp(Out7)
disp(Out8)
disp(Out9)

t=15;
r= rem(Column_S*2,t*2);
s=sprintf('%d*float32=>float16', 2);

p= fseek(fleas,2254,-1);
TDI= fread(fleas, 8,'8*uint8=>char');
char TDILS;
TDILS= sscanf(char(TDI),'%s');

Out10 = sprintf(' Time direction indicator along line direction: %s',TDILS);
disp(Out10)
clear TDI


maxxS=max(RegisterS);
mixxS=min(RegisterS);
upperS=mixxS(1);
lowerS=maxxS(1);
leftS=mixxS(2);
rightS=maxxS(2);


figure
s=sprintf('%d*float32=>float16', 2);


for z=1:1:uint32(lowerS-upperS+1)
    y=double((uint32(upperS)+z-1)*uint32(Record_Length_S)+(uint32(leftS)-1)*
        Number_Of_Bytes_PerPixel_S+1132);
    p=fseek(fdats,y,-1);
    p=fread(fdats, double(int16(rightS-leftS+1)*2), s, 0, 'b');
    ComS(z,1:int16(rightS-leftS+1)*2)=p';
end

clear z y p
```

```
if TDILS=='ASCEND'
    ComS = rot90(ComS',1);
end


clear i

Re =  single(ComS(:,1:2:int16(rightS-leftS+1)*2));
ComS =  single(ComS(:,2:2:int16(rightS-leftS+1)*2));
ComS = Re+i*ComS;
clear Re

imshow(abs(ComS).^2,[1 k1])
```

CoorToPix.m:

```
function Pix= CoorToPix(Coor,flea)
p= fseek(flea,12507305,-1);
for i=1:25
j = fread(flea, 20,'20*uint8=>char');
a(i)=sscanf(char(j),'%f');
end
for i=1:25
j = fread(flea, 20,'20*uint8=>char');
b(i)=sscanf(char(j),'%f');
end
j = fread(flea, 20,'20*uint8=>char');
P0= sscanf(char(j),'%f');
j = fread(flea, 20,'20*uint8=>char');
L0= sscanf(char(j),'%f');

for i=1:size(Coor,1)
P=Coor(i,1)-P0;
L=Coor(i,2)-L0;
Pix(i,2)=a(1)*L^4*P^4+a(2)*L^3*P^4+a(3)*L^2*P^4+a(4)*L*P^4+a(5)*P^4+a(6)*L^4*P
    ^3+a(7)*L^3*P^3+a(8)*L^2*P^3+a(9)*L*P^3+a(10)*P^3+a(11)*L^4*P^2+a(12)*L^3*P
    ^2+a(13)*L^2*P^2+a(14)*L*P^2+a(15)*P^2+a(16)*L^4*P+a(17)*L^3*P+a(18)*L^2*P+a
    (19)*L*P+a(20)*P+a(21)*L^4+a(22)*L^3+a(23)*L^2+a(24)*L+a(25);
Pix(i,1)=b(1)*L^4*P^4+b(2)*L^3*P^4+b(3)*L^2*P^4+b(4)*L*P^4+b(5)*P^4+b(6)*L^4*P
    ^3+b(7)*L^3*P^3+b(8)*L^2*P^3+b(9)*L*P^3+b(10)*P^3+b(11)*L^4*P^2+b(12)*L^3*P
    ^2+b(13)*L^2*P^2+b(14)*L*P^2+b(15)*P^2+b(16)*L^4*P+b(17)*L^3*P+b(18)*L^2*P+b
    (19)*L*P+b(20)*P+b(21)*L^4+b(22)*L^3+b(23)*L^2+b(24)*L+b(25);
end
```

PixToCoor.m:

```
function Coor= PixToCoor(Pix,flea)
p= fseek(flea,12506265,-1);
for i=1:25
j = fread(flea, 20,'20*uint8=>char');
a(i)=sscanf(char(j),'%f');
end
for i=1:25
j = fread(flea, 20,'20*uint8=>char');
b(i)=sscanf(char(j),'%f');
end
j = fread(flea, 20,'20*uint8=>char');
P0= sscanf(char(j),'%f');
j = fread(flea, 20,'20*uint8=>char');
L0= sscanf(char(j),'%f');

for i=1:size(Pix,1)
    P=Pix(i,2)-P0;
    L=Pix(i,1)-L0;
    Coor(i,1)=a(1)*L^4*P^4+a(2)*L^3*P^4+a(3)*L^2*P^4+a(4)*L*P^4+a(5)*P^4+a(6)*L
        ^4*P^3+a(7)*L^3*P^3+a(8)*L^2*P^3+a(9)*L*P^3+a(10)*P^3+a(11)*L^4*P^2+a
        (12)*L^3*P^2+a(13)*L^2*P^2+a(14)*L*P^2+a(15)*P^2+a(16)*L^4*P+a(17)*L^3*P
        +a(18)*L^2*P+a(19)*L*P+a(20)*P+a(21)*L^4+a(22)*L^3+a(23)*L^2+a(24)*L+a
        (25);
```

```
    Coor(i,2)=b(1)*L^4*P^4+b(2)*L^3*P^4+b(3)*L^2*P^4+b(4)*L*P^4+b(5)*P^4+b(6)*L
        ^4*P^3+b(7)*L^3*P^3+b(8)*L^2*P^3+b(9)*L*P^3+b(10)*P^3+b(11)*L^4*P^2+b
        (12)*L^3*P^2+b(13)*L^2*P^2+b(14)*L*P^2+b(15)*P^2+b(16)*L^4*P+b(17)*L^3*P
        +b(18)*L^2*P+b(19)*L*P+b(20)*P+b(21)*L^4+b(22)*L^3+b(23)*L^2+b(24)*L+b
        (25);
end
```

PreRegis.m:

```
sizeM=size(ComM);

ComM4=medfilt2(abs(ComM),[5 5]);
ComS4=medfilt2(abs(ComS),[5 5]);

clear RegisterS RegisterM Registered mask


y=double(uint16(sizeM/40));

RR1=[2*y(1):uint16(y(1)/2):(sizeM(1)-2*y(1))]';
RR2=[2*y(2):uint16(y(2)/2):(sizeM(2)-2*y(2))]';

range=double(uint16(min(2*y)/10*9));
mask=double(uint16(range/2));

for xxx = 1:size(RR1,1)
for zzz = 1:size(RR2,1)
    RegisterM((xxx-1)*size(RR2,1)+zzz,:)=[RR1(xxx),RR2(zzz)];
    Registering=RegisterM((xxx-1)*size(RR2,1)+zzz,:);
    Register2=abs(ComM4((Registering(1)-range):(Registering(1)+range),(
        Registering(2)-range):(Registering(2)+range)));
    Register2=interpft(Register2,20*size(Register2,1),1);
    Register2=interpft(Register2,20*size(Register2,2),2);
    Register1=abs(ComS4((Registering(1)-mask):(Registering(1)+mask),(Registering
        (2)-mask):(Registering(2)+mask)));
    Register1=interpft(Register1,20*size(Register1,1),1);
    Register1=interpft(Register1,20*size(Register1,2),2);
    filtered=normxcorr2(Register1,Register2);
    [l1 l2]=find(filtered==max(filtered(:)));
    Shift((xxx-1)*size(RR2,1)+zzz,:)=([l1 l2]-size(Register1)-(size(Register2)-
        size(Register1))/2)/20;
    Corr_Coef((xxx-1)*size(RR2,1)+zzz,:)=max(filtered(:));
    clc
    pp0=sprintf('progressing...%3.2f%% finished',100/(size(RR1,1)*size(RR2,1))
        *((xxx-1)*size(RR2,1)+zzz));
    disp(pp0)
end
end

clear Register2 Register1 filtered l1 l2

mean_shift=mean(Shift);

if mean_shift(1)<5
    mean_shift(1)=5;
end
if mean_shift(2)<1
    mean_shift(2)=1;
end

error_shift=find(abs(Shift(:,1))>abs(3*mean_shift(1)));
Shift(error_shift,:)=[];
RegisterM(error_shift,:)=[];
error_shift=find(abs(Shift(:,2))>abs(4*mean_shift(2)));
Shift(error_shift,:)=[];
RegisterM(error_shift,:)=[];

RegisterM=double(RegisterM);
RegisterS=RegisterM-Shift;
```

```
RegisterS=rot90(RegisterS,1)';
RegisterM=rot90(RegisterM,1)';


ComCom=cp2tform(RegisterS,RegisterM,'Polynomial',4)
ComS=imtransform(ComS, ComCom,'bicubic','FillValues',0,'Xdata',[1 sizeM(2)],'
    YData',[1,sizeM(1)]);
sizeM=size(ComM);
ComS=ComS(1:sizeM(1),1:sizeM(2));
figure,imshow(abs(ComS).^2,[1 k1])
```

ObiRegis.m:

```
RegisterM(:,1)=RegisterM(:,1)-upperM+1;
RegisterM(:,2)=RegisterM(:,2)-leftM+1;

RegisterS(:,1)=RegisterS(:,1)-upperS+1;
RegisterS(:,2)=RegisterS(:,2)-leftS+1;

RegisterS=rot90(RegisterS,1)';
RegisterM=rot90(RegisterM,1)';

sizeM=size(ComM);

ComCom=cp2tform(RegisterS,RegisterM,'Projective')
ComS=imtransform(ComS, ComCom,'bicubic','FillValues',0,'Xdata',[1 sizeM(2)],'
    YData',[1,sizeM(1)]);

ComS=ComS(1:sizeM(1),1:sizeM(2));
figure,imshow(abs(ComS).^2,[1 k1])
```

FineRegist.m:

```
sizeM=size(ComM);
Com3=Com3(1:sizeM(1),1:sizeM(2));

clear Registering RegisterM Registered mask


y=20;

RR1=[2*y:y:(sizeM(1)-y)]';
RR2=[2*y:y:(sizeM(2)-y)]';

range=15;
mask=10;

for xxx = 1:size(RR1,1)
for zzz = 1:size(RR2,1)
    RegisterM((xxx-1)*size(RR2,1)+zzz,:)=[RR1(xxx),RR2(zzz)];
    Registering=RegisterM((xxx-1)*size(RR2,1)+zzz,:);
    Register2=abs(ComM((Registering(1)-range):(Registering(1)+range),(
        Registering(2)-range):(Registering(2)+range)));
    Register1=abs(Com3((Registering(1)-mask):(Registering(1)+mask),(Registering
        (2)-mask):(Registering(2)+mask)));
    filtered=normxcorr2(Register1,Register2);
    [l1 l2]=find(filtered==max(filtered(:)));
    Registered((xxx-1)*size(RR2,1)+zzz,:)=[l1 l2]-size(Register1)-range+mask+
        RegisterM((xxx-1)*size(RR2,1)+zzz,:);
    Corr_Coef((xxx-1)*size(RR2,1)+zzz,:)=max(filtered(:));
end
end

clear Register2 Register1 filtered l1 l2
```

Interferogram.m:

```
Interf=ComM.*conj(ComS);
Interf2=imfilter(Interf,ones(25));
P=atan2(imag(Interf2),real(Interf2));
figure,imshow(P,[])
```

Appendix G:

Python code for interative calibration of baseline

iterationSingapore.py:

```
# -*- coding: utf-8 -*-


from gamma import *
import os

rawimagedir="/home/christop/data2/Palsar"
slcdir = "/home/christop/data2/Singapore/slc"
interfdir = "/home/christop/data2/Singapore/interf"
demdir = "/home/christop/data2/Singapore/dem"
diffdir = "/home/christop/data2/Singapore/diff"
srtmdir = "/home/christop/data/SRTM"

iterinterfdir= "/mnt/raid2/tiangang/Singapore/interf6"
                                #Comment when working on same directory
iterationdir= "/mnt/raid2/tiangang/Singapore/interf6"      #To Be modified

cmd="mkdir -p "+iterationdir
os.system(cmd)

#images =[("20090220","A0903776-001","ALPSRP163897150","HH"),
          #("20090708","A0903776-004","ALPSRP184027150","HH"),
          #("20091008","A0903776-007","ALPSRP197447150","HH")
         #]

#images =[("20090220","A0903776-002","ALPSRP163897160","HH"),
          #("20090708","A0903776-005","ALPSRP184027160","HH"),
          #("20091008","A0903776-008","ALPSRP197447160","HH")
         #]

images =[("20081226","486_0010_20081226_FBS_11","ALPSRP155730010"),
("20061221","486_0010_20061221_FBS_11","ALPSRP048370010"),
("20070923","486_0010_20070923_FBD_11","ALPSRP088630010"),
("20090928","486_0010_20090928_FBD_11","ALPSRP195990010"),
("20090210","486_0010_20090210_FBS_11","ALPSRP162440010"),
("20070623","486_0010_20070623_FBD_11","ALPSRP075210010"),
("20081110","486_0010_20081110_FBS_11","ALPSRP149020010"),
```

```
("20090628","486_0010_20090628_FBD_11","ALPSRP182570010")
]


g = Gamma(rawimagedir=rawimagedir, slcdir=slcdir, interfdir=interfdir, demdir=
    demdir, diffdir=diffdir, srtmdir=srtmdir, iterinterfdir=iterinterfdir)

#g.generateDEM("20090220")

NumMaster=len(images)
NumSlave=NumMaster-1
NumIter=15

#for image in images:
#  print "*****************************************************"
#  print "******* SLC/MLI/DEM generation: "+image[0]
#  print "*****************************************************"
#  g.generateSLC(image)
#  g.generateOversampled(image[0])
#  g.generateMLI(image[0])
#  g.generateDEM(image[0])



interfList = []
for imageMaster in images:
  for imageSlave in images:
    if (imageMaster[0] != imageSlave[0]):
      interfList.append((imageMaster[0], imageSlave[0]))

#interfList = [("20090708","20090220"),("20090708","20091008")
    ,("20090220","20091008")]
#interfList = [(images[0][0],images[1][0]),(images[0][0],images[2][0]),(images
    [1][0],images[0][0]),(images[1][0],images[2][0]),(images[2][0],images
    [0][0],(images[2][0],images[1][0]))]
#iterationList = [(images[0][0],images[1][0],images[2][0]),(images[1][0],images
    [0][0],images[2][0]),(images[2][0],images[0][0],images[1][0])]


#for master, slave in interfList:
#  g.generateInterferogram(master, slave)
  #generate all interferogram and baseline file


tal_ave_res_base_orb=[[[] for ni in range(NumMaster)] for mi in range(NumIter)]
tal_ave_res_bdot_orb=[[[] for ni in range(NumMaster)] for mi in range(NumIter)]

tal_var_res_base_orb=[[[] for ni in range(NumMaster)] for mi in range(NumIter)]
tal_var_res_bdot_orb=[[[] for ni in range(NumMaster)] for mi in range(NumIter)]

global_ave=[0]*NumIter
global_var=[0]*NumIter

global_rate_ave=[0]*NumIter
global_rate_var=[0]*NumIter

IterationFile=iterationdir+'/'+'iteration.txt'
IterationFileRate=iterationdir+'/'+'iterationRate.txt'
IterationPosFile=iterationdir+'/'+'Position.txt'

iter=open(IterationFile,'w')
iterRate=open(IterationFileRate,'w')
iterPos=open(IterationPosFile,'w')

for x in range(0,NumMaster):

  iter.write("%(S0)16s %(S1)16s " % {"S0": 'M'+images[x][0]+'Ave',"S1": 'Var'})
```

```
  iterRate.write("%(S0)16s %(S1)16s " % {"S0": 'M'+images[x][0]+'Ave',"S1": 'Var
      '})
  iterPos.write("%(S0)16s %(S1)16s %(S2)16s" % {"S0": 'M'+images[x][0]+' T',"S1
      ": ' C',"S2": ' N'})

iter.write("%(S6)16s %(S7)16s %(S8)16s \n\n" % {"S6": 'TotalAbsAve',"S7":'
    TotalVar',"S8":'TotalMovement'})
iterRate.write("%(S6)16s %(S7)16s %(S8)16s \n\n" % {"S6": 'TotalAbsAve',"S7":'
    TotalVar',"S8":'TotalMovement'})
iterPos.write("\n\n")

#Initilize  Position

masterPos=images[0][0]
slaveListPos=[]
for imageSlave in images:
    if (masterPos != imageSlave[0]):
        slaveListPos.append(imageSlave[0])


Pos=[[[[] for xi in range(3)] for ni in range(NumMaster)] for mi in range(
    NumIter+1)]
Pos[0][0]=[5000,5000,5000]

iterPos.write("%(#1)16.7f %(#2)16.7f %(#3)16.7f " % {"#1": Pos[0][0][0],"#2":Pos
    [0][0][1],"#3":Pos[0][0][2]})

for j in range(0, NumSlave):
    (base_orbt,base_orbc,base_orbn,bdot_orbt,bdot_orbc,bdot_orbn)=g.readBaseline
        (masterPos,slaveListPos[j],".rough.base")
    Pos[0][j+1][0]=5000+base_orbt
    Pos[0][j+1][1]=5000+base_orbc
    Pos[0][j+1][2]=5000+base_orbn

    iterPos.write("%(#1)16.7f %(#2)16.7f %(#3)16.7f " % {"#1": Pos[0][j
        +1][0],"#2":Pos[0][j+1][1],"#3":Pos[0][j+1][2]})


iterPos.write("\n")
```

```
#initilize the Folder !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#Becareful !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

for i in range(0,NumMaster):
    master=images[i][0]
    slaveList=[]
    for imageSlave in images:
        if (master != imageSlave[0]):
            slaveList.append(imageSlave[0])
    for j in range(0, NumSlave):
        g.IterInitFolder(master,slaveList[j])


interfdir=iterinterfdir   #MUST Becareful
g = Gamma(rawimagedir=rawimagedir, slcdir=slcdir, interfdir=interfdir, demdir=
    demdir, diffdir=diffdir, srtmdir=srtmdir, iterinterfdir=iterinterfdir)
```

```
#initialize the baseline file

for i in range(0,NumMaster):
    master=images[i][0]
    slaveList=[]
    for imageSlave in images:
        if (master != imageSlave[0]):
            slaveList.append(imageSlave[0])
    for j in range(0, NumSlave):
        g.IterInitBaseline(master,slaveList[j])


#Interation Starts

for n in range(0,NumIter):

  factor=1/(float(n)+1)

  for i in range(0,NumMaster):
    master=images[i][0]
    slaveList=[]
    for imageSlave in images:
        if (master != imageSlave[0]):
            slaveList.append(imageSlave[0])

    print "************processing:\nMasterImage:",master,"\nSlaveImages:",
        slaveList

    tal_res_base_orbt=0
    tal_res_base_orbc=0
    tal_res_base_orbn=0
    tal_res_bdot_orbt=0
    tal_res_bdot_orbc=0
    tal_res_bdot_orbn=0

    res_base_orbt=[0]*NumSlave
    res_base_orbc=[0]*NumSlave
    res_base_orbn=[0]*NumSlave
    res_bdot_orbt=[0]*NumSlave
    res_bdot_orbc=[0]*NumSlave
    res_bdot_orbn=[0]*NumSlave

    for j in range(0, NumSlave):


        (base_orbt,base_orbc,base_orbn,bdot_orbt,bdot_orbc,bdot_orbn)=g.
            readBaseline(master,slaveList[j],".iter.base")
        print base_orbt,base_orbc,base_orbn,bdot_orbt,bdot_orbc,bdot_orbn



        g.writeBaseline(slaveList[j],master,".iter.base",-base_orbt,-base_orbc,-
            base_orbn,-bdot_orbt,-bdot_orbc,-bdot_orbn) #make the reverse
            interferogram coherent
        g.IterRefineBaseline(master,slaveList[j])
        (res_base_orbt[j],res_base_orbc[j],res_base_orbn[j],res_bdot_orbt[j],
            res_bdot_orbc[j],res_bdot_orbn[j])=g.readBaseline(master,slaveList[j
            ],".iter.base_res")
```

```
        print res_base_orbt[j],res_base_orbc[j],res_base_orbn[j],res_bdot_orbt[j
            ],res_bdot_orbc[j],res_bdot_orbn[j]

        tal_res_base_orbt=tal_res_base_orbt+res_base_orbt[j]
        tal_res_base_orbc=tal_res_base_orbc+res_base_orbc[j]
        tal_res_base_orbn=tal_res_base_orbn+res_base_orbn[j]
        tal_res_bdot_orbt=tal_res_bdot_orbt+res_bdot_orbt[j]
        tal_res_bdot_orbc=tal_res_bdot_orbc+res_bdot_orbc[j]
        tal_res_bdot_orbn=tal_res_bdot_orbn+res_bdot_orbn[j]

    ave_res_base_orbt=tal_res_base_orbt/NumSlave
    ave_res_base_orbc=tal_res_base_orbc/NumSlave
    ave_res_base_orbn=tal_res_base_orbn/NumSlave
    ave_res_bdot_orbt=tal_res_bdot_orbt/NumSlave
    ave_res_bdot_orbc=tal_res_bdot_orbc/NumSlave
    ave_res_bdot_orbn=tal_res_bdot_orbn/NumSlave



    Pos[n+1][i][0]=Pos[n][i][0]+factor*ave_res_base_orbt
    Pos[n+1][i][1]=Pos[n][i][1]+factor*ave_res_base_orbc
    Pos[n+1][i][2]=Pos[n][i][2]+factor*ave_res_base_orbn


    tal_ave_res_base_orb[n][i]=(ave_res_base_orbt**2+ave_res_base_orbc**2+
        ave_res_base_orbn**2)**0.5
    tal_ave_res_bdot_orb[n][i]=(ave_res_bdot_orbt**2+ave_res_bdot_orbc**2+
        ave_res_bdot_orbn**2)**0.5

    var_res_base_orbt=0
    var_res_base_orbc=0
    var_res_base_orbn=0
    var_res_bdot_orbt=0
    var_res_bdot_orbc=0
    var_res_bdot_orbn=0


    for j in range(0, NumSlave):
        (base_orbt,base_orbc,base_orbn,bdot_orbt,bdot_orbc,bdot_orbn)=g.
            readBaseline(master,slaveList[j],".iter.base")


        base_orbt=base_orbt+factor*ave_res_base_orbt
        base_orbc=base_orbc+factor*ave_res_base_orbc
        base_orbn=base_orbn+factor*ave_res_base_orbn
        bdot_orbt=bdot_orbt+factor*ave_res_bdot_orbt
        bdot_orbc=bdot_orbc+factor*ave_res_bdot_orbc
        bdot_orbn=bdot_orbn+factor*ave_res_bdot_orbn

        g.writeBaseline(master,slaveList[j],".iter.base",base_orbt,base_orbc,
            base_orbn,bdot_orbt,bdot_orbc,bdot_orbn)
        g.writeBaseline(slaveList[j],master,".iter.base",-base_orbt,-base_orbc,-
            base_orbn,-bdot_orbt,-bdot_orbc,-bdot_orbn)

        var_res_base_orbt=var_res_base_orbt+((res_base_orbt[j]-ave_res_base_orbt
            )**2)/NumSlave
        var_res_base_orbc=var_res_base_orbc+((res_base_orbc[j]-ave_res_base_orbc
            )**2)/NumSlave
        var_res_base_orbn=var_res_base_orbn+((res_base_orbn[j]-ave_res_base_orbn
            )**2)/NumSlave
        var_res_bdot_orbt=var_res_bdot_orbt+((res_bdot_orbt[j]-ave_res_bdot_orbt
            )**2)/NumSlave
        var_res_bdot_orbc=var_res_bdot_orbc+((res_bdot_orbc[j]-ave_res_bdot_orbc
            )**2)/NumSlave
        var_res_bdot_orbn=var_res_bdot_orbn+((res_bdot_orbn[j]-ave_res_bdot_orbn
            )**2)/NumSlave

    tal_var_res_base_orb[n][i]=var_res_base_orbt+var_res_base_orbc+
        var_res_base_orbn
```

```
    tal_var_res_bdot_orb[n][i]=var_res_bdot_orbt+var_res_bdot_orbc+
        var_res_bdot_orbn

  global_ave[n]=0
  global_var[n]=0
  global_rate_ave[n]=0
  global_rate_var[n]=0

  for i in range(0,NumMaster):
      global_ave[n]=global_ave[n]+abs(tal_ave_res_base_orb[n][i])
      global_var[n]=global_var[n]+tal_var_res_base_orb[n][i]
      global_rate_ave[n]=global_rate_ave[n]+abs(tal_ave_res_bdot_orb[n][i])
      global_rate_var[n]=global_rate_var[n]+tal_var_res_bdot_orb[n][i]


  for x in range(0,NumMaster):

    iter.write("%(#0)16.7f %(#1)16.7f " % {"#0": tal_ave_res_base_orb[n][x
        ],"#1":tal_var_res_base_orb[n][x]})
    iterRate.write("%(#0)16.7f %(#1)16.7f " % {"#0": tal_ave_res_bdot_orb[n
        ][0],"#1":tal_var_res_bdot_orb[n][0]})
    iterPos.write("%(#1)16.7f %(#2)16.7f %(#3)16.7f " % {"#1": Pos[n+1][x
        ][0],"#2":Pos[n+1][x][1],"#3":Pos[n+1][x][2]})



  iterPos.write("\n")
  iter.write("%(#6)16.7f %(#7)16.7f %(#8)16.7f \n" % {"#6": global_ave[n],"#7":
      global_var[n],"#8":global_ave[n]*factor})
  iterRate.write("%(#6)16.7f %(#7)16.7f %(#8)16.7f \n" % {"#6": global_rate_ave[
      n],"#7":global_rate_var[n],"#8":global_rate_ave[n]*factor})

iter.close()
iterRate.close()
iterPos.close()
```

References

[1] T Farr, "Chapter 5. radar interactions with geologic surfaces," *Guide to Magellan Image Interpretation*.

[2] A. Rosenqvist, M. Shimada, N. Ito, and M. Watanabe, "ALOS PALSAR: A pathfinder mission for global-scale monitoring of the environment," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, pp. 3307–3316, 2007.

[3] "ALOS PALSAR level 1 product format description (vol.2: Level 1.1/1.5) rev.l," 2009.

[4] "PAGER M 7.0 Haiti region," *United States Geological Survey, 12 January 2010*.

[5] "DLR-Nachrichten," *Lufthansa News*, vol. 86, pp. 42, 1997.

[6] D. Bercovici, "The generation of plate tectonics from mantle convection," *Earth and Planetary Science Letters*, vol. 205, pp. 107–121, 2003.

[7] E. Christophe, A. S. Chia, T. Yin, and L. K. Kwoh, "2009 earthquakes in sumatra: The use of L-BAND interferometry in a SAR-HOSTILE environment," *Geoscience and Remote Sensing Symposium, IGARSS. IEEE International*, pp. 1202 – 1205, 2010.

[8] T. Yin, E. Christophe, S. C. Liew, and S. H. Ong, "Iterative calibration of relative platform position: A new method for SAR baseline estimation," *Geoscience and Remote Sensing Symposium, IGARSS. IEEE International*, p. 4470, 2010.

[9] C. Gauchet, E. Christophe, A. S. Chia, T. Yin, and S. C. Liew, "(accepted) InSAR monitoring of the LUSI mud volcano, East Java, from 2006 to 2010," *Geoscience and Remote Sensing Symposium, IGARSS. IEEE International*, 2011.

[10] E. Christophe, C. M. Wong, and S. C. Liew, "Mangrove detection from high resolution optical data," *Geoscience and Remote Sensing Symposium, IGARSS. IEEE International*, 2010.

[11] G. G. J. Ernst, M. Kervyn, and R. M. Teeuw, "Advances in the remote sensing of volcanic activity and hazards, with special consideration to applications in developing countries," *International Journal of Remote Sensing*, vol. 29, pp. 6687–6723, 2008.

[12] J. C. Curlander, "Location of spaceborne SAR imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. GE-20, no. 3, 1982.

[13] C. Werner, U. Wegmuller, T. Strozzi, A. Wiesmann, and M. Santoro, "PALSAR multi-mode interferometric processing," *The First Joint PI symposium of ALOS Data Nodes for ALOS Science Program*, 2007.

[14] U. Wegmuller and C. Werner, "GAMMA SAR processor and interferometry software," *Third ERS Symposium on Space at the service of our Environment*, p. 1687, 1997.

[15] R. F. Hanssen, T. M. Weckwerth, H. A. Zebker, and R. Klees, "High-resolution water vapor mapping from interferometric radar measurements," *Science*, vol. 283, pp. 1297, 1999.

[16] R. M. Goldstein and C. L. Werner, "Radar interferogram filtering for geophysical applications," *Geophysical Research Letters*, vol. 25, pp. 4035–4038, Dec. 1998.

[17] M. Costantini, "A novel phase unwrapping method based on network programming," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 36, pp. 813–821, Mar. 1998.

[18] M. Costantini and P. A. Rosen, "A generalized phase unwrapping approach for sparse data," *Geoscience and Remote Sensing Symposium, IGARSS. IEEE International*, vol. 28, pp. 267–269, 1999.

[19] S. Shi, "Dem generation using ers-1/2 interferometric sar data," *Geoscience and Remote Sensing Symposium, Proceedings, IEEE International*, vol. 2, pp. 788–790, 2000.

[20] Y. Gorokhovich and A. Voustianiouk, "Accuracy assessment of the processed SRTM-based elevation data by cgiar using eld data from usa and thailand and its relation to the terrain characteristics," *Remote Sensing of Environment*, vol. 104, pp. 409415, 2006.

[21] D. Kosmann, B. Wessel, and V. Schwieger, "Global digital elevation model from TANDEM-X and the calibration/validation with worldwide kinematic gps-tracks," *FIG Congress 2010, Facing the Challenges : Building the Capacity*, 2010.

[22] R. J. Davies, M. Brumm, M. Manga, R. Rubiandini, R. Swarbrick, and M. Tingay, "The East Java mud volcano (2006 to present): an earthquake or drilling trigger?," *Earth and Planetary Science Letters*, vol. 272, pp. 627638, 2008.

[23] A. Mazzini, A. Nermoen, M. Krotkiewski, Y. Podladchikov, S. Planke, and H. Svensen, "Strike-slip faulting as a trigger mechanism for overpressure release through piercement structures. implications for the LUSI mud volcano, indonesia," *Marine and Petroleum Geology*, vol. 26, pp. 17511765, 2008.

[24] C. Yonezawa, T. Yamanokuchi, N. Tomiyama, and Y. Oguro, "Comparison of atmospheric phase delay on ALOS PALSAR interferogram and cloud distribution pattern on simultaneously observed AVNIR-2 images," *Geoscience and Remote Sensing Symposium, IEEE International*, vol. 3, pp. III–1170, 2009.

[25] A. O. Konca, "Partial rupture of a locked patch of the sumatra megathrust during the 2007 earthquake sequence," *Nature*, vol. 456, pp. 631–635, Dec. 2008.

[26] S. Knedlik and O. Loffeld, "Baseline estimation and prediction referring to the SRTM," *Geoscience and Remote Sensing Symposium, Proceedings, IEEE International*, vol. 1, pp. 161, Nov. 2002.

[27] N. Kudo, S. Nakamura, and R. Nakamura, "The accuracy verification for GPS receiver of ALOS by SLR," *The* 15^{th} *International Workshop on Laser Ranging*, Oct. 2006.

[28] D. T. Sandwell, R. J. Mellors, X. Tong, M. Wei, and P. Wessel, "GMTSAR software for rapid assessment of earthquakes," *American Geophysical Union, Fall Meeting*, 2010.

[29] P. A. Rosen, S. Henley, G. Peltzer, and M. Simons, "Updated repeat orbit interferometry package released," *Eos Trans. AGU.*, vol. 85, no. 5, Nov. 2007.

[30] M. Pritchard, "New insar results from north america from the winsar consortium," *FRINGE 2007: Advances in SAR Interferometry from ENVISAT and ERS missions*, 2007.

[31] R. F. Hanssen, "Radar interferometry-data interpretation and error analysis," 2001.

[32] A. Ferretti, C. Prati, and F. Rocca, "Multibaseline InSAR DEM reconstruction: the wavelet approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, no. 2, pp. 705–715, Mar 1999.

[33] C. Werner, S. Hensley, R. M. Goldstein, P. A. Rosen, and H. A. Zebker, "Techniques and applications of SAR interferometry for ERS-1: Topographic mapping, change detection, and slope measurement," in *Proc. First ERS-1 Symp.*, Nov. 1992, pp. 205–210.

[34] D. P. Belcher and N. C. Rogers, "Theory and simulation of ionospheric effects on synthetic aperture radar," *Radar, Sonar & Navigation, IET*, vol. 3, pp. 541, 2009.

[35] A. Ferretti, C. Prati, and F. Rocca, "Permanent scatterers in SAR interferometry," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, pp. 820, 2001.

[36] A. Ferretti, C. Prati, and F. Rocca, "Nonlinear subsidence rate estimation using permanent scatterers in differential sar interferometry," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 5, 2000.

[37] A. Ferretti, G. Savio, R. Barzaghi, A. Borghi, S. Musazzi, F. Novali, C. Prati, and F. Rocca, "Submillimeter accuracy of insar time series: Experimental validation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, May 2007.

[38] D. Small, C. Werner, and D. Nuesch, "Baseline modelling for ERS-1 SAR interferometry," *Geoscience and Remote Sensing Symposium, IGARSS. IEEE International*, vol. 3, pp. 1204 – 1206, Aug. 1993.

[39] M. Costantini, F. Minati, A. Quagliarini, and G. Schiavon, "SAR interferometric baseline calibration without need of phase unwrapping," *Geoscience and Remote Sensing Symposium, IGARSS. IEEE International*, vol. 1, pp. 493 – 495, 2004.

[40] E. Christophe and J. Inglada, "Open source remote sensing: Increasing the usability of Cutting-Edge algorithms," *IEEE Geoscience and Remote Sensing Newletter*, vol. 150, pp. 9–15, 2009.