Haute Ecole LEONARD de VINCI

Institut Supérieur Industriel

Post-wildfire Vegetation Loss Mapping using Bitemporal Synthetic Aperture Radar

Student:

de Patoul Benoît

Tutor:

Combéfis Sébastien, Ph.D.

ECAM Brussels Engineering School

Promoter:

ZhiQiang Chen, Ph.D.

University of Missouri-Kansas City

This Master's thesis was written with the aim of obtaining the Master's degree in Industrial Engineering (electronics).

Academic Year 2016-2017

Acknowledgment

A huge thank you goes to Dr. ZhiQiang Chen for giving me the opportunity to participate in his research and to be part of his researcher's team. I thank you for his help, his advices, and his patience during all the research. I would like to thank Dr. Combéfis who also helped me with all his advices. Also, I would like to thank my family and my friends for all their support and encouragement, supporting me in good times as in bad times. This has not been an easy task, but with the help of everyone everything is possible.

Summary of the Master's thesis

First name: Benoît

Last name: de Patoul

Master in Industrial Engineering in electronics

Post-wildfire Vegetation Loss Mapping using Bitemporal Synthetic Aperture Radar

Wildfire events followed by heavy precipitation have been proven causally related to breakouts of mudflow or debris flow, which, can demand rapid evacuation and threaten residential communities and civil infrastructure. For example, in the case of the city of Glendora, California, it was afflicted by a severe wildfire and flooding event in its past (1968) and the mudslides and debris flow of 1969 killed 34 people. Therefore, the burn area mapping due to wildfire events is critical to agencies for preparing for secondary hazards, particularly mudflow and flooding prior to the next large rainfall. However, rapid post-wildfire mapping of burned areas (that are usually covered by vegetation; hence also called vegetation loss mapping) is not readily obtained by regular remote sensing methods, e.g. various optical methods, due to the presence of smoke, haze, and rainy/cloudy conditions that often follow a wildfire event. In this paper, we will introduce and develop a methodology framework that uses Synthetic Aperture Radar images that are sensed prior to and after a wildfire event. A supervised machine learning algorithm will be proposed to classify burned and intact areas using the pre- and the post-event NASA UAVSAR SAR data. The 2014 Colby Fire event, which affected the downstream city of Glendora, was used to evaluate the proposed framework. It burned 1952 acres of forest and destroyed five homes

ECAM

Cahier des charges relatif au travail de fin d'études de

De Patoul Benoît, inscrit en 2ème Master, électronique

- Année académique : 2016-2017

- Titre provisoire : Wildfire detection and quantification using UAVSAR data

- Objectifs à atteindre :

  This project aims to develop digital image processing methods including pattern classification and quantification methods for mapping wildfire induced damage. Wildfire events have appeared frequently occurred around world, particularly in the United States. Optical methods are limited due to the smoke and haze amid any wildfire events. Synthetic aperture radar data is an all-weather remote sensing method that can penetrate smokes and hazes, and the UAVSAR platform from NASA can provide temporal imagery data sets before and after a wildfire event; hence the UAVSAR data is primarily used in this project. In this project, I will aim to develop automatic approach to wildfire mapping based on bi-temporal UAVSAR data, and the accuracy of the proposed methods will be evaluated against field data.

- Principales étapes :

  First, past methods will be reviewed, including Principle Component Analysis (PCA) based method. Second, advanced feature extraction and pattern classification methods will be selected and tuned in this project for improving the accuracy of the existing method. The proposed methods will be evaluated using ground-based photo pictures that were captured by NASA engineers. Last, the proposed methods will be implemented using Matlab. The research results and findings will be summarized and published through a journal venue.

Fait en trois exemplaires à Kansas City, le 08/02/2017

L'étudiant	*Le tuteur*	*Le promoteur*
Nom – Prénom : de Patoul Benoît	Nom – Prénom : Dr. Combéfis Sébastien	Nom – Prénom : Dr. ZhiQiang Chen
	Département/Unité : Génie électrique	Société: University of Missouri-Kansas City
Signature :	Signature :	Signature :
		02/23/2017

Abbreviations

SAR = Synthetic Aperture Radar

InSAR = Interferometry Synthetic Aperture Radar

PolSAR = polarimetric Synthetic Aperture Radar

SVM = Support Vector Machine

RVM = Relevance Vector Machine

UAVSAR = Uninhabited Aerial Vehicle Synthetic Aperture Radar

NASA = National Aeronautics and Space Administration

DEM = Digital Elevation Mode

PCA = Principal Component Analysis

RGB = red, green, blue

FV = feature vector

TABLE OF CONTENTS

1 INTRODUCTION

This Master's thesis was written with the aim of obtaining the Master's degree in Industrial Engineering with a specialization in electronics in the Brussels School of Engineering, ECAM (http://www.vinci.be/fr-be/Ecam/Pages/Accueil.aspx). The research lasted four months and was undertaken in the United States of America in the Missouri-Kansas City University, under the supervision of Doctor Zhiqiang Chen as promoter, and Doctor Combéfis Sébastien as tutor.

The field of this research is image processing and classification using multi-class supervised machine learning, all this been processed by the MATLAB program. The data (radar images) were provided by the National Aeronautics and Space Administration (NASA) using the airborne radar "Uninhabited Aerial Vehicle Synthetic Aperture Radar" (UAVSAR, https://uavsar.jpl.nasa.gov/).

Figure 1: UAVSAR, from (NASA, s.d.)

Wildfire events followed by heavy precipitation have been proven causally related to breakouts of mudflow or debris flow, which, can demand rapid evacuation and can threaten residential communities and civil infrastructure. Therefore, the burn area mapping due to wildfire events is critical to agencies for preparing for secondary hazards, particularly mudflow and flooding prior to the next large rainfall. However, rapid post-wildfire mapping of burned areas cannot be obtained by regular remote sensing methods, e.g. various optical methods, due to the presence of smoke, haze, and rainy/cloudy conditions that often follow a wildfire event. This research aims to provide a post-wildfire detection (burned areas) and quantification using bitemporal *Synthetic Aperture Radar* images (one image before the event and the other one, after the event). It uses and compares two kinds of multi-class supervised machine learnings:

- Support Vector Machine (SVM);
- Relevance Vector Machine (RVM).

The document is split in three parts:

- theory;
- research;
- conclusion.

The first part contains the theory about radar images to achieve a better global understanding of this thesis, and different subjects that are used in the research. The covered subjects are:

- Synthetic Aperture Radar (SAR);
- Interferometric Synthetic Aperture Radar (InSAR);
- Polarimetric Synthetic Aperture Radar (PolSAR).
- Principal Component Analysis (PCA)
- Support Vector Machine (SVM)
- Histogram of Oriented Gradients (HOG)

Each concept is explained without excessive details.

The second part contains the research about the detection and quantification of post-wildfire. This part will also be used as a support to write the research paper to be published and, furthermore, presented in the Engineering Mechanics Institute conference (EMI) in United States, San Diego on June 7, 2017.

The third part concludes this document by exploring the results of the research, if all the objectives have been reached, and the different improvements that could be done.

2 THEORY

2.1 SYNTHETIC APERTURE RADAR (SAR)

2.1.1 Introduction

Synthetic Aperture Radar is a microwave imaging with specific characteristics:

- active because it has day and night operational capabilities;
- weather independent because it uses microwaves;
- accurate distance measurements because we can employ interferometric techniques;
- polarization can be exploited;
- fine spatial resolution because it is independent from distance.

The radar principle is simple. A transmitter sends a signal that will be backscattered by an object (see Figure 2). The total time delay of the received echo will give us the distance of the object using the formula:

$$t = \frac{2R}{c}$$

t = time delay

R = distance between the object and the transmitter

c = speed of light

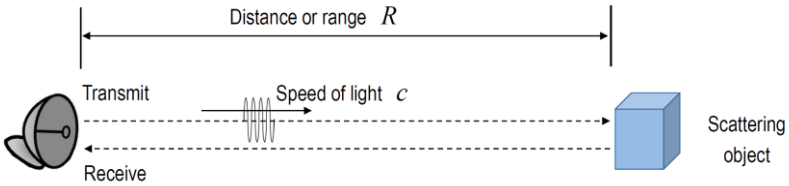


Figure 2: radar principle, from (Lopez-Sanchez, 2014)

SAR uses the radar principle with a side looking geometry (see Figure 3). Side looking geometry allows the radar to distinguish the objects by their range. The microwaves transmitted from the radar must reach the objects (scatterers) on the ground and then come back to obtain a SAR image. Because of the side looking, the slant ranges of each scatterer is different, giving for each object a different delay between transmission and reception.

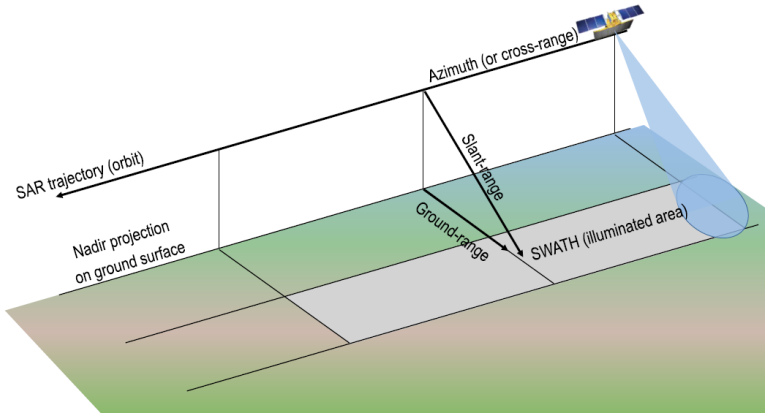


Figure 3: side looking geometry, from (Lopez-Sanchez, 2014)

A digital SAR image can be seen as a two-dimensional array formed by columns and rows, creating a mosaic view forming each pixel. Each pixel is a small area of the Earth called a resolution pixel or resolution cell. A complex number composes each cell, giving an amplitude and a phase information about the microwave backscattered by all the scatterers within the resolution cell. Different rows of the image correspond to different ground range locations, whereas different columns are associated with different azimuth locations (see Figure 3). The location and resolution of the cell depend only on the SAR system characteristics. For example, in the European Remote satellite case, the resolution cell is about 5 meters in azimuth and 9 meters in slant-range.

The total energy backscattered by all the objects within one cell is measured using the backscattering coefficient σ_0. This coefficient will set the color of the pixel. A high coefficient value gives a brighter pixel, whereas a low value gives a darker pixel.

$$\sigma_0 = 10 \log\left(\frac{E_r}{E_{iso}}\right) [dB]$$

E_r = Energy received from the target

E_{iso} = Energy received from an isotropic target

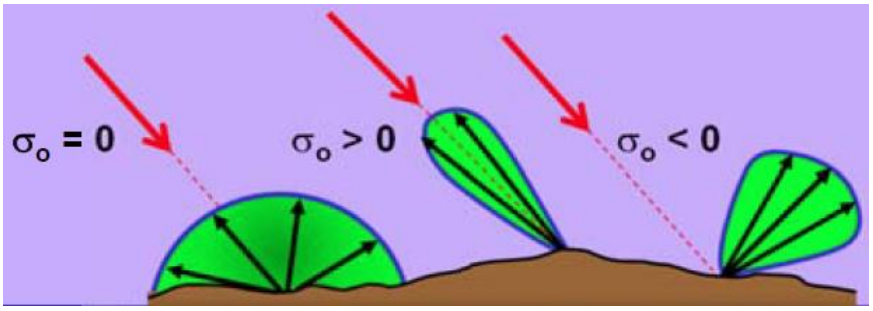


Figure 4: backscattering coefficient, from (Papathanassiou)

Values	Scenes
σ_0 > 0 dB	Man-made objects, urban areas Slopes facing the radar
-10 dB < σ_0 < 0 dB	Very rough surfaces Forests (dense vegetation)
-20 dB < σ_0 < -10 dB	Rough surfaces (sea with wind) Agricultural crops
σ_0 < -20 dB	Calm water Smooth surfaces (roads) Very dry ground (sand)

Figure 5: values of backscattering coefficient, from (Lopez-Sanchez, 2014)

The microwave can penetrate the ground depending on his wavelength, the higher is the wavelength, the higher is the penetration, and vice-versa. That is why different frequency bands are used, depending on the purposes of the measures, and the type of soil (see Figure 6).

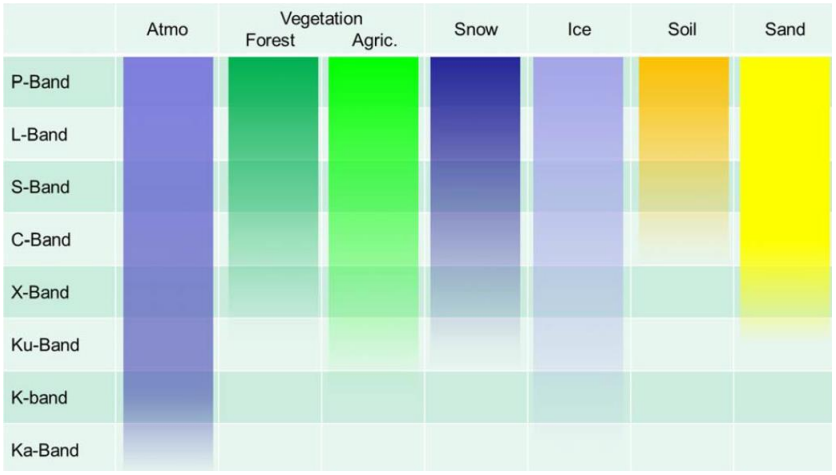


Figure 6: frequency bands, from (Papathanassiou)

2.1.2 Amplitude and phase information

The complex number corresponds to the ratio of the received signal (signal backscattered by the cell) over the signal transmitted (see Figure 8). As said before, each signal backscattered by each pixel, is characterized by a vector representing an amplitude and a phase (see Figure 7).

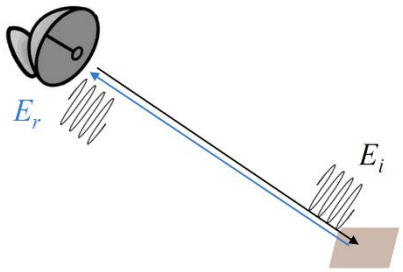


Figure 8: transmitted and received signal, from (Lopez-Sanchez, 2014)

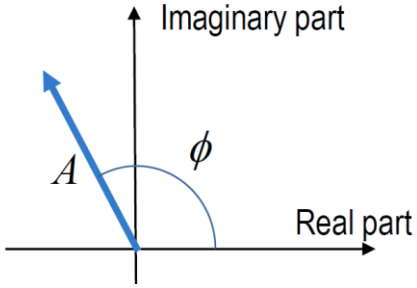


Figure 7: vector, from (Lopez-Sanchez, 2014)

$$\rho = \frac{E_r}{E_i} = a + jb = Ae^{j\phi}$$

ρ = energy ratio

E_r= Energy received from the target

E_i= Energy received from an isotropic target

The amplitude is the amount of signal returned to the radar depending on the roughness of the terrain. Therefore, it relies on the kind of ground the satellite is measuring. For example, the water has a low signal intensity backscattered toward the radar, making the pixel dark. Whereas the urban areas have a strong signal intensity backscattered, making the pixels brighter (see Figure 5). Using these differences, we can infer the kind of ground we are looking at.

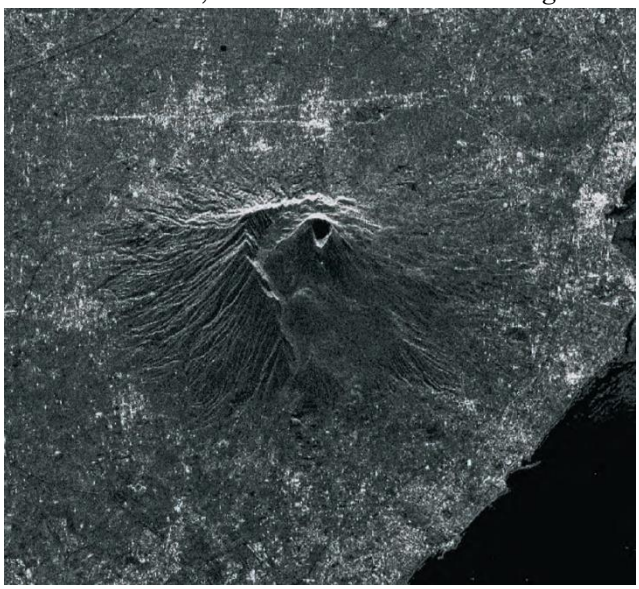


Figure 9: amplitude image, from (Ferretti, Monti-Guarnieri, Prati, & Rocca, February 2007)

On the Figure 9 we can clearly distinguish a mountain (mountain Etna, Italy) in the middle with water on the lower right and some urban areas around. There is a kind of distortion effect on the mountain due to the side looking of the satellite. It allows us to deduce from which side the satellite was looking at. The distortion effect is more detailed later in this document (see 2.1.3).

Because of the almost pure sinusoidal nature of the transmitted waves, the phase information from one SAR image gives us the last fraction of the two-way travel distance. This distance is always smaller than the transmitted wavelength. Due to the huge ratio between the resolution cell dimension and the wavelength, the phase change from one pixel to another looks random (see Figure 10) which has not practical utility. The phase information is used in the InSAR method, which allows highly accurate distance results in remote sensing.

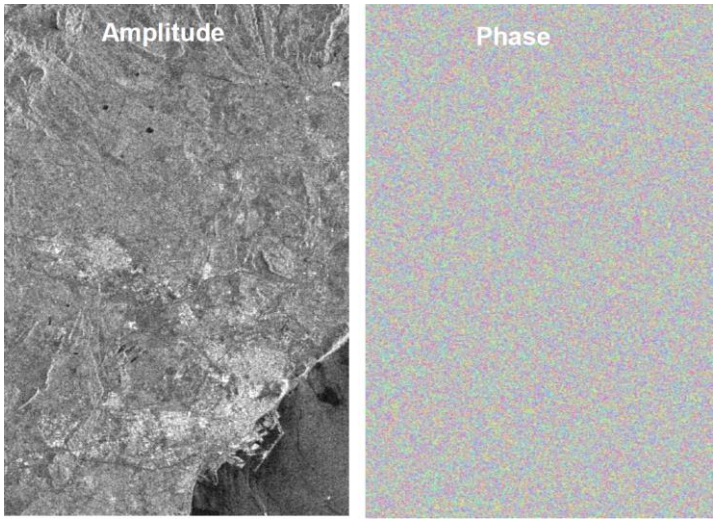


Figure 10: amplitude and phase information, from (Lopez-Sanchez, 2014)

Each pixel or resolution cell contains many different backscattering objects. The vector result of the pixel is the sum of all the individual echoes from each scatterer (vectors) within the cell. The result is different from pixel to pixel due to the random phase because of multiple reflections between scatterers. This produces fluctuations and therefore a granular appearance of the images (see Figure 11). This effect is called speckle. This problem causes a noisy appearance of a homogenous image, harder interpretations, erroneous quantitative estimation on one pixel, etc.

Figure 11: speckle on the image of Linate Airport, from (Ferretti, Monti-Guarnieri, Prati, & Rocca, February 2007)

The solution to this is to take more images of the same area from slightly different looking angles or at different times. The average of the images from the same place reduces the speckle problem (see Figure 12).

Figure 12: average of 60 images from Liante Airport, from (Ferretti, Monti-Guarnieri, Prati, & Rocca, February 2007)

2.1.3 Resolution cell

Each resolution cell is composed by a range resolution, and an azimuth resolution. The duration of the pulses provides the (slant) range resolution, and the azimuth resolution is provided by an angular resolution from a narrow antenna beam.

2.1.3.1 Range resolution

$$\rho_r = \frac{c * \tau}{2}$$

$$\rho_r = \frac{c}{2B}$$

ρ_r = range resolution τ = pulse duration

c = speed of light B = bandwidth

We have two cases, when the resolution is high enough to distinguish two separate responses, and when the resolution is not high enough to distinguish two separate objects. The Figure 13 represents the two cases.

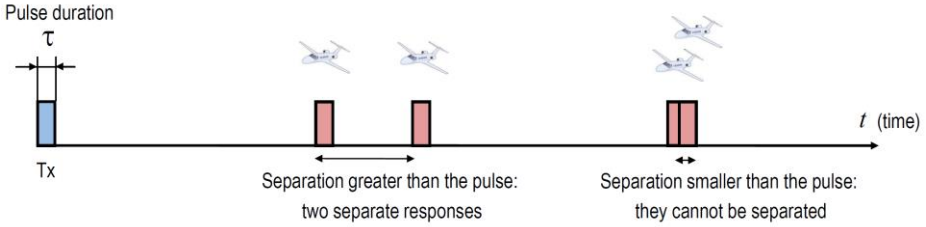


Figure 13: range resolution, from (Lopez-Sanchez, 2014)

To increase the resolution, we should reduce the pulse duration. This procedure causes a major problem when is related to higher resolutions range because of its very short pulses demanding. Very short pulses do not contain enough energy and ask for more bandwidth (so more expensive). The energy of the transmitted pulse is given by: $E = \int_0^\tau P_t * dt$, where P is the transmitted power. Looking at the energy formula, to counteract the energy problem, we could add more power, but the power is limited by the sensor hardware. The solution to this matter is to use pulse compression using chirp.

The chirp is a signal with an increasing or decreasing frequency according to time. It is used for pulse compression transforming a long duration pulse into a narrow pulse with high peak power (see Figure 14). This solution delivers a small pulse duration combined to a high peak of power.

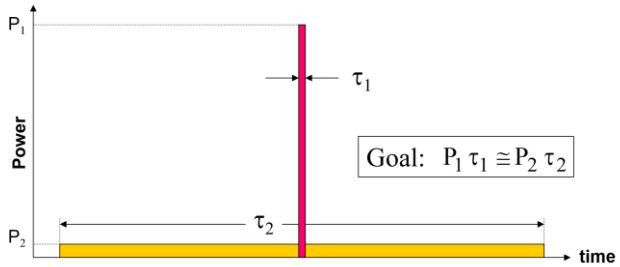


Figure 14: pulse compression

The link between the ground range resolution and the slant range resolution is shown in Figure 15, with θ been the incidence angle, ρ_y the ground range resolution and, ρ_r the slant range resolution. We notice how the width of the ground resolution varies following the swath, compressing or expanding the ground range resolution. This creates a distortion on the SAR images.

$$\rho_y = \frac{\rho_r}{\sin(\theta)}$$

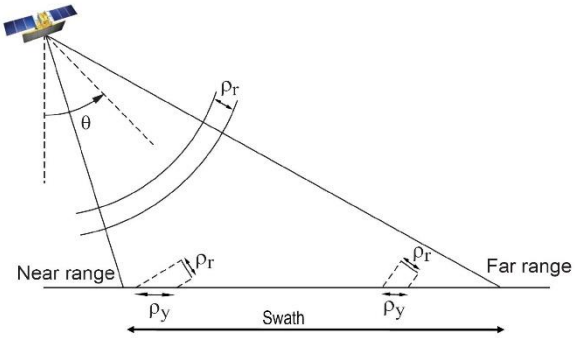


Figure 15: slant range and ground range, from (Lopez-Sanchez, 2014)

We have three cases of geometrical distortions:

1. Foreshortening

In this case, the resolution cell is compressed or expanded following the angle of the slope or the swath. Compressed because the ground resolution width is higher than the slant resolution width and therefore, it must be compressed ($0 < \alpha < \theta$). Expanded because the ground resolution width is lower than slant resolution width and therefore, it must be expanded ($-\theta < \alpha < 0$). On the left of the volcano in the Figure 9 we can see the compressed effect and on the right, the expanded effect.

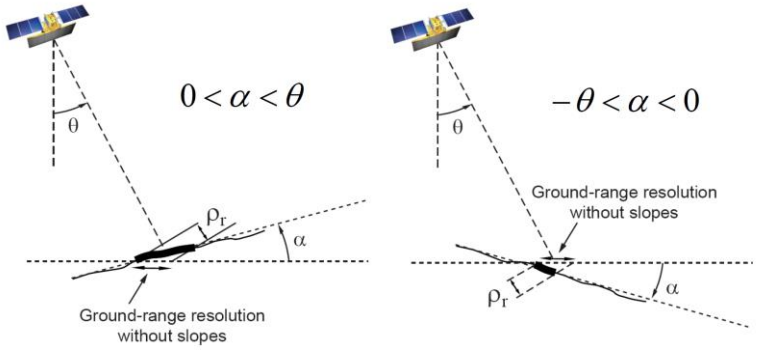


Figure 16: foreshortening, from (Lopez-Sanchez, 2014)

2. Layover

The geometry of the image is inverted when $\alpha \geq \theta$. The Figure 9 shows an example of this effect on the north of the volcano.

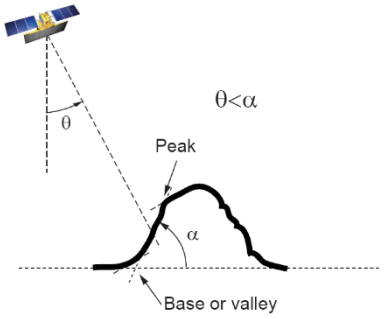


Figure 17: layover, from (Lopez-Sanchez, 2014)

3. Shadow areas

Some regions of the ground are hidden because the radar pulses do not reach them. This is the case when $\alpha \leq \theta - \pi/2$.

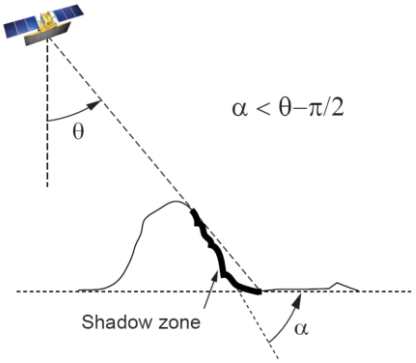


Figure 18: shadow areas, from (Lopez-Sanchez, 2014)

2.1.3.2 Azimuth resolution

This is where the name Synthetic Aperture comes from. An artificial antenna (aperture) is created (synthesized) by processing coherently the echoes received over a long azimuth distance. However, why do we not just use the width of the antenna beam (the real aperture of the radar) like shown in Figure 19? The problem is due to the relationship between the resolution in azimuth ρ_a and the range R:

$$\rho_a = R * \frac{\lambda}{l_a}$$

ρ_a = azimuth resolution

R = range

λ = wavelength

l_a = antenna width

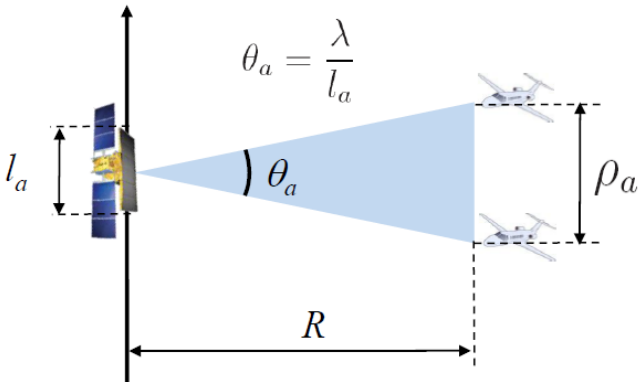


Figure 19: real aperture radar, from (Lopez-Sanchez, 2014)

For example, if we want an azimuth resolution of 6 meters with C band ($\lambda = 5.6cm$) using satellite radar like ERS with an altitude of around $800km$ (R), we would need an antenna size of $9km$. Impossible to do!

Solution? Synthetic Aperture Radar concept. This method uses the flight path of the platform to simulate a large antenna operating in a pulse mode. Each target at different locations backscatter the signal during the platform trajectory. This means that each target will have a different reflectivity value for every position of the satellite. Because the satellite approaches and then, moves away from the target (see Figure 21), to distinguish every target, the method uses the Doppler effect. This means that each point will have a unique Doppler history. In Figure 20 we can see the target and the synthetic aperture created by the movement of the satellite.

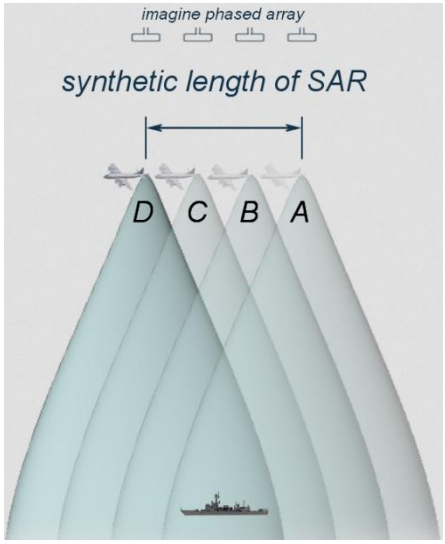


Figure 21: SAR, from (Wolff, n.d.)

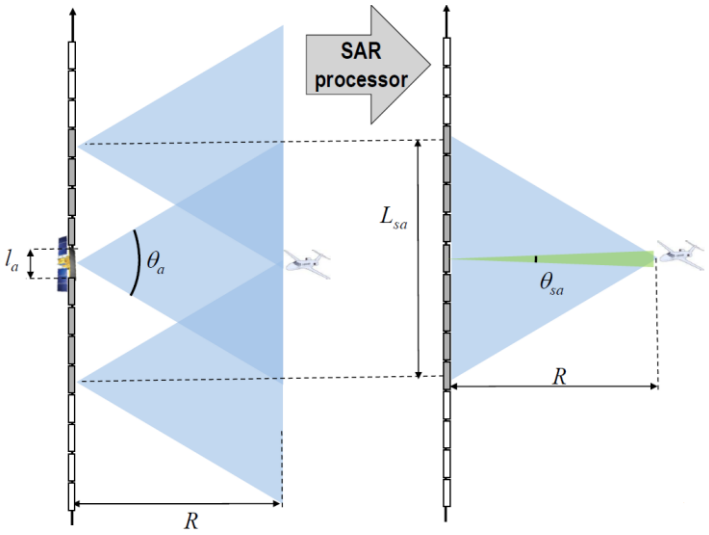


Figure 20: SAR, from (Lopez-Sanchez, 2014)

$$L_{sa} = \theta_a * R \qquad \theta_{sa} = \frac{\lambda}{2L_{sa}}$$

$$L_{sa} = \frac{\lambda}{l_a} * R \qquad \theta_{sa} = \frac{l_a}{2R}$$

$$\rho_a = \frac{l_a}{2}$$

l_a = antenna width

λ = wavelength

L_{sa} = length of the synthetic aperture

ρ_a = azimuth resolution

R = range

The azimuth resolution does not depend anymore on the range of the transmitter, allowing high resolution, even with satellites.

2.2 INTERFEROMETRIC SYNTHETIC APERTURE RADAR (INSAR)

2.2.1 Introduction

In the SAR method, we only use the amplitude information because of the speckle on the phase image (see Figure 10). In order to use the phase information, we need to have two SAR images of the same area. A satellite or a plane can observe the same area from slightly different angles by using simultaneously two radars mounted on the same platform, or by observing at different times (exploiting the repeated orbit of the satellite).

The InSAR method gives us the possibility to create an interferogram, which will lead us to the generation of topographic maps and Digital Elevation Model (DEM) (3D model representation of an area), and the observation of displacement on the Earth's surface (differential InSAR).

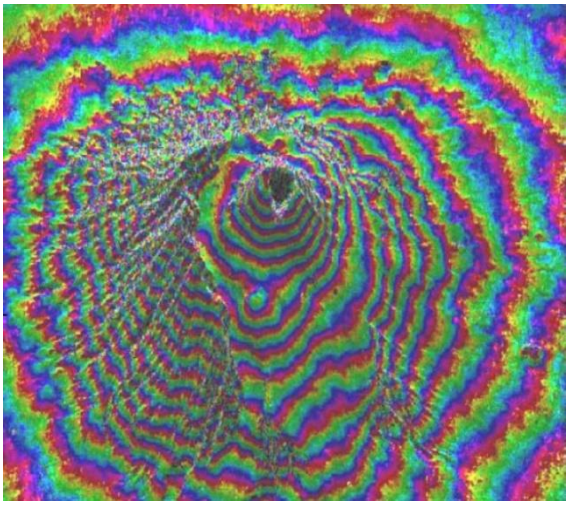


Figure 23: topographic interferogram, from (Lopez-Sanchez, 2014)

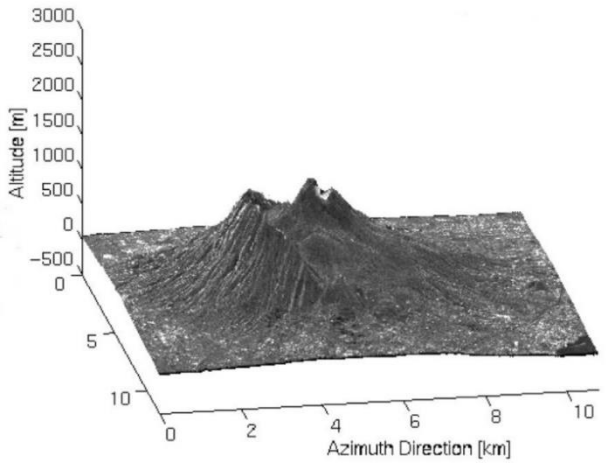


Figure 22: Digital Elevation Modelisation, from (Lopez-Sanchez, 2014)

2.2.2 Interferogram generation

We said before that an amplitude and a phase (complex value) characterize every pixel of a SAR image. The amplitude is the measure of the target reflectivity, whereas the phase encodes changes at the surface. The InSAR method uses the phase information of two SAR images to determine the phase difference between each pair of corresponding pixels (the result is called the interferometric phase). An interferogram is generated by cross-multiplying pixel by pixel of the first SAR image (called master), with the complex conjugate of the second (called slave). Therefore, the interferogram amplitude is a multiplication of the two amplitudes of the two SAR images, and the interferogram phase is the phase difference between the two phases from the two SAR images.

The following figure shows an example of a topographic interferogram generation:

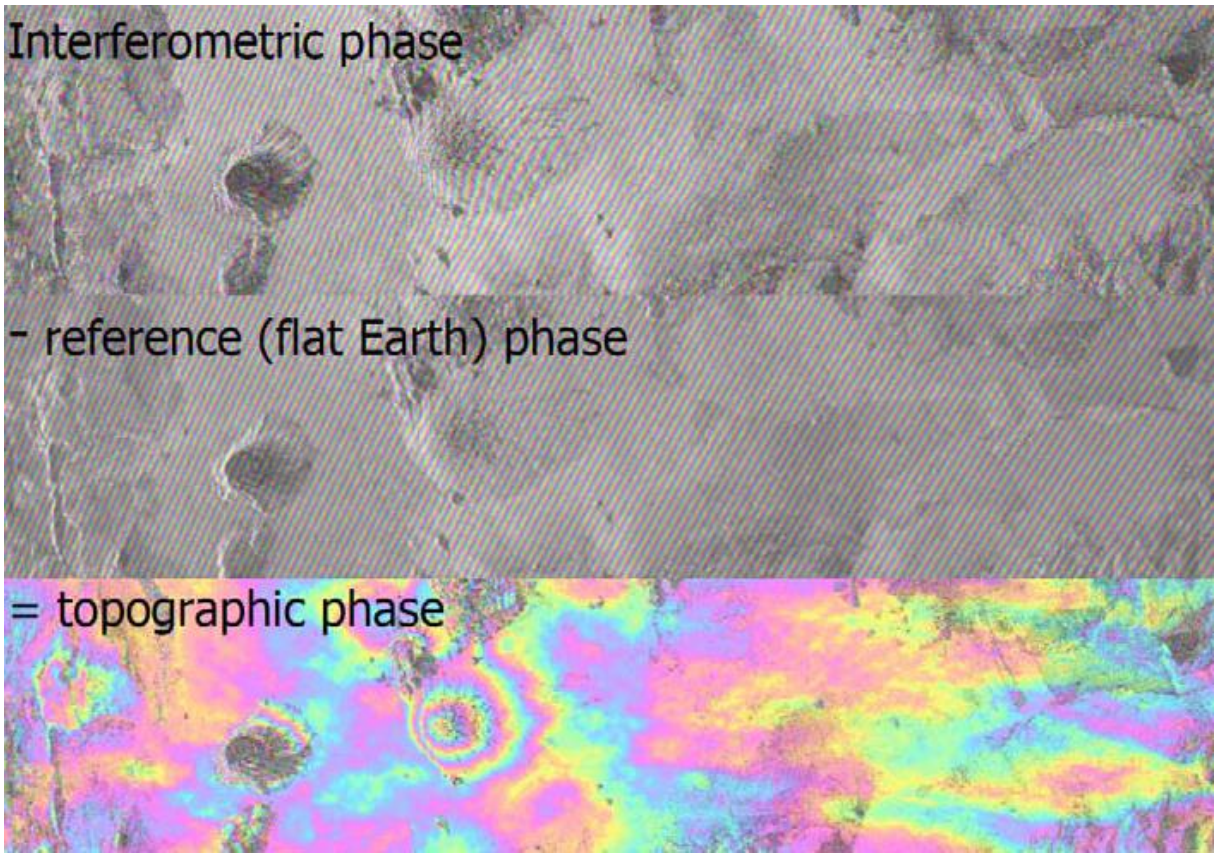


Figure 24: topographic phase, from (Hanssen, 2014)

The first step is to get the interferogram from the two complex SAR images (the interferometric phase). Once we got the interferometric phase, we subtract the phase of the flat Earth interferogram to obtain the topographic interferogram. The flat Earth interferogram represents an Earth without any topography variation; in other words, a perfect circular Earth. This operation is called interferogram flattening and it generates a phase map proportional to the relative terrain altitude.

2.2.3 Interferometric phase

We have one single satellite, which took in two different times (t_1 and t_2) and from slightly different looking angles, the amplitude and phase information of one pixel. Using the information provided by the pixel, let us derive the interferometric phase equation.

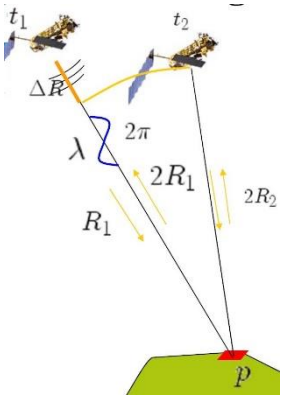


Figure 25: interferometric phase, from (Hanssen, 2014)

$$t_1 \rightarrow y_1 = A_1 e^{j\phi_1} \qquad t_2 \rightarrow y_2 = A_2 e^{j\phi_2}$$

$$with\ \frac{2R_n}{\lambda} = \frac{\phi_n}{2\pi}$$

$$interferometric\ phase \rightarrow\ \phi_{int} = \phi_1 - \phi_2$$

$$\phi_{int} = \frac{4\pi(R_1 - R_2)}{\lambda}$$

$$\phi_{int} = 4\pi * \frac{\Delta R}{\lambda}$$

2.2.4 Topographic interferogram

To obtain the topographic phase, which will lead us to the topographic interferogram, we subtract the reference phase (phase of the flat Earth interferogram) from the interferometric phase. This operation is called interferogram flattening. Assuming that there is not ground deformation between the times t_1 and t_2, the result of the operation is:

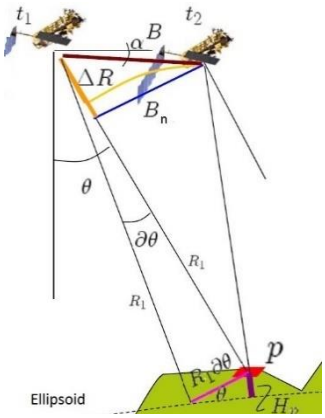


Figure 26: topographic phase, from (Hanssen, 2014)

$$\phi_{int} = \frac{4\pi * B_n * H_p}{\lambda * R_1 * \sin(\theta)}$$

Where B_n is called the perpendicular baseline and H_p is the topographic height. This allows us to generate a topographic interferogram showed in Figure 27.

The topographic fringes in Figure 27 gives us the height difference between the grounds within different fringe colors. The difference between two same color fringes is 2π which corresponds to a height difference. To know how much is the height difference, we must calculate the height ambiguity.

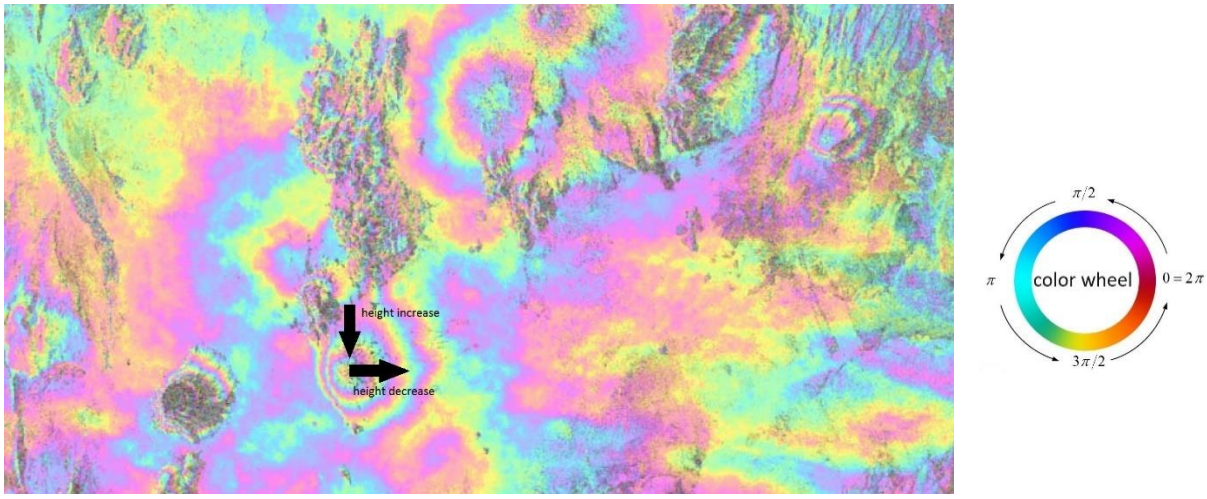


Figure 27: topographic interferogram, from (Hanssen, 2014)

The height ambiguity is defined as the height difference that generates an interferometric phase of 2π after the interferogram flattening. The equation is:

$$H_p = \frac{\lambda * R_1 * \sin(\theta)}{4\pi * B_n} * \partial\phi$$

$$if\ \partial\phi = 2\pi \rightarrow H_{2\pi} = \frac{\lambda * R_1 * \sin(\theta)}{2 * B_n}$$

We notice how the value of the height ambiguity depends on the perpendicular baseline. A high value of height ambiguity will give us less topography accuracy, whereas a small value of height ambiguity will give us a higher accuracy. That is why the perpendicular baseline is very important. The Figure 28 shows an example of two different height ambiguity values.

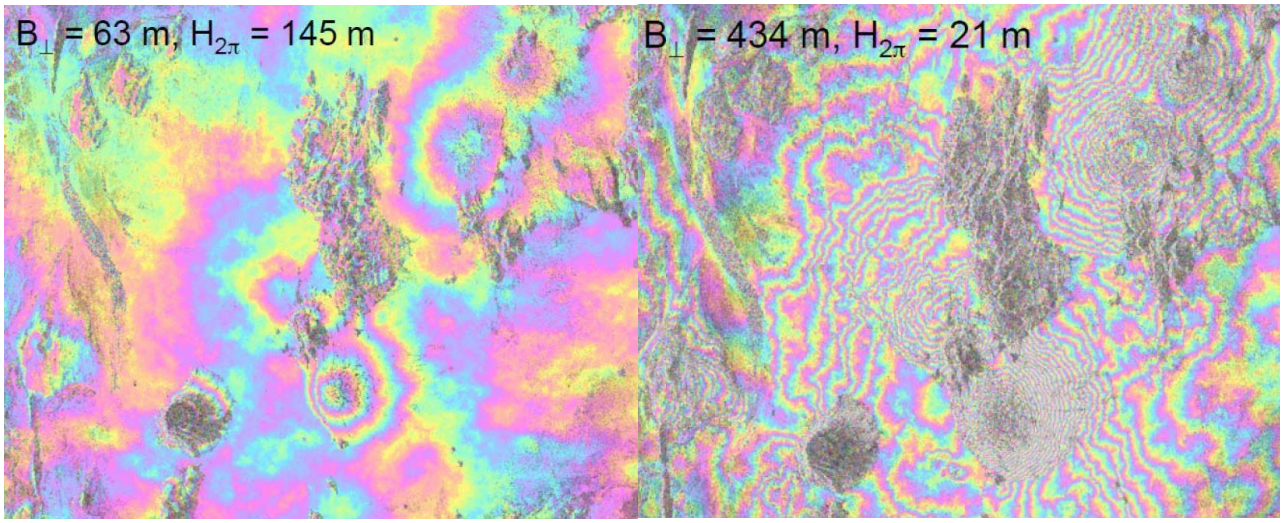


Figure 28: height ambiguity, from (Hanssen, 2014)

2.2.5 Differential interferogram

If in the time interval between two SAR images the ground has slightly changed, the interferometric phase (after the interferogram flattening) will also contain the additive phase of that slightly change called motion phase:

$$\phi_{int} = \frac{4\pi}{\lambda} * d$$

where d is the relative pixel displacement projected on the slant range direction.

This means that after the interferogram flattening, the interferometric phase will not only contain the topographic phase, but also the motion phase:

$$\phi_{int} = \frac{4\pi * B_n * H_p}{\lambda * R_1 * \sin(\theta)} + \frac{4\pi}{\lambda} * d$$

To obtain the differential interferogram we need to erase the topographic phase contribution by making the value of the perpendicular baseline equal to zero. To do this, we need that the satellite takes the pixel information in t_2 at exactly the same position than it did in t_1. In reality, the satellite cannot be in the exact same position, we will always have a difference of position between the two different times. The only way to get a differential interferogram, is to subtract the topographic contribution component using a Digital Elevation Model.

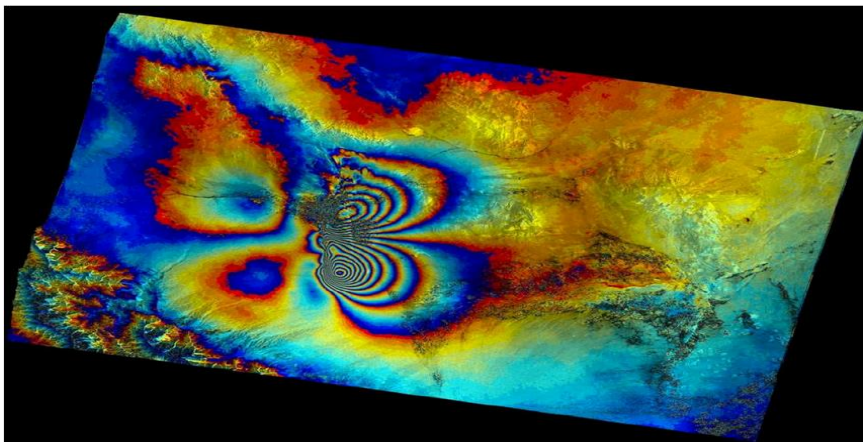


Figure 29: differential interferogram of Bam earthquake in Iran, from (European Space Agency)

The differential interferogram show us the movement of the ground between two different times. The height between two fringes of the same color is calculated by:

$$d = \frac{2\pi * \lambda}{4\pi}$$

Let us notice that the interferometric phase is much more sensitive to the motion phase than the topographic phase. For example, let's say we use a satellite with $\lambda = 5.6cm$ and with a perpendicular baseline of $150m$, the interferometric phase will be:

$$\phi_{int} = \frac{H_p}{10} + 225d$$

A deformation value d will have much more impact than a topographic value H_p.

2.3 POLARIMETRIC SYNTHETIC APERTURE RADAR (POLSAR)

2.3.1 Introduction

In the Synthetic Aperture Radar part (2.1), we use the amplitude of the backscattered signal to generate a radar image. In the Interferometric Synthetic Aperture Radar part (2.2) we saw how to use the phase information of the backscattered signal to generate an interferogram, giving us more ground height information. In this part, we will use the polarimetry of the electromagnetic waves to get more ground information. I will not go into the details, but I will give an overview of the subject to have an idea of why we use it and how it works.

Polarimetry is the measurement and the interpretation of the polarization state of an electromagnetic wave. The polarization information of the signals backscattered is related to its geometrical structure reflectivity, shape, orientation, and geophysical properties (humidity, roughness …).

2.3.2 Radar polarimetry

For full polarimetry, the radar must be able to transmit horizontal and vertical waves, and to receive horizontal and vertical waves. When the radar transmits a horizontal or a vertical wave, the signal back scattered can be with a horizontal polarization, or a vertical polarization. The radar measures the signal back scattered after transmitting a horizontal wave and a vertical wave (see …). Thereby, for one cell we will generate a matrix of 4 values called Sinclair matrix, which will contain the values of the backscattered signals. Assuming that we are in the case of a monostatic system (the receiver and the transmitter are collocated), the matrix will be:

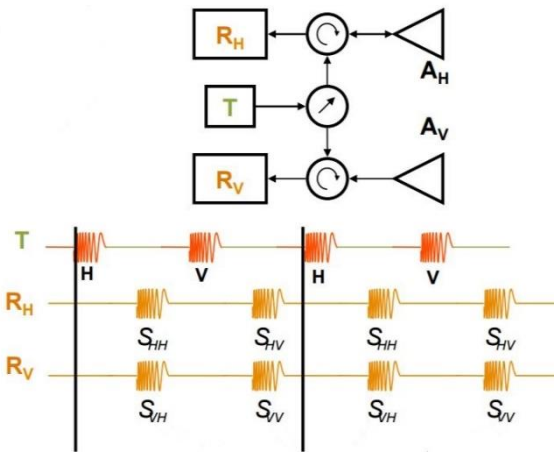


Figure 30: full polarimetric radar monostatic system, from (Gabriel Vasile, 2013)

$$Sinclair\ matrix \rightarrow [S] = \begin{bmatrix} S_{HH} & S_{HV} \\ S_{VH} & S_{VV} \end{bmatrix}$$

where S_{HH} is the horizontal backscattered signal after sending a horizontal wave, S_{VH} is the vertical backscattered signal after sending a horizontal wave, S_{VV} is the vertical backscattered signal after sending a vertical wave and S_{HV} is the horizontal backscattered signal after sending a vertical wave

If, for example, we transmit a horizontal wave and we receive a horizontal wave (HH) and a vertical wave (VH), and if we transmit a vertical wave, and we receive a horizontal wave (HV) and a vertical wave (VV), we can generate an image for each signal back scattered. Assuming that HV=VH, we have:

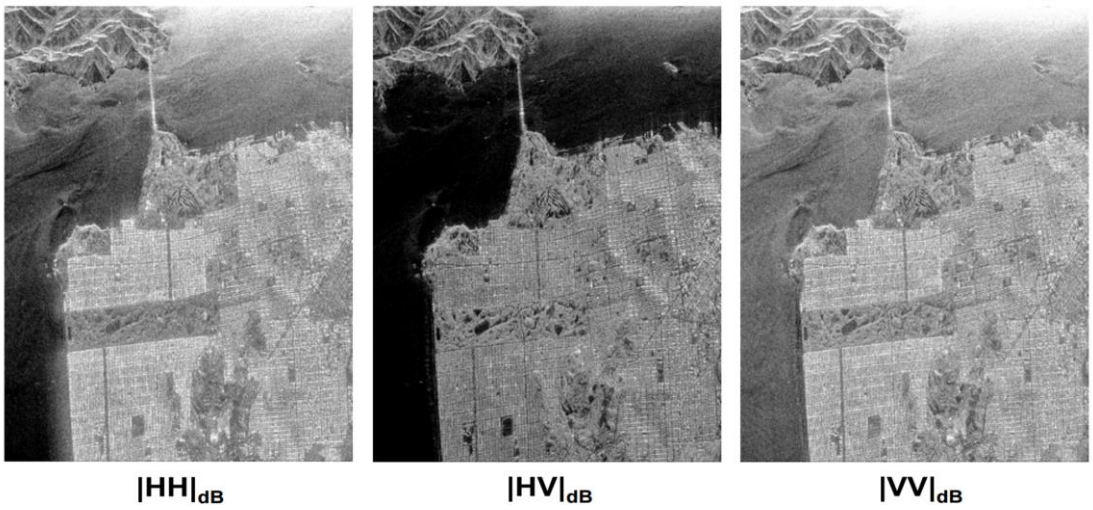


Figure 31: backscattered signals, from (Pottier, 2017)

With these three images (giving us three parameters), we could generate an RGB image and thereby, generate a colored picture of the ground. For example, if we use an HH image as the blue component, an HV image as the green component, and a VV image as the red component we should have:

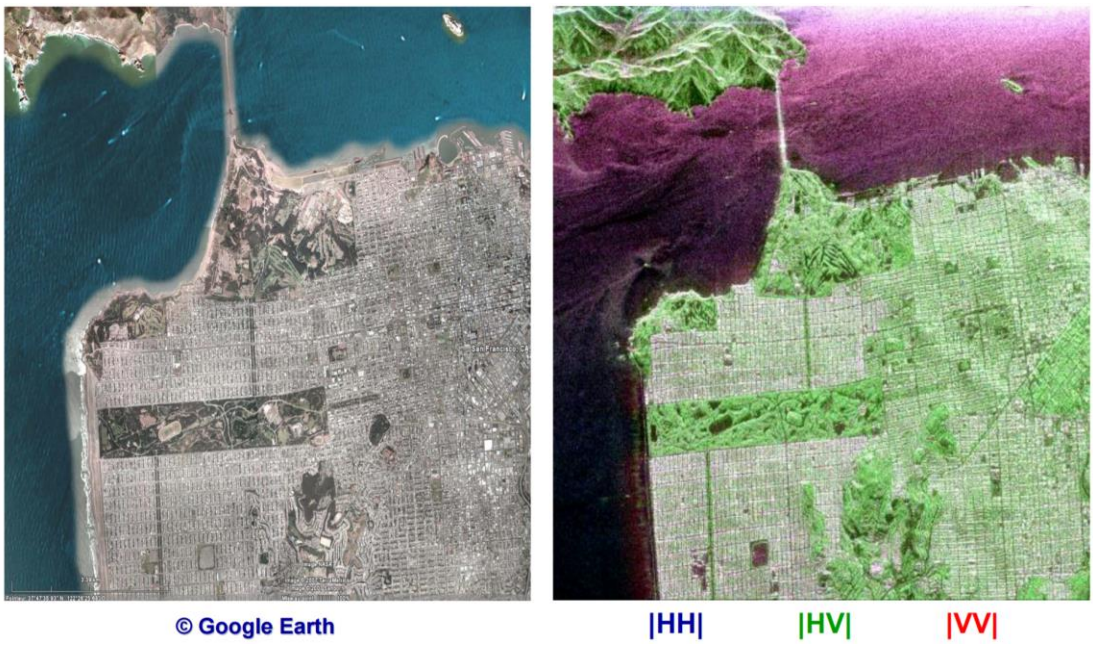


Figure 31: colored image with |HH|, |HV| and |VV|, from (Pottier, 2017)

The resulting image has not an easy physical interpretation due to the colors (the sea is not purple in real life). We can improve this image using different RGB values, HH+VV as blue color, HV as green color and HH-VV as red color.

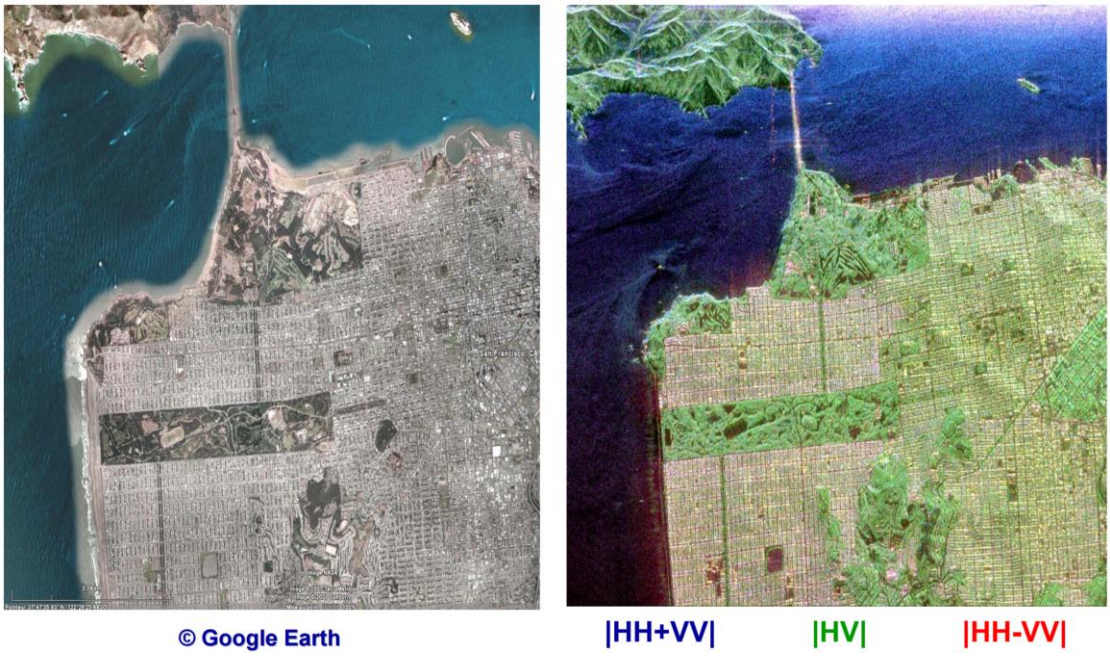


Figure 32: colored image with |HH+VV|, |HV| and |HH-VV|, from (Pottier, 2017)

The new colored image allows us an easier physical interpretation. Each image component has a physical meaning, HH+VV represents a single bounce scattering of the transmitted signal (for example the signals scattered by the sea usually have a single bounce), HV represents multiple bounces scattering due to the target volume (vegetation) and HH-VV represents a double bounce scattering (for example the scattered signals from the city).

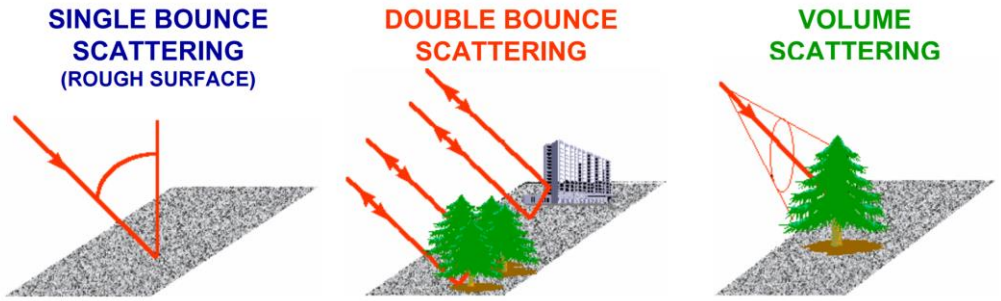


Figure 33: bounce scaterring of signals, from (Pottier, 2017)

2.3.3 Polarization equations

The propagation wave can be described as an electric field vector ($\vec{E}_{z,t}$) with two components, a horizontal component (E_x) and a vertical component (E_y):

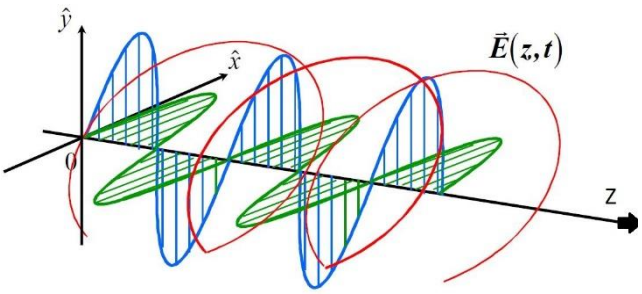


Figure 34: electric field vector, from (Pottier, 2017)

$$\vec{E}_{z,t} = \begin{cases} E_x = A_x * \cos(2\pi ft - kz - \varphi_x) \\ E_y = A_y * \cos(2\pi ft - kz - \varphi_y) \\ E_z = 0 \end{cases}$$

f = frequency

kz = propagation information

φ_n = phase

A_n = amplitude

If we eliminate the time space propagator ($2\pi ft - kz$) of the equation above, it leads us to an ellipse equation called, the polarization ellipse:

$$\left(\frac{E_x}{A_x}\right)^2 - 2\left(\frac{E_x * E_y}{A_x * A_y}\right)\cos(\varphi_y - \varphi_x) + \left(\frac{E_y}{A_y}\right)^2 = sin^2(\varphi_y - \varphi_x)$$

Thereby, we can represent the polarization of the wave in the plane as an ellipse. The electric field vector ($\vec{E}_{z,t}$) moves in time along the polarization ellipse. The geometrical parameters to describe the ellipse are:

Amplitude = A

Orientation angle = $-\frac{\pi}{2} \leq \phi \leq \frac{\pi}{2}$

Ellipticity angle = $0 \leq \tau \leq \frac{\pi}{4}$

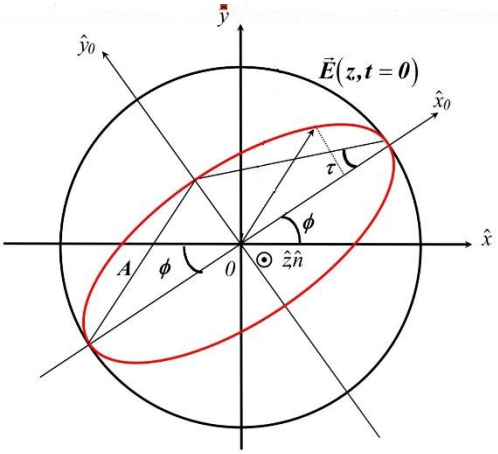


Figure 35: polarization ellipse, from (Pottier, 2017)

Their equations are:

$$A = \sqrt{A_x^2 + A_y^2} \qquad \tan(2\phi) = 2\frac{A_x * A_y}{A_x^2 - A_y^2}\cos(\varphi_y - \varphi_x) \qquad \sin(2\tau) = 2\frac{A_x * A_y}{A_x^2 + A_y^2}\sin(\varphi_y - \varphi_x)$$

A full polarimetric radar will measure the back scattered wave, giving the values of the amplitudes A_x and A_y, and the values of the phases φ_x and φ_y. Another way to represent the polarization of a wave is using Jones Vector:

$$\vec{E}_{z,t} = \begin{cases} E_x = A_x * \cos(2\pi ft - kz - \varphi_x) \\ E_y = A_y * \cos(2\pi ft - kz - \varphi_y) \\ E_z = 0 \end{cases} \rightarrow E = \begin{bmatrix} E_x \\ E_y \end{bmatrix}$$

Using Jones Vector, the orientation angle and the ellipticity angle, we can analyze the polarization state of a wave. The au-dessous shows some polarizations states.

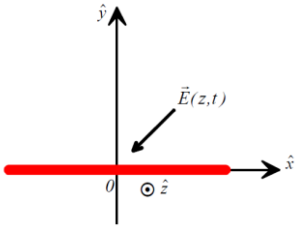
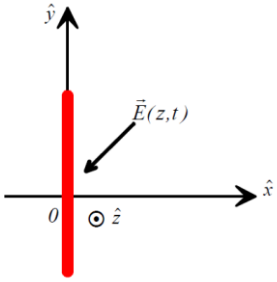
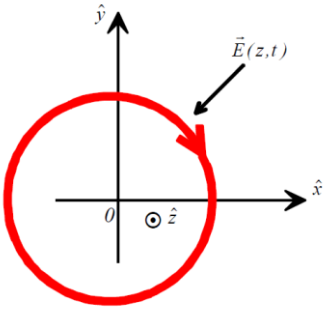
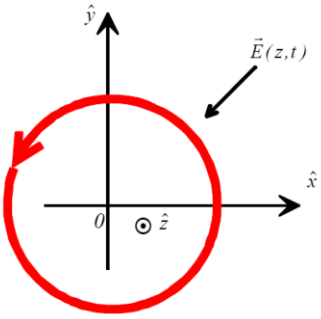
Polarization state	Conditions	Figure
Horizontal polarization state	$E = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\phi = 0$ $\tau = 0$	
Vertical polarization state	$E = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\phi = \frac{\pi}{2}$ $\tau = 0$	
Left circular polarization state	$E = \begin{bmatrix} 1 \\ j \end{bmatrix}$ $-\frac{\pi}{2} \leq \phi \leq \frac{\pi}{2}$ $\tau = \frac{\pi}{4}$	
Right circular polarization state	$E = \begin{bmatrix} 1 \\ -j \end{bmatrix}$ $-\frac{\pi}{2} \leq \phi \leq \frac{\pi}{2}$ $\tau = -\frac{\pi}{4}$	

Figure 36: table of polarization states, from (Pottier, 2017)

2.3.4 Backscattering polarization equations

In the polarization equations, we talked about the polarization ellipse and which parameters do we need to figure it out the polarization wave. In this part, we will see the information received by the radar receiver and the equations to write it down. At the beginning, we talked about the Sinclair matrix, we will use this matrix as the starting point to generate four other ways to show the information in a monostatic system ($S_{HV} = S_{VH}$): the target vectors Pauli and Lexicographic, and the 3x3 Coherency and Covariance matrices. We do not generate new information, we just represent the information in another way which can be easier to work on it depending on the application.

The Sinclair matrix is generated for each pixel using the information from the radar receiver:

$$[S] = \begin{bmatrix} S_{HH} & S_{HV} \\ S_{VH} & S_{VV} \end{bmatrix}$$

where S_{HH} is the horizontal backscattered signal after sending a horizontal wave, S_{VH} is the vertical backscattered signal after sending a horizontal wave, S_{VV} is the vertical backscattered signal after sending a vertical wave and S_{HV} is the horizontal backscattered signal after sending a vertical wave.

After we generate the Sinclair matrix, we can vectorize it and write the two target vectors, allowing us a direct physical interpretation of the backscattering processes (we replace S_{HV} and S_{VH} by S_{xx}).

$$[S] \xrightarrow{vectorisation} \underline{S} = \frac{1}{2} Trace([S][\psi])$$

$$Pauli\ basis\ vector \rightarrow [\psi_P] = \left\{ \sqrt{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \sqrt{2} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \sqrt{2} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \sqrt{2} \begin{bmatrix} 0 & -j \\ j & 0 \end{bmatrix} \right\}$$

$$Pauli\ target\ vector \rightarrow \underline{k}_{3P} = \frac{1}{\sqrt{2}} [S_{HH} + S_{VV} \quad S_{HH} - S_{VV} \quad 2S_{xx}]^T$$

$$Lexicographic\ basis\ vector \rightarrow [\psi_L] = \left\{ 2 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad 2 \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad 2 \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \quad 2 \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right\}$$

$$Lexicographic\ target\ vector \rightarrow \underline{k}_{3L} = \begin{bmatrix} S_{HH} & \sqrt{2} S_{xx} & S_{VV} \end{bmatrix}^T$$

In the Pauli target vector, we have mixed polarizations, $S_{HH} + S_{VV}$ represents a single bounce scattering (see Figure 33), $S_{HH} - S_{VV}$ represents a double bounce scattering and S_{HV} represents a multiple bounce scattering. The Pauli target vector gives us a direct physical interpretation of the target physical properties and the scattering mechanisms. The Lexicographic target vector has only single channels, allowing us to analyze single polarizations. When we need to process the information, e.g. reduce the speckle we prefer to use the 3x3 Coherency and Covariance matrices because the target vectors contain complex information with a random phase.

$$Covariance\ matrix \rightarrow [C] = \underline{k}_{3L} * \underline{k}_{3L}^{*T}$$

$$[C] = \begin{bmatrix} |S_{HH}|^2 & \sqrt{2}S_{HH}S_{xx}^* & S_{HH}S_{VV}^* \\ \sqrt{2}S_{HH}^*S_{xx} & 2|S_{xx}|^2 & \sqrt{2}S_{xx}S_{VV}^* \\ S_{HH}^*S_{VV} & \sqrt{2}S_{xx}^*S_{VV} & |S_{VV}|^2 \end{bmatrix}$$

$$Coherency\ matrix \rightarrow [T] = \underline{k}_{3P} * \underline{k}_{3P}^{*T}$$

$$[T] = \begin{bmatrix} |S_{HH} + S_{VV}|^2 & (S_{HH} + S_{VV})(S_{HH} - S_{VV})^* & 2(S_{HH} + S_{VV})S_{xx}^* \\ (S_{HH} + S_{VV})^*(S_{HH} - S_{VV}) & |S_{HH} - S_{VV}|^2 & 2(S_{HH} - S_{VV})S_{xx}^* \\ 2(S_{HH} + S_{VV})^*S_{xx} & 2(S_{HH} - S_{VV})^*S_{xx} & 4|S_{xx}|^2 \end{bmatrix}$$

The Covariance matrix contains the intensities of the different polarizations in the diagonal (real numbers) and the covariance (complex number) between the channels out of the diagonal and thereby, we can see how they are correlated. It allows us to directly analyze single polarization channels as well as their correlations. The Coherency matrix contains the different scattering mechanisms in the diagonal (real numbers) and their correlations (complex numbers) out of the diagonal. It allows us to directly analyze single scattering mechanisms as well as their respective contribution to the total signal. If one of these matrices is known, it is possible to calculate the other one using the unitary transformation matrix $[U]$.

$$[C] = [U]^{-1}[T][U] \rightarrow \underline{k}_{3L} = [U]^{-1}\underline{k}_{3P}$$

$$[T] = [U][C][U]^{-1} \rightarrow \underline{k}_{3P} = [U]\underline{k}_{3L}$$

$$with\ [U] = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & -1 \\ 0 & \sqrt{2} & 0 \end{bmatrix}\ and\ [U]^{-1} = [U]^{*T}$$

2.4 PRINCIPAL COMPONENT ANALYSIS (PCA)

2.4.1 Introduction

Principal Component Analysis is a method of dimension reduction. It performs a linear transformation on the data so that most of the information is captured by the first principal components which are linear combinations of the original variables. All the principal components are orthogonal, avoiding any redundant information. The first principal component has most of the variance, followed by the second, the third, and so on. Thereby to get most of the information, we do not need to pick all the dimensions, but only the firsts principal components. This means that we reduce the dimensionality of the original data.

Example:

Let us say we have original data plotted in 3 dimensions, and we plot it for each single dimension.

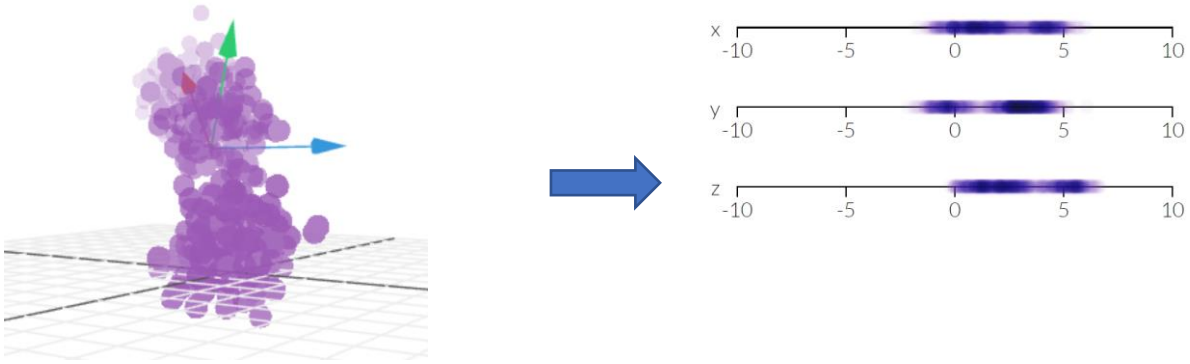


Figure 37 : 3D data from (Powell, s.d.)

If we pick only one or two of the dimensions, we would lose a lot of information because we look at the information on the wrong way. That is why we perform a PCA, it will show the same information, but in another way, allowing us to pick less dimensions containing most of the variance.

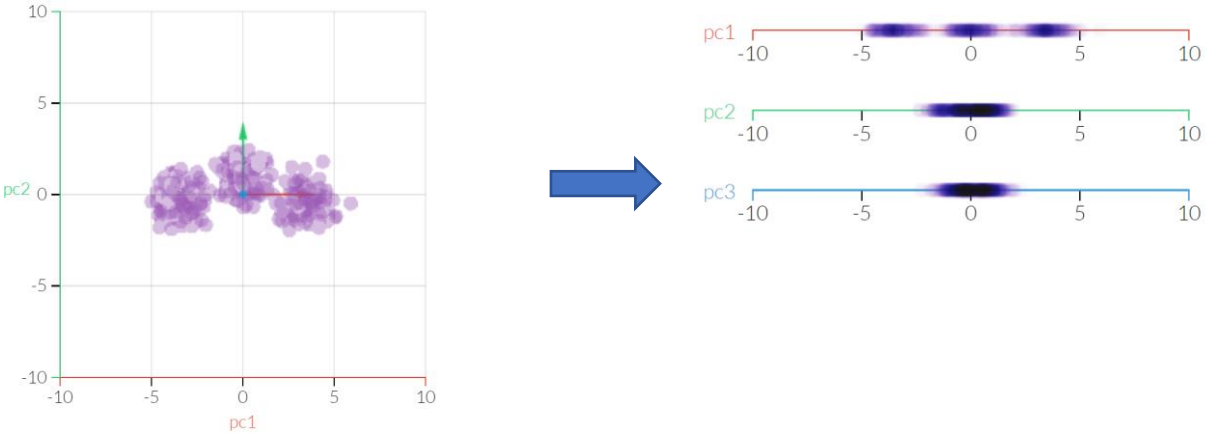


Figure 38: PCA from (Powell, s.d.)

We can see how the first principal component (pc1) contains most of the variation. In this case, using only the first principal component, will allow us to reduce the dimension of the data while keeping most of the information.

2.4.2 Methodology

There are three steps to perform a PCA. Let us say we want to perform a PCA to a 2-dimensional data set, the steps are:

1. The first step consists of calculating the covariance matrix.

In our case of a 2D data set, the covariance will look like:

$$C = \begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{bmatrix}$$

$$cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

where $\bar{x}$, $\bar{y}$ are the mean of the x data set and of the y data set, respectively. A positive value of covariance indicates both dimensions increase or decrease together. A negative value of covariance indicates that when one dimension increases the other decreases or vice-versa. A zero value indicates an independence between the two dimensions.

2. The second step consists of calculating the eigenvectors and eigenvalues.

The idea here is to convert the data into the new axes (principal components) by multiplying the original data by the eigenvectors. The eigenvectors indicate the direction of the principal components and to each eigenvector will correspond an eigenvalue. The magnitude of the eigenvalue indicates the distribution of the data.

Before calculating the eigenvectors, we calculate the eigenvalues using the formula:

$$\det(C - \lambda I) = 0$$

where C is the covariance matrix, I is the diagonal identity matrix and λ represent the eigenvalues. The eigenvectors can be calculated using the formula:

$$Ce_i = \lambda_i e_i$$

where e_i is the eigenvector and λ_i is its eigenvalue. The first principal component is the eigenvector with the higher eigenvalue and the second principal component (in our example we only have two because we only have a 2D data set) is the eigenvector with the less eigenvalue.

3. The third and last step is to re-orient the data onto our new axes, the principal components.

To calculate the re-oriented data, we use the formula:

$$re - oriented\ data = [original\ data][eigenvectors]$$

$$re - oriented\ data = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \dots & \dots \\ x_n & y_n \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix} (for\ our\ example)$$

The re-oriented data will be a matrix of size $n * 2$ in our example.

2.5 SUPPORT VECTOR MACHINE (SVM)

2.5.1 Introduction

Support Vector Machine is a supervised machine learning algorithm which can be used for both, classification of the input data, or regression (predict the value of a variable). Originally, the algorithm has been developed to be used for binary classification, or in other words, to classify information between two classes. It is supervised because we will train the algorithm using training data each assigned to its respective class. Once the algorithm trained, we just have to send the data (called the input data) to be classified. In this research, we adapt the algorithm to be used as a multiclass classifier.

In the case of two classes which can be linearly separable, the algorithm goal is to find the optimal hyperplane separating these two classes. Optimize the hyperplane means to maximize the margin of the data.

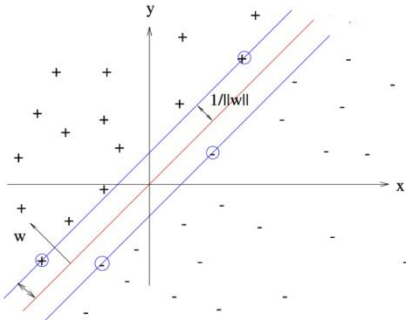


Figure 39: SVM

2.5.2 Equations

Let us imagine we have two classes of data (linearly separable) separated by a straight line called the hyperplane, and that the objective is to find the optimal hyperplane by maximizing the margin. We define our data and the hyperplane as:

$$data \rightarrow (\vec{x}_1, y_1), (\vec{x}_2, y_2) \dots (\vec{x}_n, y_n)$$

$$hyperplane \rightarrow \vec{w} * \vec{x} + b = 0$$

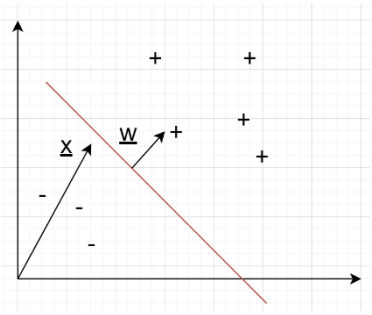


Figure 40: hyperplane separating the two sets of data

Where $\vec{x}$ is a vector pointing to a value, $\vec{w}$ is the normal vector to the hyperplane with an unknown length, y_n are either -1 or +1 indicating the class (positive or negative) to which, the pointed value of $\vec{x}_n$ belongs, and b is a constant.

The hyperplane equation contains two unknown parameters, the length of $\vec{w}$ and the value of the constant b. To solve this, we will add more constraints. Let us say that we have a positive value pointed by the vector $\vec{x}_+$ and a negative value pointed by the vector $\vec{x}_-$. We define that if the sample belongs to the positive class, the value will be $+1$ or greater, and if the sample belongs to the negative class, the value will be -1 or less.

$$\vec{w} * \vec{x}_- + b \leq -1$$

$$\vec{w} * \vec{x}_+ + b \geq 1$$

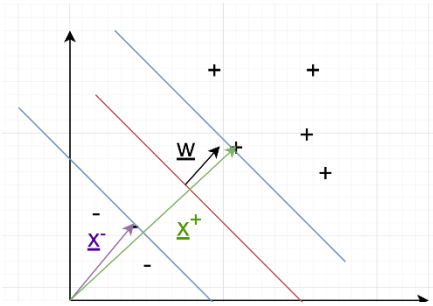


Figure 41: Margin = distance between the two blues lines

These two new equations will add two more straight lines (see figure 41) parallel to the hyperplane. Let us we multiply the equations by their corresponding y_i:

$$y_- = -1 \quad \rightarrow \quad y_-(\vec{w} * \vec{x}_- + b) \geq 1$$

$$y_+ = +1 \quad \rightarrow \quad y_+(\vec{w} * \vec{x}_+ + b) \geq 1$$

This shows us that whether the sample is in the positive or negative class, the equation will remain the same:

$$y_i(\vec{w} * \vec{x}_i + b) - 1 \geq 0$$

For the closest samples of the hyperplane (the samples in the blue lines in the figure 41), the equation will be:

$$y_i(\vec{w} * \vec{x}_i - b) - 1 = 0$$

The vectors pointing $\vec{x}_i$ on the closest samples of the hyperplane are called support vectors. Now, the objective of the algorithm is to find the optimal hyperplane which will separate the two classes. For that, we need to maximize the margin of the data which is the distance between the two blue lines in the figure. We can express the margin as:

$$margin = (\vec{x}_+ - \vec{x}_-) * \frac{\vec{w}}{||w||}$$

Where $\vec{x}_-$ and $\vec{x}_+$ are support vectors, and $\frac{\vec{w}}{||w||}$ is the unit vector.

$$using \quad y_i(\vec{w} * \vec{x}_i - b) - 1 = 0$$

$$we\ have\ with\ y_+ = 1 \rightarrow \vec{x}_+ = \frac{1+b}{\vec{w}} \quad and \quad with\ y_- = -1 \rightarrow \vec{x}_- = \frac{b-1}{\vec{w}}$$

$$margin = \left(\frac{1+b}{\vec{w}} - \frac{b-1}{\vec{w}}\right) * \frac{\vec{w}}{||w||}$$

$$margin = \frac{2}{||w||}$$

Maximize $\frac{2}{||w||}$ is equal to minimize $||w||$ and this is equal to minimize $\frac{1}{2}||w||^2$ .We wrote the last expression because it is mathematically convenient (easier to deal in the Lagrange function). So, we want to minimize $\frac{1}{2}||w||^2$ subject to the constraints $y_i(\vec{w} * \vec{x}_i - b) \geq 1$. To find the extreme of a function with constraints we use Lagrange multiplier. It will give us a new expression which we can maximize or minimize without thinking anymore in the constraints. The Lagrange function is defined by:

$$\mathcal{L}_{x,y,\lambda} = f_{x,y} - \lambda(g_{x,y} - c)$$

Where $f_{x,y}$ is the function we want to find the extremes of, $g_{x,y} = c$ is the constraint, and λ is the Lagrange multiplier. Applying it to our problem, we have:

$$\mathcal{L} = \frac{1}{2}||w||^2 - \sum \lambda_i(y_i(\vec{w} * \vec{x}_i - b) - 1)$$

Solving the above Lagrange optimization problem will give us w,b and the λ_i coefficients that determines a unique maximal margin solution.

In the case of non-linear separable data, we consider the problem to another dimension which could allow to linearly separate the data. And to find the hyperplane in that new dimension that separate the data, we use the Kernel trick.

2.6 HISTOGRAM OF ORIENTED GRADIENTS (HOG)

2.6.1 Introduction

Histogram of Oriented Gradients is a feature descriptor algorithm which converts a fixed size image to a fixed size feature vector by extracting the useful information. It describes a local object appearance and shape within an image using the distribution of oriented gradients.

2.6.2 Methodology

There are three main steps to calculate the HOG of an image:

1. Computation of the Gradient value

The first step is to calculate the vertical and horizontal gradient by filtering the image with the kernels: $[-1\ 0\ 1]$ in both horizontal and vertical directions:

$$kernel\ filters \rightarrow D_x = [-1\ 1\ 0] \quad and \quad D_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

$$gradients \rightarrow g_x = D_x * I \quad and \quad g_y = D_y * I$$

We can now calculate the magnitude of the gradient and its orientation for each pixel:

$$gradient\ magnitude \rightarrow g = \sqrt{g_x^2 + g_y^2}$$

$$orientation \rightarrow \theta = \arctan\left(\frac{g_y}{g_x}\right)$$

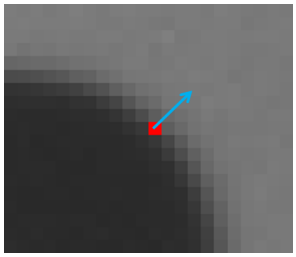


Figure 42: magnitude and orientation gradient from (McCormick, 2013)

2. Orientation binning

The second step consists of creating the cell histograms. Thereby, the image is divided into square cells containing $p * p$ pixels. The objective is to generate a histogram of n-bin using the oriented gradient from each pixel within the cell. The histogram channels can be spread from 0 to 180 degrees or from 0 to 360 degrees, depending on whether the gradient is "unsigned" or "signed".

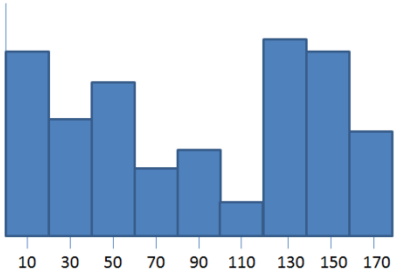


Figure 43: histogram from (McCormick, 2013)

Let's say we have an image of 60x60 pixels and that we decide to divide the image in cells of 6x6 pixels. Thereby, we will have an image containing 10x10 cells, each containing 6x6 pixels. In each cell, we will generate a histogram of 9-bin which means that we will have in the case of "unsigned" gradients, an array corresponding to 9 angles 0,20,40,60,80,100,120,140 and 160 degrees. At each angle, we will have the corresponding gradient magnitudes. This means that instead of having 10x10 matrix information for each cell, we will have a vector with 9 components (9 bins per histogram)

3. Block normalization

The last step is to normalize the cells to make our gradients independent from lighting and contrast variation. In order to do that, we regroup the cells to generate blocks containing 2x2 cells, each block will deliver a feature vector containing 36 values (4 cells * 9 bins per histogram) and can have an overlap.

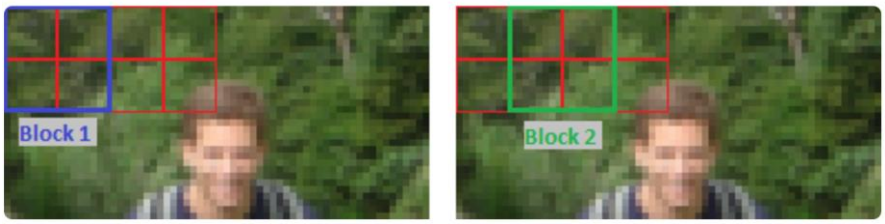


Figure 44: Blocks with 50% overlap from (McCormick, 2013)

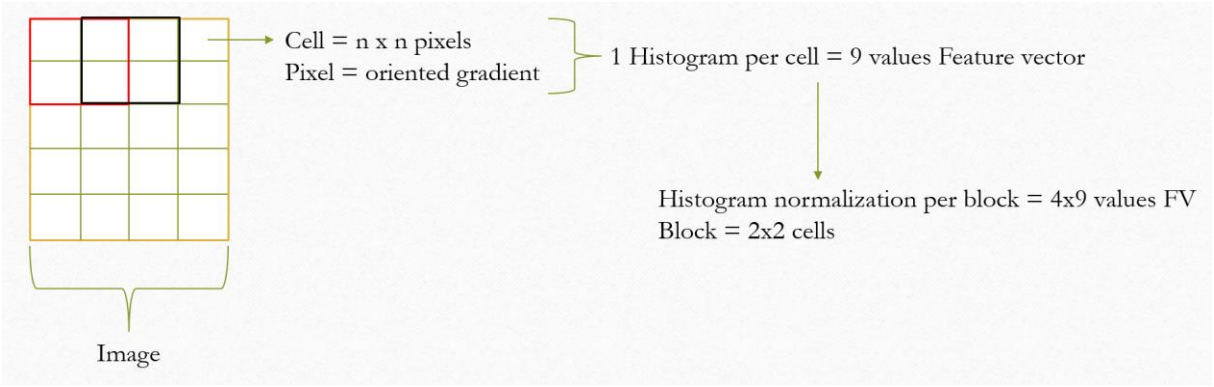


Figure 45: Overview of the HOG method using 50% overlap.

3 RESEARCH: POST-WILDFIRE VEGETATION LOSS MAPPING USING BITEMPORAL SYNTHETIC APERTURE RADAR IMAGES

3.1 INTRODUCTION

Wildfire events followed by heavy precipitation have been proven causally related to breakouts of mudflow or debris flow, which, can demand rapid evacuation and threaten residential communities and civil infrastructure. For example, in the case of the city of Glendora, California, it was afflicted by a severe wildfire and flooding event in its past (1968) and the mudslides and debris flow of 1969 killed 34 people. Therefore, the burn area mapping due to wildfire events is critical to agencies for preparing for secondary hazards, particularly mudflow and flooding prior to the next large rainfall. However, rapid post-wildfire mapping of burned areas (that are usually covered by vegetation; hence also called vegetation loss mapping) is not readily obtained by regular remote sensing methods, e.g. various optical methods, due to the presence of smoke, haze, and rainy/cloudy conditions that often follow a wildfire event. In this paper, we will introduce and develop a methodology framework that uses Synthetic Aperture Radar images that are sensed prior to and after a wildfire event. A supervised machine learning algorithm will be proposed to classify burned and intact areas using the pre- and the post-event SAR data. The 2014 Colby Fire event, which affected the downstream city of Glendora, was used to evaluate the proposed framework. It burned 1952 acres of forest and destroyed five homes.

Figure 46: Colby Fire

3.2 DATASET

The NASA's UAVSAR provided the GeoTIFF images to reconstruct the PolSAR Covariance matrix of each pixel in the monostatic case before and after the wildfire event using:

- $|S_{HH}|^2$
- $|S_{VV}|^2$
- $|S_{xx}|^2$
- $S_{HH} * S_{xx}^*$
- $S_{HH} * S_{VV}^*$
- $S_{HV} * S_{VV}^*$

$$Covariance\ matrix \rightarrow [C] = \begin{bmatrix} |S_{HH}|^2 & \sqrt{2}S_{HH}S_{xx}^* & S_{HH}S_{VV}^* \\ \sqrt{2}S_{HH}^*S_{xx} & 2|S_{xx}|^2 & \sqrt{2}S_{xx}S_{VV}^* \\ S_{HH}^*S_{VV} & \sqrt{2}S_{xx}^*S_{VV} & |S_{VV}|^2 \end{bmatrix}$$

The Covariance matrix contains the intensities (real numbers) of the different polarizations in the diagonal and the covariance (complex numbers) between the channels out of the diagonal. They provided us in total nine images, three of them containing the values of $|S_{HH}|^2$, $|S_{VV}|^2$ and $|S_{xx}|^2$, and the six images left containing the real and the imaginary part of $S_{HH} * S_{xx}^*$, $S_{HH} * S_{VV}^*$, and $S_{HV} * S_{VV}^*$. For the research, we used the three first images to generate an RGB image before and after the event, with $|S_{HH}|^2$ as red, $|S_{xx}|^2$ as green, and $2|S_{VV}|^2$ as blue.

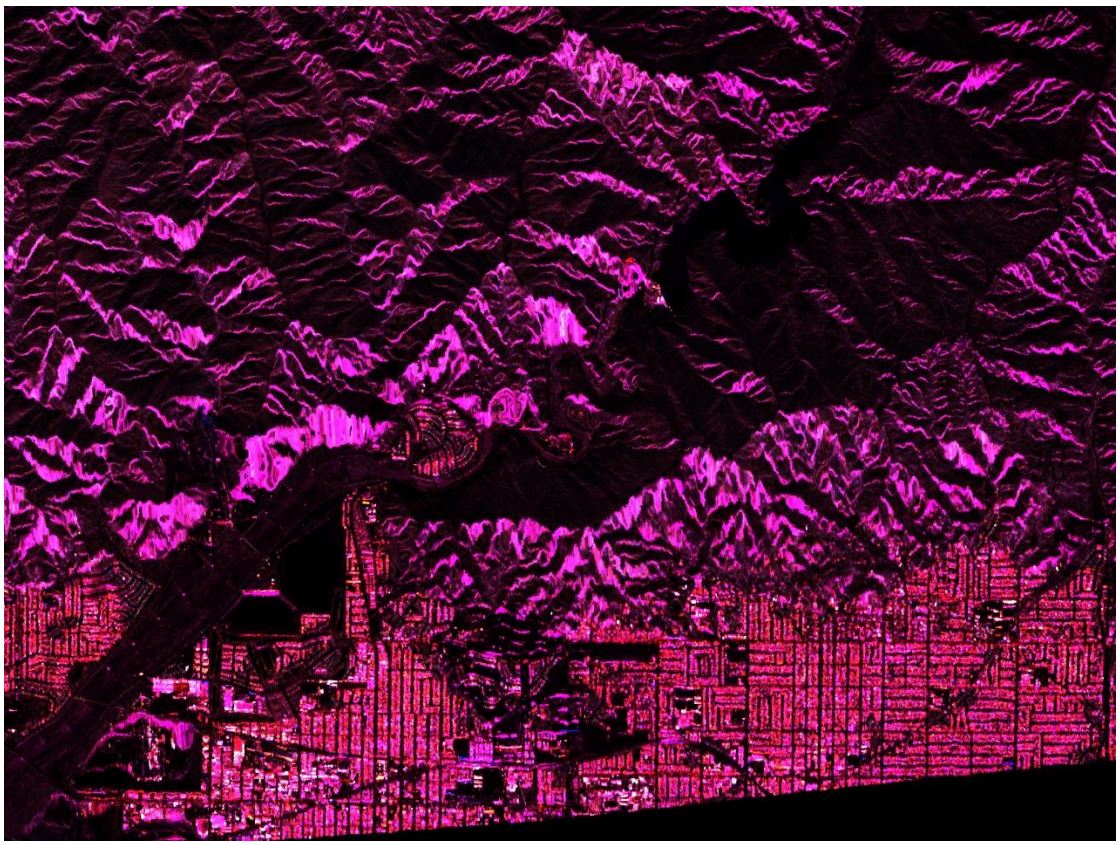


Figure 47: RGB image before the wildfire event, $|S_{HH}|^2$=R, $|S_{VV}|^2$=B and $2|S_{xx}|^2$=G

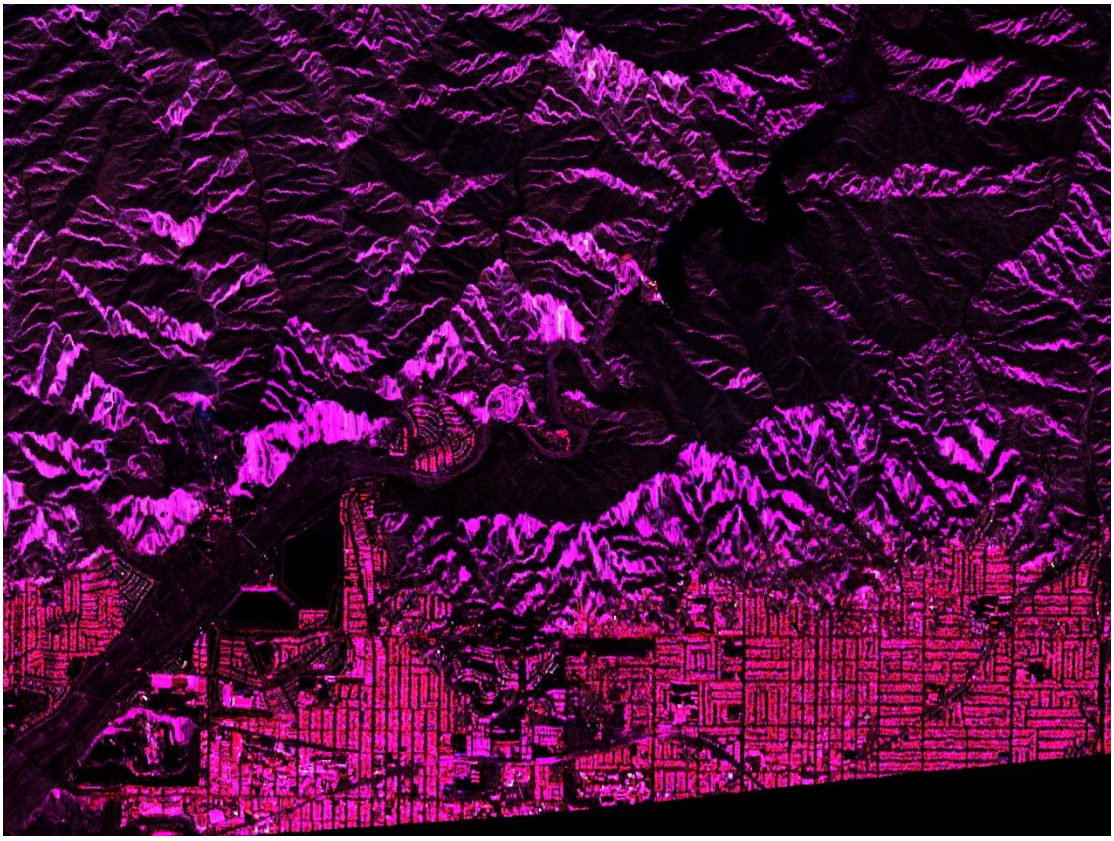


Figure 48: RGB image after the wildfire event, $|S_{HH}|^2$=R, $|S_{VV}|^2$=B and $2|S_{xx}|^2$=G

Visually, we cannot see the changes from the event between the two images, and we cannot see the burned area in the post-wildfire image (Figure 48). To detect the burned areas, we had to perform a Principal Component Analysis (PCA) differencing between the two images and then performing a per-pixel classification with a supervised machine learning using the Support Vector Machine (SVM) algorithm. We perform a PCA differencing to detect the changes. We perform a supervised machine learning to classify input data (the pixels) into four classes, based on the training of the algorithm.

The images provided by the NASA are in the GeoTIFF format, providing us with georeferencing information. This allows us to directly apply the images resulting of the classification to google earth.

3.3 METHODOLOGY

3.3.1 Generation of a pre-event and post-event RGB image

To generate the pre-event and post-event RGB image we use $|S_{HH}|^2 = R$, $|S_{VV}|^2 = B$ and $2|S_{xx}|^2 = G$ from the NASA UAVSAR covariance matrix. The results are shown in Figure 47 and Figure 48.

3.3.2 Principal Component Analysis differencing

The first step is to detect the changes between the two images applying the PCA differencing method to each component of the RGB images (Figure 47 and Figure 48). To detect and extract the changes using bi-temporal images, the first principal component contains the no changes areas between the bi-temporal images, but the second principal component contains the changes between the bi-temporal images. Thereby, the PCA differencing method re-orient all the pixels of each component (R, G, and B) from each image (pre-event and post-event) onto the second principal component. This will transpose each component of the two images onto the second principal component. The resulting components from each image are then subtracted between them and the result is shown

as a three-band PCA differencing image in Figure 49. Each band has been assigned to a color.

We directly notice the Coby wildfire event in purple, and another wildfire (Madre wildfire) which will be used as training data. We can interpret the green color as a significant change in the $|S_{xx}|^2$ polarization band between the two images. The gray is interpreted as no change in all bands. The blue is interpreted as a significant change in the $|S_{HV}|^2$ polarization band. The red is interpreted as a significant change in the $|S_{HH}|^2$ polarization band.

Another more physical interpretation of the image, is that the gray color represents no changes (many mountain areas, roads in urban areas), the green color may represent strong vegetation changes (many mountain areas), the complex combination of red, purple, blue and green, represents changes in the urban area and, the purple represents the wildfire events. This more physical interpretation allows us to distinguish four types of data to classify: no change, vegetation change, urban area change, and wildfire.

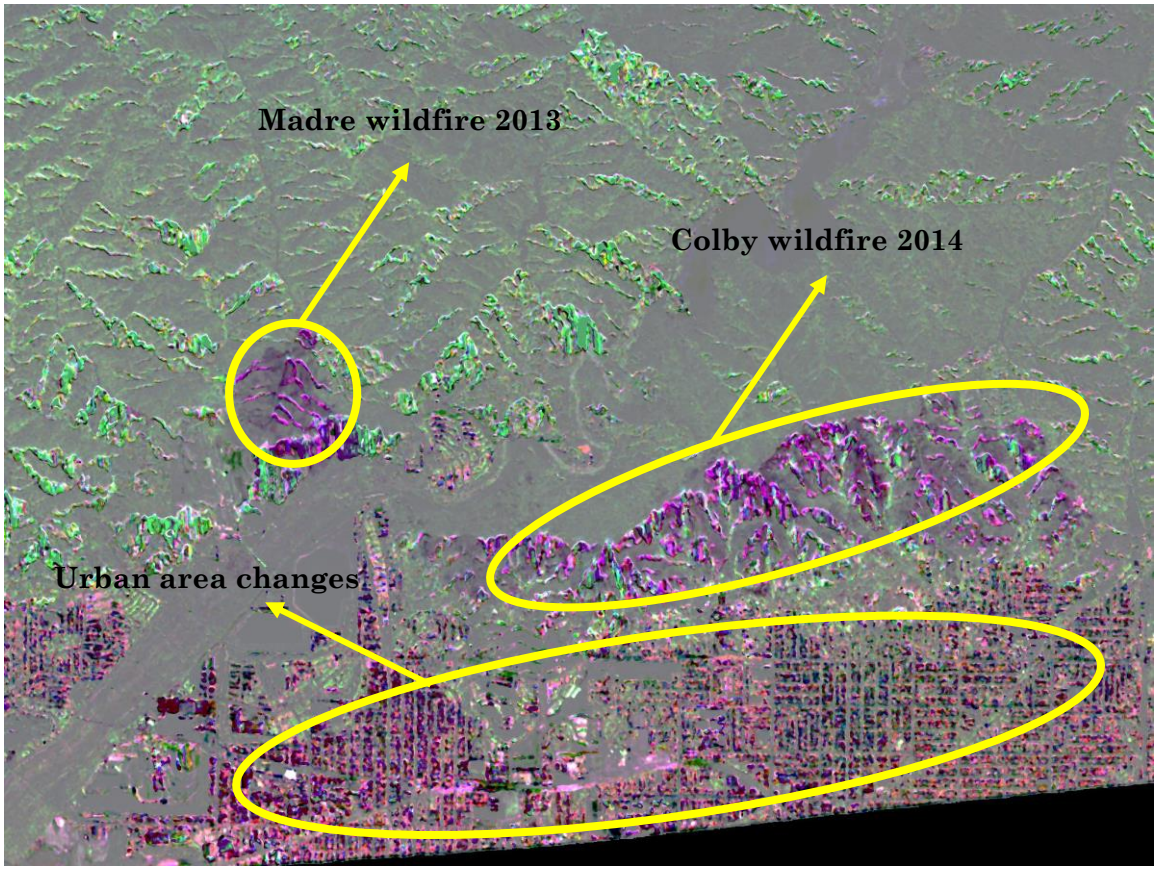


Figure 49: PCA differencing result containing 3 bands

3.3.3 Classification using Support Vector Machine algorithm

In this research, we use a linear SVM using linear Kernel. The SVM has initially been developed for binary classification, but for this research, we adapt it to use it as a multiclass classifier. To adapt it, we use the "one vs one" strategy or also known as "pairwise coupling", "all pairs" or "round robin". This strategy consists in constructing one SVM classifier for each pair of classes. Thereby, the number of classifiers with c classes will be: $c(c-1)/2$. Each SVM classifier is trained to distinguish the data between two classes.

The next step of the research is to train our classifier so it can distinguish between four different classes:

- no change;
- vegetation change;
- urban area change;
- wildfire.

The training data for each class has been selected directly from the PCA differencing image by cropping directly on the image the desired training data (in the case of the wildfire class, we took the Madre wildfire as training data) (Figure 49). Each pixel will be classified using the features from each PCA differencing band (3 features per pixel). Once the SVM trained, we classify all the pixels from the entire PCA differencing image (input data), and the result is shown on Figure 51. The problem using only the PCA differencing components as features to classify the input data, is the similarity of the wildfire pixels and urban areas pixels and is reflected in the Figure 51. To measure the accuracy of the SVM classifier, we have performed a cross validation with 10 folds and plot the confusion matrix. It tells us that 12% of the wildfire data have been mis-classified into no change class and, the more important problem, is that 34% of the urban area data has been mis-classified into wildfire. We notice that the confusion matrix does not fully represent the complete result of the classification (see Figure 50) because we visually see than more than 46% of the urban area data is mis-classified in the confusion matrix. The advantage of the confusion matrix is to allow us to detect where the problem is situated. Even if the percent of mis-classified urban area data is wrong, we can see that the problem is situated between the wildfire class and the urban area class.

		Class				**Result**	
		Vegetation	Wildfire	Urban area	No change	True Positive Rate	False Negative Rate
Class	Vegetation	63%	12%	3%	22%	63%	37%
	Wildfire	1%	84%	3%	12%	84%	16%
	Urban Area	0%	54%	34%	12%	54%	46%
	No change	0%	1%	0%	99%	99%	1%

Figure 50: Confusion matrix of the PCA differencing result

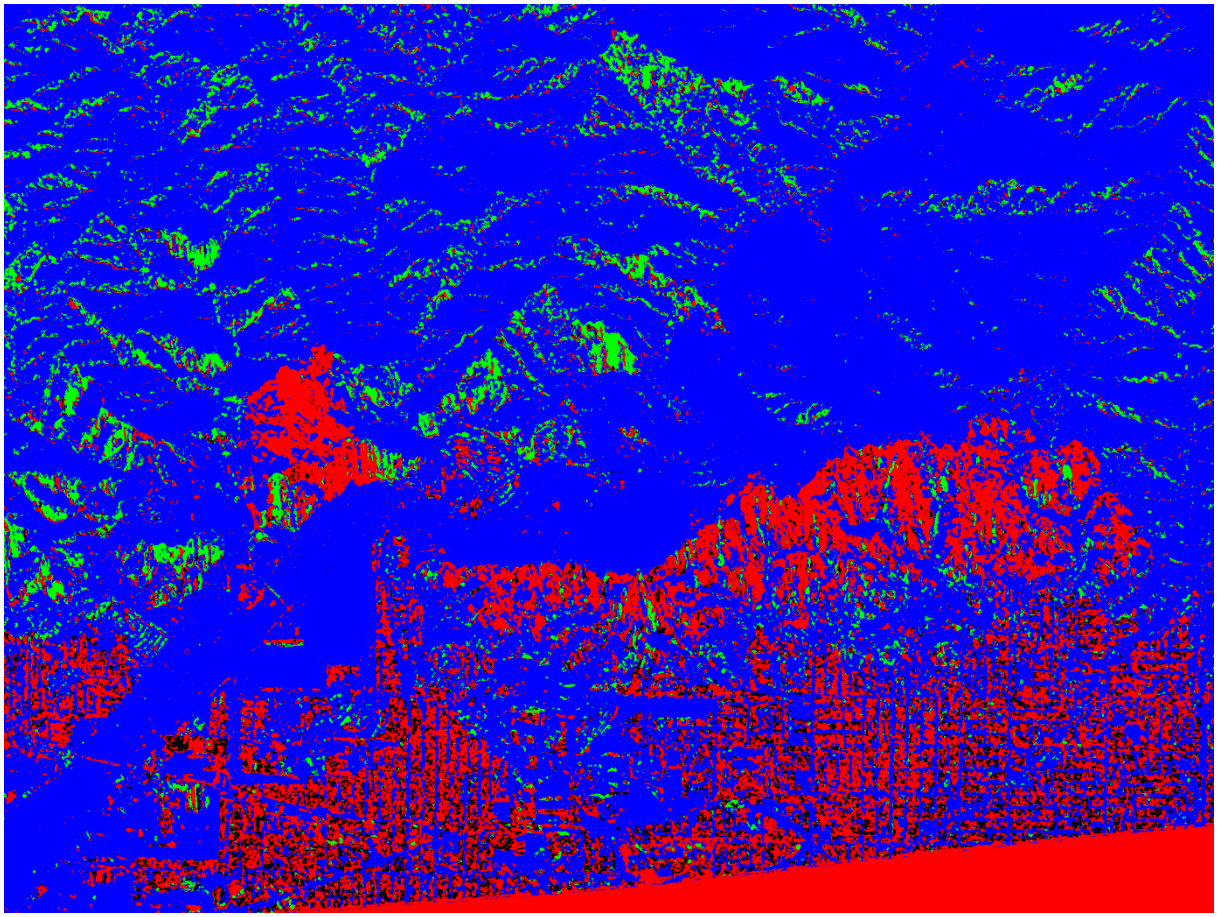


Figure 51: Per-pixel classification using 3 features from the PCA differencing image

The solution to this problem is to add more features to better differentiate the urban area pixels from the wildfire pixels. We notice that a characteristic of pixels of the urban area is that they are contained into a distinctive form (the shape of the buildings). Thereby, to add the shape as a predictor in our classification, we use the Histogram of Oriented Gradients (HOG).

The algorithm selected to perform a HOG description has been developed by Chris McCormick (it can be found here http://mccormickml.com/2013/05/09/hog-descriptor-in-matlab/). The HOG algorithm is a function who takes an image and delivers a feature vector describing the HOG of the image. The feature vector size can be calculated using the formula:

$$FV = number\ of\ blocks * number\ of\ cells\ per\ block * number\ of\ bins$$

The algorithm uses 50% of block overlap. The HOG algorithm has initially been developed to calculate the HOG of an entire image containing a distinctive form, for example a person. Applying the algorithm to different images containing a person and using the results to train a SVM classifier, results on a SVM classifier that can detect if an input image contains or not a person.

The problem is that in our case, we know that our image contains many different buildings and what we want to classify is not the whole image, but the block designed as a portion of the image. We do not need to know if there are buildings in the image, but where they are, and thereby if the block is part of a building (the urban area class). To adapt the HOG algorithm to our case, instead of sending the whole image, we send portions of the image called blocks, each containing $n * n$ pixels. Each block has an overlap of $n - 1$ pixels, giving

the possibility to have a classification of the input data accurate to a pixel. Each block is sent to the HOG algorithm, which divides the block into $2 * 2$ cells, and delivers a feature vector of 108 values.

At the beginning, we only had three features per training data (PCA bands), but now we will have 111 features:

$features = 36\ features\ per\ block * 3\ components\ (each\ band\ of\ the\ PCA\ differencing) + 3\ features\ (each\ band\ of\ the\ PCA\ differencing\ result)$.

This method is applied to both the training data and the input data.

The image below (Figure 52) shows how we divide the image into blocks. The size of the image is $8 * 10$ pixels and the block $4 * 4$ pixels. To resume, the steps are:

1. Select the first portion of the image with the red block.
2. Send the block to the HOG algorithm, which will extract the feature vector of the block.
3. Move one pixel on the right (3 pixels overlap) and select a new block (the blue block).
4. Send the blue block to the HOG algorithm, which will extract the feature vector.
5. Move one pixel on the right (3 pixels overlap) and select a new block (the orange block).
6. Send the blue block to the HOG algorithm, which will extract the feature vector and so on…

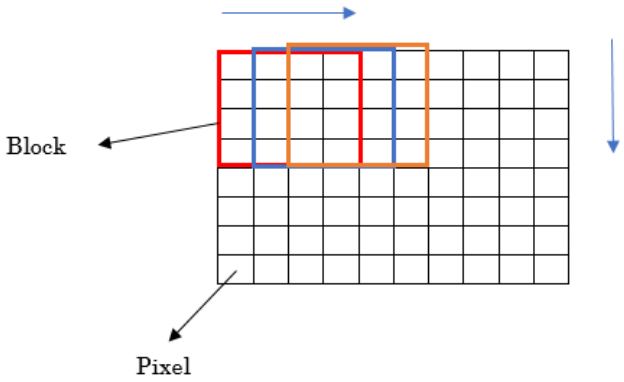


Figure 52: Image divided into $4 * 4$ pixels blocks and 3 pixels overlap

When the block reaches the top right of the image, the algorithm just restarts the process, but moving one pixel down vertically (3 pixels overlap). This method is applied to all the cropped images from the PCA differencing image used to extract the training data, and to the entire image to be classified. Using $n - 1$ pixels overlap, allow to classify the PCA differencing image like pixel-wise.

To resume the algorithm steps are:

1. Selection of the training data for each class by cropping directly into the PCA differencing image.
2. Each cropped image is divided into blocks of $n * n$ pixels with $n - 1$ pixels overlap.
3. Each block is sent to the HOG algorithm to extract the feature vector.
4. We regroup all the block results to generate the training data.
5. The algorithm is trained using the training data.
6. The PCA differencing image is divided into blocks of $n * n$ pixels with $n - 1$ pixels overlap (this allows us to have a classification accurate to a pixel).

7. Each block is sent to the HOG algorithm to extract the feature vector.
8. We regroup all the block results to generate the input data.
9. We send the input data to the SVM classifier and we get the results of the classification.
10. Using the results of the classification, we regenerate the image result.

3.4 RESULTS

The buildings have an average of 16 pixels width and thereby we have tested the algorithm with blocks of $4 * 4$ pixels, $5 * 5$ pixels, $6 * 6$ pixels, $7 * 7$ pixels and $10 * 10$ pixels. To estimate the accuracy of each test we have performed a cross validation with 5 folds. We noticed that the confusion did not represent the reality of the classification results. The confusion matrices indicate that the best result is using a block of $10 * 10$ pixels, but if we see the images of the results in Annex 2, the two best images are using blocks of $7 * 7$ and $6 * 6$ pixel. This might be due to the low number of folds used for the cross validation. Nevertheless, the confusion matrices of each test are plotted in Annex 1 just for information. The images of all the results are in Annex 2. The result which seems visually to show the best classification, is using blocks of $6 * 6$ pixels with $n - 1$ pixels overlap is shown in Figure 53. However, we still have a lot of noise all over the image due to the urban area class.

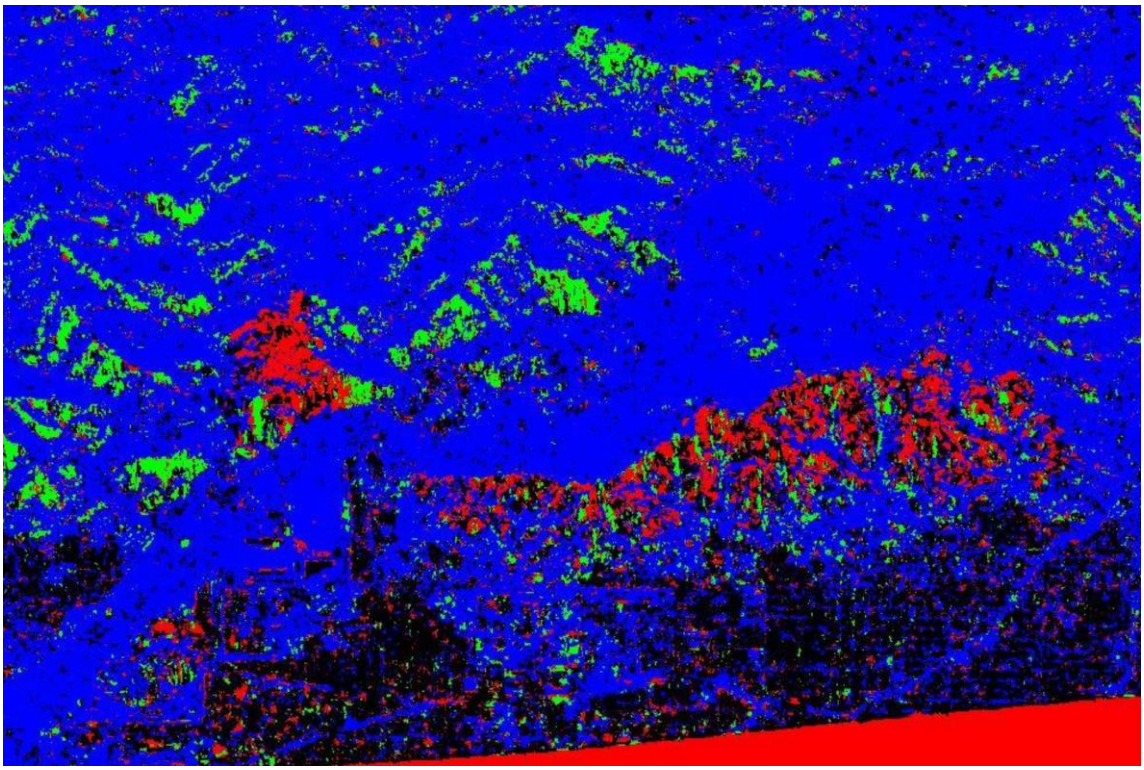


Figure 53: Result classification with blocks of $6 * 6$ pixels and 11 pixels overlap

4 CONCLUSION

Multi-temporal UAVSAR data shows a great potential in wildfire damage detection and mapping. The objectives of the research have not been completely fulfilled by this Master's thesis. The next step of the research shall be to test the Relevance Vector Machine and to generate a GeoTIFF image result. The Relevance Vector Machine has the advantage to provide probabilistic classification and, thereby, it allows to quantify the scores of the classification for each pixel classified. Generating the GeoTIFF image will allow to directly evaluate the results against field data.

There are some improvements that could be implemented:

- Improving the contrast of the image result from the PCA differencing before selecting the training data could result to an accuracy classification improvement.
- The size of the training data is too huge (111 features per pixel) and it can be reduced. One way to do this, would be to only take the oriented gradient with the highest magnitude from the three components of the PCA differencing image. This could lead from 111 features to only 39 features.
- Another way to reduce the training data would be to have less block overlap on the training data. The first tests of the research have been done with $n-1$ pixels overlap, and this may bring a lot of redundancy.
- Instead of using the covariance matrix as the starting point to generate the images, we could calculate the coherency matrix using all the data provided by the NASA.
- Generate our own HOG algorithm, which could directly process a multiple band image and choose the block overlap. Choose the block overlap directly on the HOG algorithm can drastically reduce the code complexity. We would be able not only to change the overlap on the training data, but on the input data too. If we reduce the overlap of the input data, we may have less accurate results, but we may reduce the overall error and have a better classification.
- Equalize the size of the training data of each class. I have already developed a code to do it, but I did not have enough time to test it.

ANNEXES

ANNEX 1: CONFUSION MATRICES OF THE RESULTS

Block size $n * n$ pixels	**Class**					**Result**	
		Vegetation	Wildfire	Urban area	No change	True Positive Rate	False Negative Rate
$4 * 4$	Vegetation	67%	4%	14%	15%	67%	33%
	Wildfire	2%	42%	39%	17%	42%	58%
	Urban area	1%	5%	77%	17%	77%	23%
	No change	0%	0%	5%	95%	95%	5%
$5 * 5$	Vegetation	66%	4%	15%	15%	66%	34%
	Wildfire	2%	50%	31%	17%	50%	50%
	Urban area	1%	4%	79%	16%	79%	21%
	No change	1%	0%	6%	93%	93%	7%
$6 * 6$	Vegetation	69%	4%	14%	13%	69%	31%
	Wildfire	2%	55%	25%	18%	55%	45%
	Urban area	1%	3%	82%	14%	82%	18%
	No change	1%	0%	4%	95%	95%	5%
$7 * 7$	Vegetation	75%	5%	11%	9%	75%	25%
	Wildfire	3%	66%	21%	10%	66%	34%
	Urban area	2%	4%	84%	10%	84%	16%
	No change	1%	1%	7%	91%	91%	9%
$10 * 10$	Vegetation	78%	4%	11%	7%	78%	22%
	Wildfire	3%	77%	9%	11%	77%	23%
	Urban area	1%	3%	90%	6%	90%	10%
	No change	1%	1%	4%	94%	94%	6%

Figure 54: Confusion matris of the results

ANNEX 2: IMAGE RESULTS OF THE BLOCK CLASSIFICATION

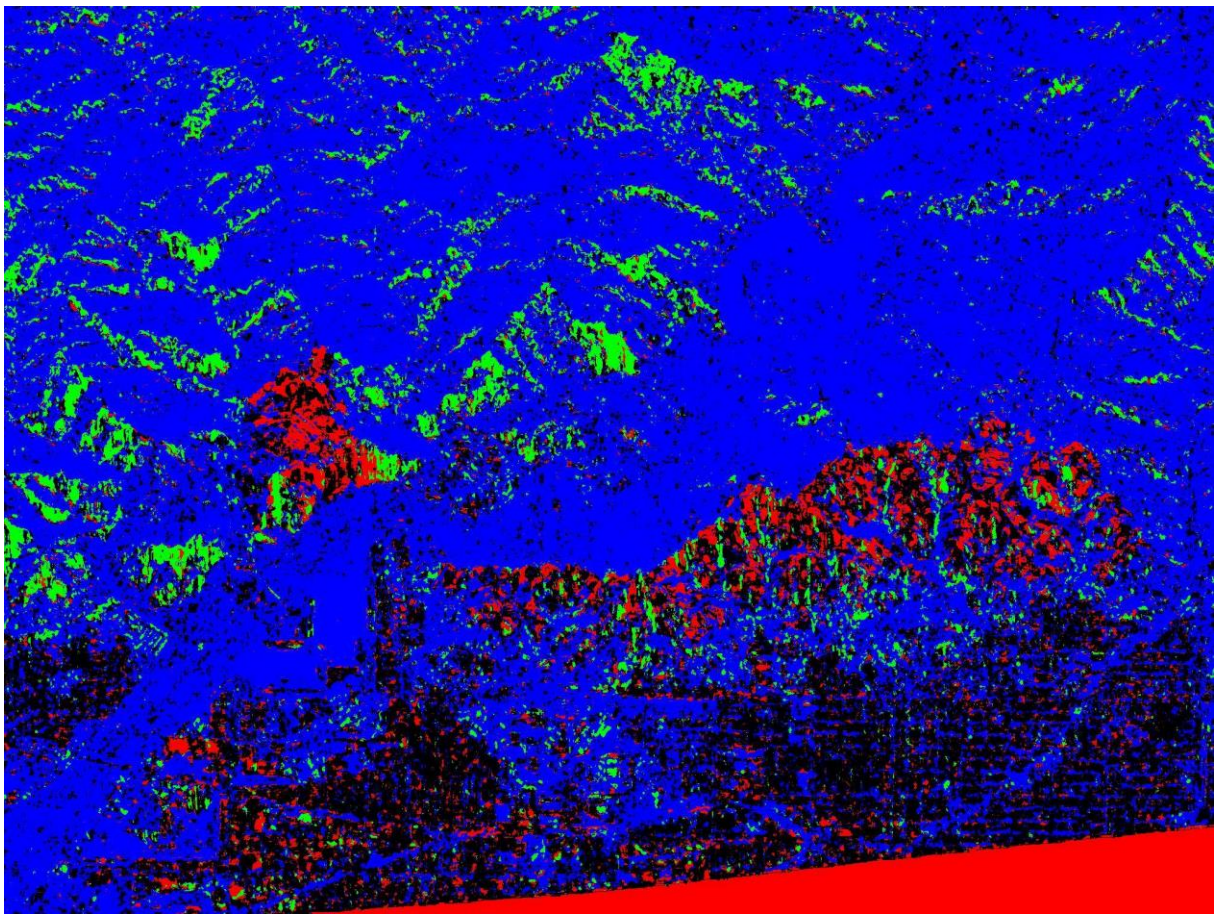


Figure 55: Result classification with blocks of $4 * 4$ *pixels and 7 pixels overlap*

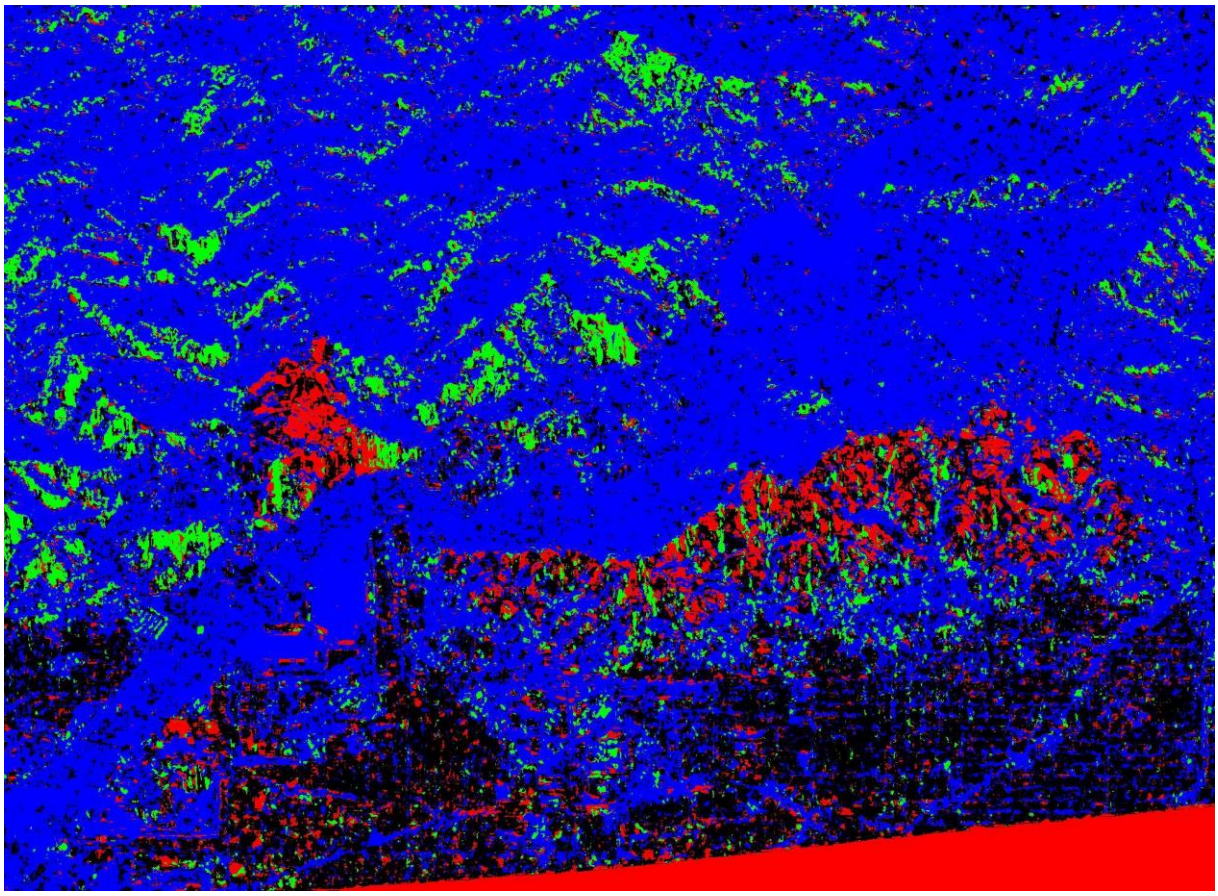


Figure 56: Result classification with blocks of 5*5 pixels *and 9 pixels overlap*

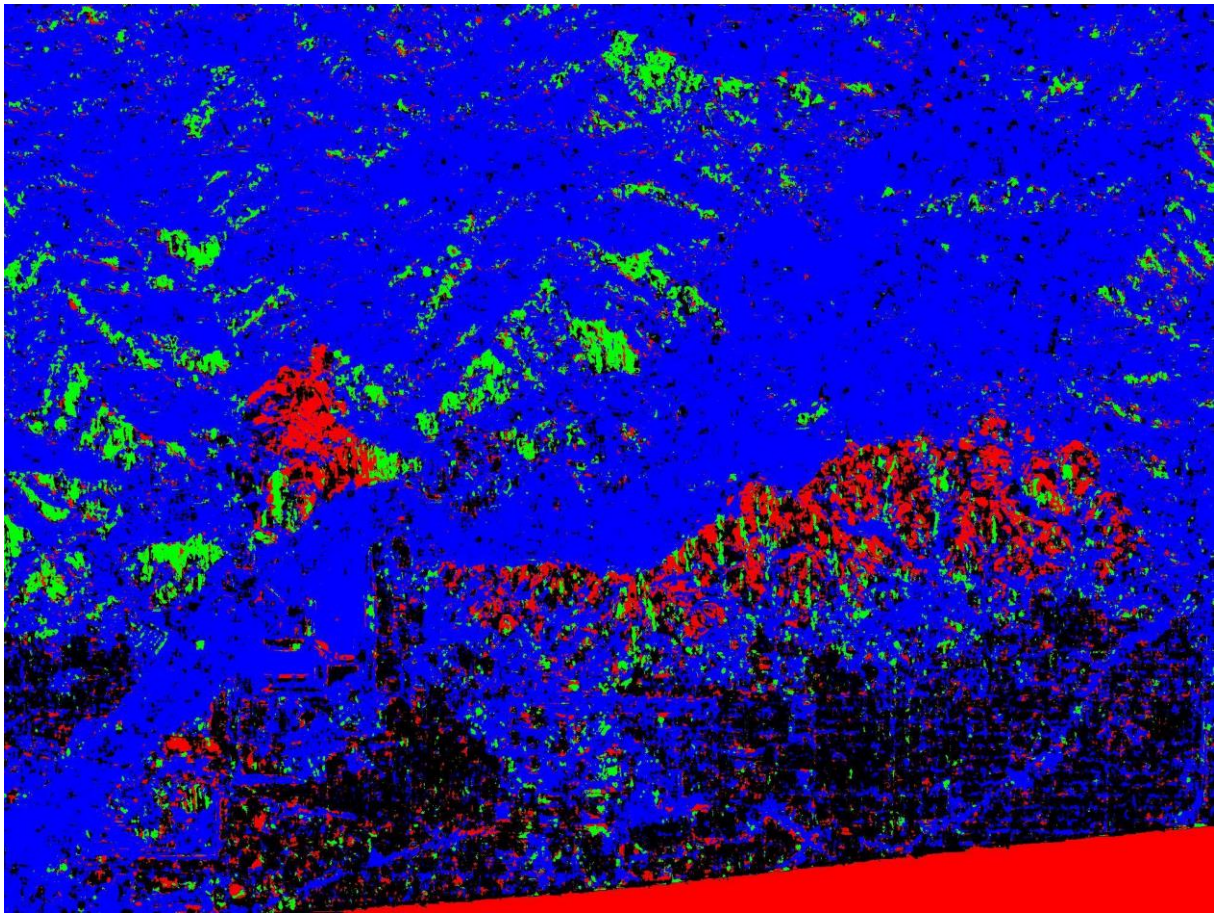


Figure 57: Result classification with blocks of $6 * 6$ pixels and 11 pixels overlap

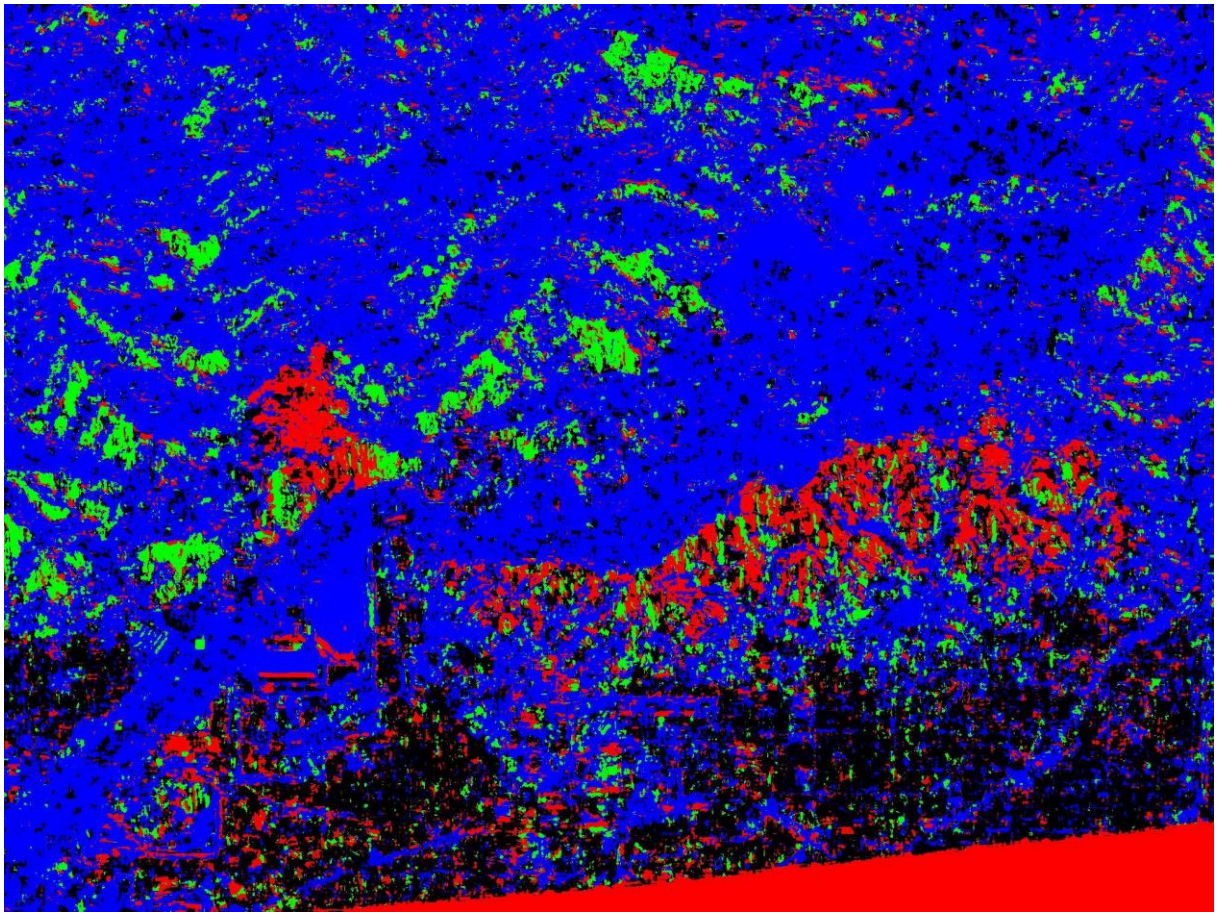


Figure 58: Result classification with blocks of $7 * 7$ pixels and 13 pixels overlap

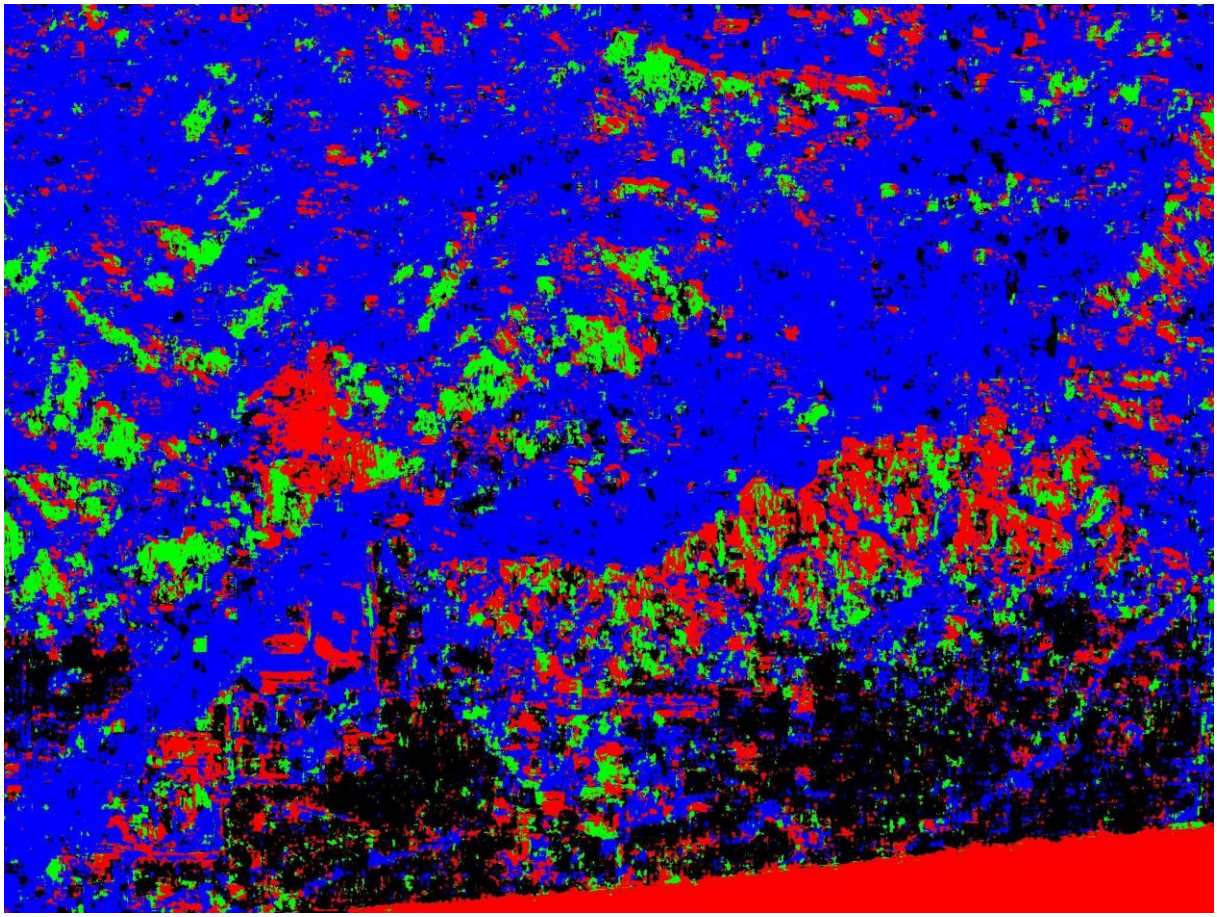


Figure 59: Result classification with blocks of $10 * 10$ *pixels and 19 pixels overlap*

LIST OF FIGURES

BIBLIOGRAPHY

Bamler, R., & Eineder, M. (n.d.). Retrieved from SAR EDU remote sensing education initiative.

D. Lu, P. M. (2003). *Change detection techniques.*

Einede, M., & Bamler, R. (n.d.). *Module 2201: SAR Interferometry Basics.* Retrieved from https://saredu.dlr.de/unit/insar_basics

Ferretti, A., Monti-Guarnieri, A., Prati, C., & Rocca, F. (February 2007). *InSAR Principles: Guidelines for SAR Interferometry Processing and Interpretation.* (K. Fletcher, Ed.) Retrieved from http://www.esa.int/About_Us/ESA_Publications/InSAR_Principles_Guidelines_for_SAR_Interferometry_Processing_and_Interpretation_br_ESA_TM-19

Gabriel Vasile, N. B. (2013). Retrieved from https://www.researchgate.net/file.PostFileLoader.html?id=58b98216ed99e1f9604c56e3&assetKey=AS%3A467837111279619%401488552467832

Hanssen, R. (2014). Retrieved from European Space Agency: http://seom.esa.int/landtraining2014/files/D2T2a_Hanssen.pdf

Javier Estornell, J. M.-G. (2013). *Principal component analysis applied toremote sensing.*

Lopez-Sanchez, J. M. (2014). Retrieved from European Space Agency: http://seom.esa.int/landtraining2014/files/D1T1b_Lopez-Sanchez.ppt

McCormick, C. (2013). Retrieved from mccormickml: http://mccormickml.com/2013/05/09/hog-person-detector-tutorial/

NASA, U. (n.d.). *What is UAVSAR?* Retrieved from National Aeronautics and Space Administration: https://uavsar.jpl.nasa.gov/education/what-is-uavsar.html

Papathanassiou, K. P. (n.d.). Retrieved from European Space Agency: http://seom.esa.int/landtraining2014/files/D1T2a-D2T1a_Papathanassiou.pdf

Pottier, E. (2017). Retrieved from European Space Agency: http://seom.esa.int/polarimetrycourse2017/files/materials/PolSAR_theory_EPottier.pdf

Powell, V. (n.d.). *setosa*. Retrieved from http://setosa.io/ev/principal-component-analysis/

Radjabi, R. F. (2016). *Wildfire detection system based on principal component analysis and image processing of remote-sensed video.*

Rosen, P. A. (2004). *Geoscience and Remote Sensing Society.* Retrieved from https://www.grss-ieee.org/wp-content/uploads/2010/06/

Thomas Jagdhuber, I. H. (2014). Retrieved from SAR EDU remote sensing education initiative: https://saredu.dlr.de/unit/polsar

Wolff, C. (n.d.). *Radar Tutorial*. Retrieved from http://www.radartutorial.eu/20.airborne/ab07.en.html

Wright, T. J. (n.d.). *European Space Agency.* Retrieved from https://earth.esa.int/documents/10174/642983/D3T2a_WRIGHT_LTC2013.pdf

ZhiQiand Chen, T. H. (2010). *Probabilistic Irban Structural Damage Classification Using Bitemporal Satellite Images.*