POLITECNICO DI TORINO

Master's degree in Computer and Communication Network Engineering

A.y. 2021/2022

April 2022 Graduation Session

Landslide monitoring using optical fiber sensing based on polarization

Supervisor
Roberto Gaudino

Co-supervisor
Giuseppe Rizzelli Martella

Candidate
Saverio Pellegrini

Abstract

The Master Thesis work has its main focus on Optical Fiber Sensing, applied to the detection of dangerous events in several types of geology monitoring, such as for debris in a mountain scenery. In particular, the monitoring system that has been built is based on the detection of vibrational events through the continuous check of the polarization state of light conveyed inside the fibers. The intensity of the stresses that happen on the fiber is strictly related to the speed with which the polarization state changes with time. The sensor proposed in this Thesis bases its working principle on this concept, in particular on the computation of the angular speed at which the Stokes vector, which conventionally represents the polarization state over the Poincaré sphere, changes along time. In the end, the purpose of this system would be to generate an alarm in correspondence of catastrophic events, in order to protect people from them. This work has been developed in collaboration with a Turinese company (Geosolving srl) and financed by a project of Piedmont Region.
All the experiments have been performed on a reduced scale model of a side of a mountain with size of approximately 3 by 0.7 meters, which is reported in Figure 1. In this way, it has been possible to generate and analyze a lot of events that would be reasonably similar to what a dangerous event on a real scale would look like. Moreover, different fiber configurations buried inside the soil of the model have been tested, changing depth with respect to the surface and disposition along the soil. This scenario gave the possibility to deeply understand the limits and the advantages of this approach, which in the end gave very interesting results.

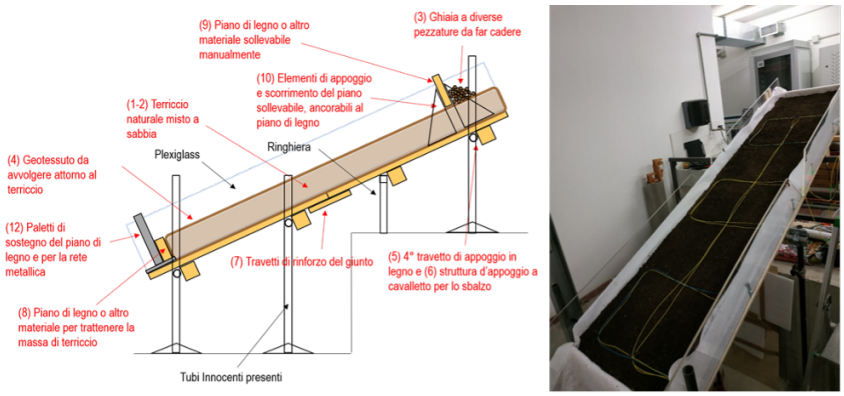

Figure 1: Characterization of the model

The first part of the Thesis reports a brief introduction to the Optical Sensing state of the art scenario, and to the description of how the experimental set up has been built, with a detailed focus on the devices used, reported in Figure 2. All the fibers connected to the devices in Figure 2 are actually buried in the soil. A description of the generation

Figure 2: Computer, Polarimeter, Laser

of events has been also provided, along with a preliminary but deep analysis of how the fibers reacted to them, and how to improve the processing of the time samples acquired, to get the best out of them.
The remaining and most important portion of this experimental Thesis has been instead dedicated to the creation and optimization through Matlab of a real time algorithm that would be able to detect in real time the occurrences of dangerous events and, in a real world scenario, potentially protect people from accessing an unsafe area, by triggering an alarm (example: switch on of a red traffic light when something anomalous is detected). It has been found experimentally that the proposed system works perfectly in the reduced scale model: the detection of some dangerous events happens correctly and in real time, with a delay of no more than a few seconds. These experiments have been possible thanks to the interaction between a proper algorithm (written in Matlab), and a Polarimeter, which is the actual sensor. In Figure 3.23 a first example of a real

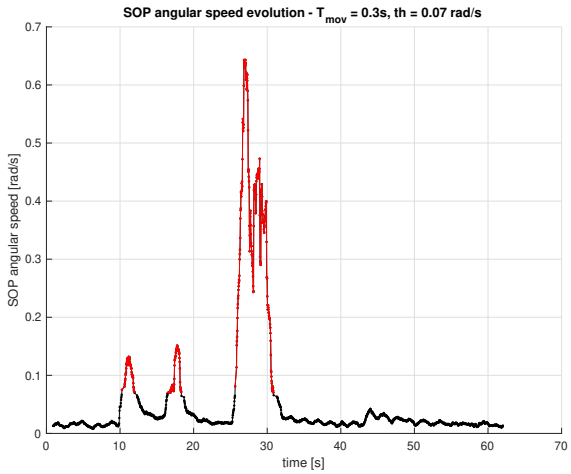

Figure 3: Real time Matlab output

time acquisition is reported, performed when three events are generated (single rock, cylinder and debris flow falling): the black part of the curve represents the values of angular speed that are under threshold, the red ones are instead above it, labeled as dangerous, and will trigger an alarm.
The only limits that have been observed are about the inability of the system to distinguish between different events, and also the fact that the current version of the monitoring system does not allow to localize the position of the vibration events along the fiber. The first issue has been actually discussed and partially solved, thanks to the application of Neural Networks that could be able to correctly classify events. A focus on the implementation of these ones in the scenario described up to now is an interesting possible follow-up of this Thesis. The second issue is on the contrary an intrinsic characteristic of the proposed approach: our system can potentially sense an anomalous vibration occurring in any position along a long fiber, but it cannot spatially locate it. On the other hand, this system is very convenient in terms of cost, since it is a lot cheaper than a distributed optical sensor, which would be able instead to localize continuosly everything that happens along the fiber.

Contents

1 Introduction **1**
- 1.1 General Introduction on Optical Fiber Sensors . . . 1
- 1.2 Introduction to polarization in Single Mode optical Fibers . . . 4
  - 1.2.1 Fundamentals on polarization . . . 4
  - 1.2.2 Stokes parameters and Poincaré sphere . . . 7
  - 1.2.3 Birefringence . . . 9
- 1.3 Goal of the Thesis . . . 11
- 1.4 Outline of the contents of the Thesis . . . 13

2 Experiments **15**
- 2.1 Experimental setup . . . 15
  - 2.1.1 Optical Source . . . 16
  - 2.1.2 Mountain downhill model . . . 17
  - 2.1.3 Polarimeter . . . 21
  - 2.1.4 Processing block and GUI . . . 23
- 2.2 First tests and results obtained . . . 24
  - 2.2.1 Tests performed . . . 25
  - 2.2.2 Preliminary analysis of the data acquired . . . 27
  - 2.2.3 First steps into SOPAS processing . . . 29
  - 2.2.4 Detection problem . . . 33
- 2.3 Spectrogram analysis, sampling frequency optimization . . . 37
- 2.4 Rockfall barrier Fiber performances . . . 45
  - 2.4.1 RP1 Configuration . . . 46
  - 2.4.2 RP2 Configuration . . . 47
  - 2.4.3 RP3 Configuration . . . 48
  - 2.4.4 RP4 Configuration . . . 49

3 Real time algorithm **51**
- 3.1 Algorithm description and working principle . . . 51
- 3.2 Real-time mode algorithm implementation . . . 58
- 3.3 Limits and improvements of the real time system . . . 60

4 Neural Networks approach **67**
4.1 Introduction to the main concepts of NN. 68
4.2 Neural Network applied on the SOPAS evolutions 70

5 Conclusions **79**

A SOPAS behaviors, Offline detection maps **82**

B SOPAS long acquisitions plots **87**

C Real time algorithm: Matlab script **89**

Bibliography **93**

Acknowledgements

I would like to deeply thank my Supervisors Roberto Gaudino and Giuseppe Rizzelli, which guided me constantly and with enthusiasm during the whole Thesis work. I would like to extend my sincere gratitude to Geosolving srl, which had a fundamental role in the developement and success of this project. Finally, I would like to express my gratitude to my family and friends, which always supported me during my studies.

Chapter 1

Introduction

This first Chapter takes into consideration some state of the art technologies used for fiber sensing, and introduces the kind of approach that has been exploited in this particular work. Also, since the kind of monitoring system that has been built is completely based on polarization, a brief recap on some basic electromagnetic concepts is given.

1.1 General Introduction on Optical Fiber Sensors

All the results achieved while working on this Thesis have been obtained through the usage of a system which acts as an Optical Fiber Sensor. It is then quite important to make a short introduction to what is the state of art of this field, by highlighting the different technologies that could be adopted, with respect to the one which has been actually chosen in my research work.
By definition, an Optical Fiber Sensor is a device that uses light to convey the informations which it senses. In general the whole sensing system is made of three main different parts:

- **Interrogator:** composed by an Optical source, which is emitting light to be sent inside the fiber, plus a properly designed receiver. Light will act as a probe, meaning that it will be modulated by the sensor in accordance with the value measured by it. This first block includes also a Receiver and a Processing block, which will basically first convert the optical signal received back from the sensor into an electrical one, and then process the data.

- **Optical transit cable:** this is where the probe light is conveyed, and where the modulated one is returned back to the Interrogator. Usually it is a Single Mode Fiber plus, in some situations, fiber Bragg grating.

- **Remote sensor:** this block will respond to just certain attributes of the light at its input, and send back a modulated optical signal, according to the value measured.

Figure 1.1 reports graphically the different parts described in the bullets above. Actually also a Patch panel is reported: this is just the connection point from the Optical source to the Transit cable, and from the Transit cable to the Receiver. This kind of general structure of an Optical Fiber Sensing system is very important to keep in mind, since it will be very similar to the architecture that will be adopted for the actual tests. A

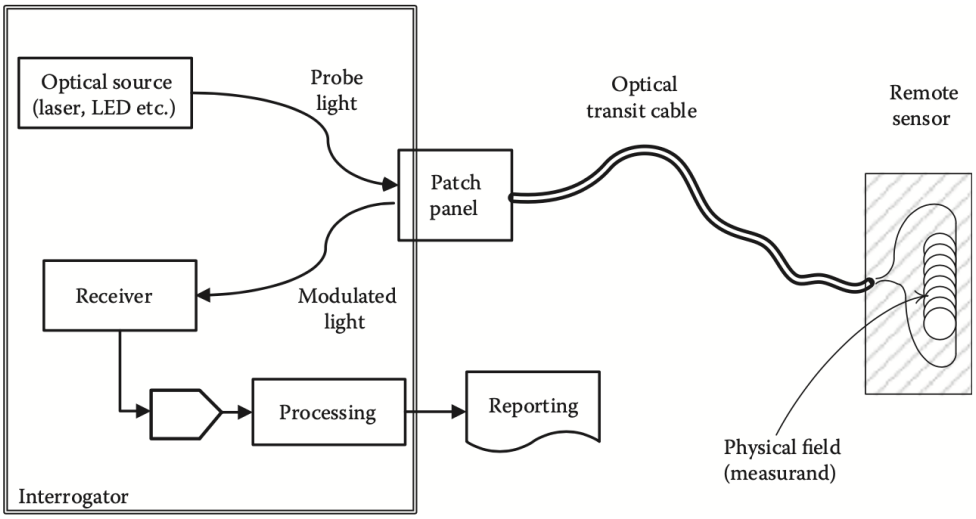

Figure 1.1: General Optical Fiber Sensing architecture (Picture taken from [5])

classification of some different possible topologies also exists, based on the general structure reported in Fig. 1.1. This is very useful to understand the limits and the advantages of the topology that has been adopted for the monitoring system described in this work, and it is reported in Figure 1.2. Three different categories can be identified:

- Single point sensor (Fig. 1.2 (a)): this is a very simple topology, which works exactly as the one described in Figure 1.1. The main disadvantage is that sensing is happening at no specific location along the fiber. To make it very simple, if some kind of event that the sensor is meant to detect happens, assuming that the detection will happen correctly, it is impossible to localize it along the fiber.

- Multiplexed point sensor array (Fig. 1.2 (b, c)): in this case, although the idea behind the ladder topology and the reflective topology is different, the final concept is that the presence of different sensors allows a better mapping, since it is able to provide feedbacks in different and distributed spots of the space. The only problem with this approach is that the sensors are placed in a discrete way, and if the shape of the spatial profile is not known, is very difficult to place the different sensors a priori in an optimal way, granting good performances.

- Distributed optical fiber sensor (Fig. 1.2 (d)): this last topology is the best one,

since it is allows to have a continuous mapping along the spatial profile, with no need to place sensors in a discrete way. With this topology, every point of the fiber contributes to the output. Anyway these solutions usually require more expensive interrogators compared to the previous ones.

Gradually, the actual system adopted for this Thesis will be introduced, with an important reference to 1.1, along with the discussion behind the reasons that lead to the choice of a topology with respect to another. Given the large number of attributes of light that can be

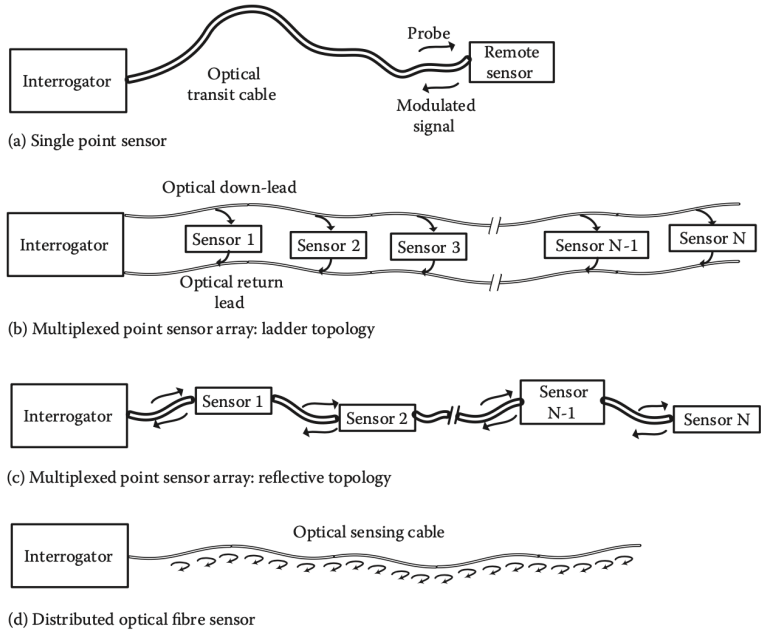

Figure 1.2: Some different sensing topologies (Picture taken from [5])

measured (intensity, polarization, phase, propagation time, optical spectrum...), Optical fiber sensing is extremely versatile, suitable for a lot of different applications going from the mechanical ones (strains, vibrations, or generic mechanical stresses detection) to temperature and chemical ones, to name a very few. In this specific scenario, the goal of my research is to build a sensing system based on polarization, which is one of the different attributes of light, to detect mechanical events happening on the fiber, by means of a software (Matlab) algorithm. In particular this approach is thought to be be applied to a mountain scenery, where dangerous events, potentially jeopardizing lives of many people, could happen. Since usually they are very intense and generate mechanical stresses and vibrations, fiber sensing based on polarization could be a truly optimal way to detect them. It follows that, making it very simple, the basic idea is to have a system made of one Laser, one Polarimeter and one fiber connecting the two edges. The Polarimeter would basically collect all the States Of Polarization (SOP) of the input light and convert them into a sampled electrical signal, which would then be

processed by a computer connected to it (the very specific experimental set up with all the details will be described in Chapter 2, with all the differences and similarities with respect to what has been introduced above in this Section).

1.2 Introduction to polarization in Single Mode optical Fibers

In order to build a system like the one described in Section 1.1, a quite solid understanding of polarization in Single Mode Fibers (SMF) is fundamental.

1.2.1 Fundamentals on polarization

Polarization of waves is describing the orientation of the field vectors in the individual and resultant waves [13]. Let's consider the phasor notation in Equation 1.1 for the Electric field ($\vec{E}$) in a SMF.

$$\vec{E} = (\hat{x}E_1 + \hat{y}E_2 e^{j\phi})e^{-jkz} \tag{1.1}$$

and consequently for the Magnetic field ($\vec{H}$) in Equation 1.2

$$\vec{H} = \frac{1}{\eta}(-\hat{x}E_2 e^{j\phi} + \hat{y}E_1)e^{-jkz} \tag{1.2}$$

where k is the wavenumber, η is the intrinsic impedance of the medium, ω is the angular frequency and E_1, E_2 are the amplitudes of the Electric field components, which will be identified as E_x and E_y. From basic electromagnetic theory it is true that $\vec{E} \perp \vec{H}$ so, conventionally, polarization is referred to the orientation of the Electric field only, being the Magnetic one always perpendicular to that. The parameter ϕ is extremely important, since it represents the relative phase shift between the two components of $\vec{E}$. Depending on this value and on the amplitudes of the two components E_1 and E_2, three different kind of polarizations exist:

1. Linear polarization.
2. Circular polarization.
3. Elliptical polarization.

Let's start from the first one, Linear polarization. Referring to 1.1, an electromagnetic wave is Linearly polarized if the the condition $\phi = 0$ is true, meaning that E_x and E_y are perfectly in phase. Equation 1.1 becomes

$$\vec{E} = (\hat{x}E_1 + \hat{y}E_2)e^{-jkz} \tag{1.3}$$

Graphically, the situation is described in Figure 1.3. By analysing this one and Equation

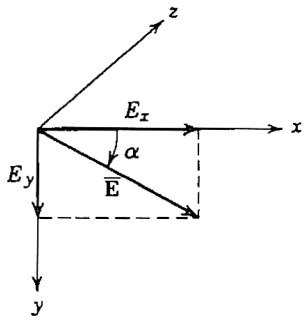

Figure 1.3: Linear polarization (picture taken from [13])

1.3, it is clear that the direction of the resulting $\vec{E}$ vector is not changing in space during propagation, and while propagating it defines a plane: the angle α reported is constant over z. For this reason this kind of polarization is called Linear.
For what concerns instead Circular polarization, it is obtained when $E_1 = E_2$ and the two E_x and E_y components are in quadrature, so $\phi = \pm\,\pi/2$. In this specific case, what happens is that Equation 1.1 becomes

$$\vec{E} = (\hat{x} \pm j\hat{y})E_1 e^{-jkz} \tag{1.4}$$

which, separating the instantaneous forms of the x and y components of the Electric field, can be written as:

$$E_y(z,t) = \pm E_1 \sin(\omega t - kz) \tag{1.5}$$

and

$$E_x(z,t) = \pm E_1 \cos(\omega t - kz) \tag{1.6}$$

This is very interesting: angle α was absolutely constant for Linear polarization, as already highlighted, and equal to

$$\alpha = \tan^{-1}(\frac{E_y}{E_x}) = \tan^{-1}(\frac{E_2}{E_1}) \tag{1.7}$$

where the components of the Electric field are in phase, so not shifted one with respect to the other: the istantaneous forms of the two components differed only for the amplitudes. For Circular polarization, these components are now instead in quadrature. This changes the situation, since they can be described as two oscillating sinusoidal functions having a phase shift of $\pi/2$, so basically as a sine and cosine. By rewriting Equation 1.7 for the Circular case, Eq. 1.8 is obtained

$$\alpha = \tan^{-1}\left(\pm\frac{\sin(\omega t - kz)}{\cos(\omega t - kz)}\right) = \pm(\omega t - kz) \tag{1.8}$$

This is telling that, on the contrary of Linear polarization, α is not constant with time, but it rotates with angular speed equal to $\pm\omega t$. As a consequence, by looking at Figure 1.3, the

angle α is actually increasing (or decreasing, depending on the direction) with constant speed, so the resulting $\vec{E}$ vector is rotating on the z plane, describing a circumference and explaining the name of this second kind of polarization. A graphical representation of what happens in the direction of propagation is reported in Figure 1.4.

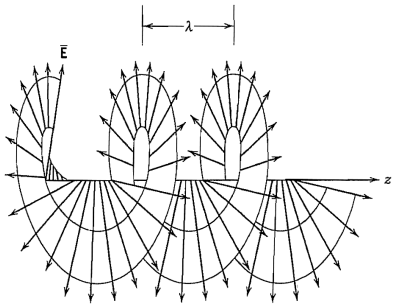

Figure 1.4: Circular polarization (picture taken from [13])

Elliptic polarization is instead the most general case that can be considered: the condition with which Elliptic polarization is obtained is if $E_x \neq E_y$ and ϕ is any value but 0 or $\pm\pi/2$ (excluded because they would coincide with the Linear or Circular case). Being this one the most generic situation, it "contains" also the other two kinds of polarizations, which can be retrieved if considering the conditions described above, that for this case will be avoided, to prevent making this case collapse in one of the other two. By referring, as always, to the instantaneous forms of Eq. 1.1, which represent the most general situation, the following is obtained

$$E_x(z,t) = E_1 \cos(\omega t) \tag{1.9}$$

and

$$E_y(z,t) = E_2 \cos(\omega t + \phi) \tag{1.10}$$

In this case, α is not rotating with constant speed, nor it is constant, because the two components of $\vec{E}$ are not in quadrature. This means that, being the phase shift ϕ between

(a) Linear, π (b) Linear, $\pi/4$ (c) Circular (d) Elliptical

Figure 1.5: Different kind of polarizations seen up to now

the two component generic as their amplitudes, the geometric shape described on the

xy plane will be an ellipse. This last kind of polarization is very important because it is the most generic one and allows to describe any polarization state. As a recap of what has been said said up to now, in Figure 1.5 the different geometries described by the kinds of polarizations seen up to now have been reported.

1.2.2 Stokes parameters and Poincaré sphere

Another way to easily describe the different kinds of polarizations of an electromagnetic wave is by means of some important parameters called Stokes parameters, which will be the true core of all the subsequent work. Before talking about them, an important indicator called Deegree Of Polarization (DOP, p in the Equation below) needs to be introduced. This is defined as in Eq. 1.11 and accounts for "how much" an electromagnetic wave is polarized or not.

$$p = \frac{\sqrt{S_1^2 + S_2^2 + S_3^2}}{S_0} \tag{1.11}$$

S_0, S_1, S_2, S_3 are the Stokes parameters. Varying p, three cases need to be considered:

- $p = 1$, Completely polarized light.
- $0 < p < 1$, Partially polarized light.
- $p = 0$, Unpolarized light.

In the case of completely polarized light, elliptical polarization can be defined as follows:

$$\begin{cases} S_0 = |E_1|^2 + |E_2|^2 \\ S_1 = |E_1|^2 - |E_2|^2 \\ S_2 = 2E_1E_2\cos(\phi) \\ S_3 = 2E_1E_2\cos(\phi) \end{cases} \tag{1.12}$$

Like usual, by changing amplitudes and ϕ values in Eq. 1.12 as explained in the previous Section, the description of Linear and Circular polarization can be found in terms of these parameters. In the case instead of partially polarized light p is not equal to 1, it then needs to be taken into account. For this reason a more general expression needs to be introduced, in Eq. 1.13.

$$\begin{cases} S_0 = I \\ S_1 = Ip\cos(2\psi)\cos(2\chi) \\ S_2 = Ip\sin(2\psi)\cos(2\chi) \\ S_3 = Ip\sin(2\chi) \end{cases} \tag{1.13}$$

This case refers to a much more generic condition, where light is a mixture of completely polarized and unpolarized light, and cannot be described by means of 1.12, which is

a different situation. The arguments of the sines and cosines inside 1.13 (ψ and χ), refer to the ellipse in Figure 1.6 Basically, every polarization state can be described by

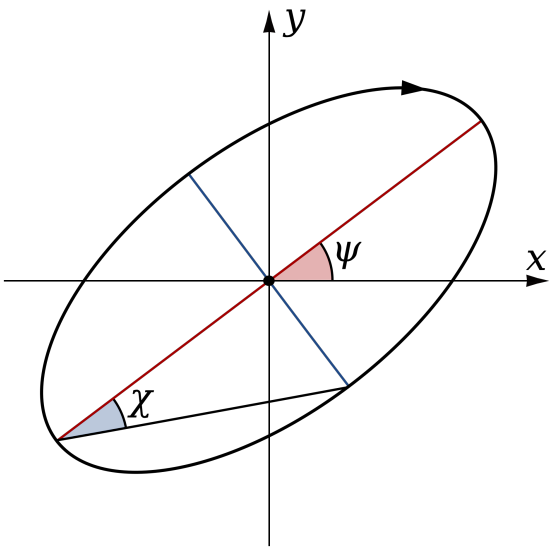

Figure 1.6: Reference ellipse for Stokes parameters definition

the above four parameters by simply substituting the right values. The parameter I is instead representing the intensity. What is very important to take into consideration is the relationship that exists between these four parameters, which is the following, in Equation 1.14, derived from Eq. 1.11.

$$S_0 p = \sqrt{S_1^2 + S_2^2 + S_3^2} \tag{1.14}$$

This means that, if considering a three dimensional space, neglecting S_0 which is just not angles dependent, one vector could be represented, written as:

$$\vec{S} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} \tag{1.15}$$

and since the relationship between the three components is the one in 1.14, it's possible to say that the vector $\vec{S}$ is actually describing a sphere of radius $I \cdot p$ in a three dimensional space. The sphere that comes out is called Poincaré sphere, in Figure 1.7. Every point on the surface of this sphere is defined by a different value of ψ, χ, I, and p, which at this point are defining a different State Of Polarizations of the wave. The fact that one single SOP could be represented by a vector having its tail on the origin of the sphere and its edge on the surface (green vector in Figure 1.7), means that if for some reason the SOP of the wave changes, the vector will move, and a different point of the

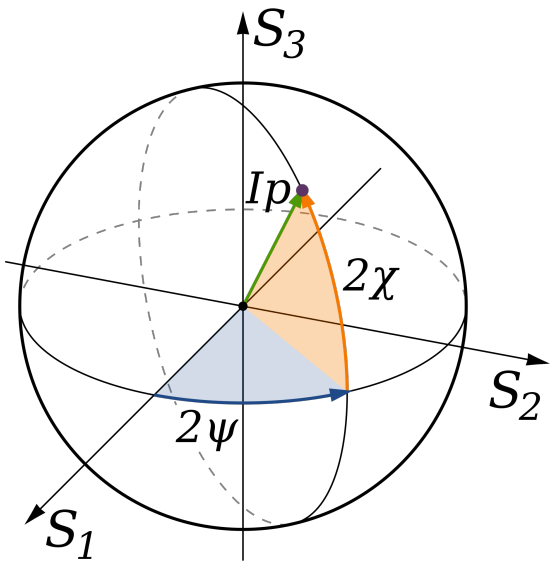

Figure 1.7: Poincaré sphere

sphere will be "pointed". This concept will be of extreme importance for the following discussions on this Thesis, in particular to define the SOP Angular Speed, which is the speed at which the vector $\vec{S}$ moves when the SOP changes over time.
Unpolarized light can instead be obtained in terms of Stokes parameters by just substituting in 1.13 $p = 0$, obtaining the expression in Eq. 1.16.

$$\begin{cases} S_0 = I \\ S_1 = 0 \\ S_2 = 0 \\ S_3 = 0 \end{cases} \tag{1.16}$$

For the purposes of this Thesis the Poincaré sphere is of fundamental importance, so the most generic case needs to be considered: all the subsequent work will refer to the definition in Eq. 1.13.

1.2.3 Birefringence

Birefringence is a phenomenon appearing inside real SMFs, which has strong consequences on polarization. Before focusing on the effects it has on this one, let's consider the concept behind, by first of all defining the notion of Weakly Guiding Fibers [12], [7]. If the ratio between the fiber core and cladding refractive index (n_{co} and n_{cl}, respectively) respects the condition in Eq. 1.17, inside a generic kind of fiber,

$$\frac{n_{co}}{n_{cl}} \simeq 1 \tag{1.17}$$

it appears that some propagation modes have basically the same cut-off frequency, and for this reason are called degenerate: they can be merged together as a single one, being their differences negligible, and some new modes are generated, called Linearly Polarized ($LP_{l,m}$) since it can be shown that the combination of these degenerate modes gives linearly polarized fields. This condition defines the set of fibers called Weakly Guiding Fibers, to which the SMF belongs. Despite this, the SMF is conceived and built in a way such that the fundamental mode, which is the LP_{01}, is propagating exclusively. The very important fact about the LP modes, and this is the key point to understand the concept of birefringence, is that they are also degenerate in polarization, meaning that each one of them exists in two linear polarizations, $\hat{x}$ and $\hat{y}$, propagating along the fiber. This means that considering an SMF in which only LP_{01} can be identified, two degenerate (in polarization) modes are actually propagating, one along x and the other along y direction, both of them seeing two different n_x and n_y refractive indexes. Now, inside an SMF, if ideal conditions are considered, the two orthogonal modes LP_{01x} and LP_{01y} are actually propagating in a truly degenerate way, so experiencing the exact same refractive index on both axes: this means that, in the ideal case, $n_x = n_y$. The consequence of this theoretical behavior is that the two orthogonal modes are nominally propagating at the same speed, there is not one faster than the other, and no consequences on the SOP are appearing. In a much more realistic case anyway, fibers are subjected to a lot of different stresses, starting from the most common bends up to the non perfectly circular symmetry of the fiber core, all leading to a less strong degeneracy of the orthogonal modes which are now seeing an effective refractive index which is different for both of them, so $n_x \neq n_y$. The birefringence can now be defined as

$$\Delta\beta = \left|\beta_x - \beta_y\right| = \frac{2\pi}{\lambda}\left|n_x - n_y\right| \tag{1.18}$$

where the parameters β_x and β_y are the propagation constants referred to the two orthogonal modes. This means that, if assuming that the birefringence is constant along the whole fiber of length L, the phase difference experienced by the two modes is

$$\Delta\theta = \Delta\beta L \tag{1.19}$$

which would mean that the two modes are propagating with two different group velocities, and consequently experiencing a different group delay $\Delta\tau$ between them: a fast mode and a slow mode will appear, as reported in Figure 1.8. This effect on the phase has a huge impact on the polarization state. If a beam of light with a certain SOP is launched in a SMF, it can be certainly decomposed into two different contributions, one on the x axis, the other on the y axis. Now, since the effect of birefringence appears on the phase shift between the two components (see Equation 1.19), it is very clear, now that it has been explained how polarization works, that it modifies a lot the SOP of the light flowing inside the fiber, as it propagates. This means that the State Of Polarization, in real "quiet" conditions, is absolutely not constant over the propagation direction z, due the the phase shift in Eq. 1.19, but it is actually contant over time. If some outer

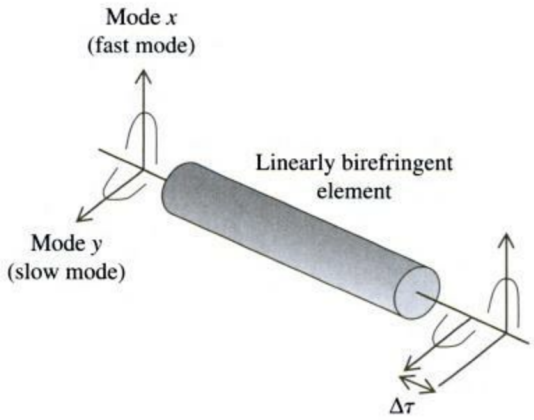

Figure 1.8: Group delay introduced by a birefringent element (picture taken from [3])

bends or more general mechanical stresses and vibrations happen over time, the effect of birefringence is much more intense: physical changes on the fiber appear, $\Delta\beta$ changes over time, and consequently $\Delta\theta$. The SOP change is stronger, and not only along the propagation direction, but also over time, due to the external stresses. Birefringence is basically the concept around which the adopted system is based.

1.3 Goal of the Thesis

The mountain scenario is very often inclined to be subject of dangerous events, which many times have created unsafe conditions and jeopardized the lives of many people. Events such as landslides, debris flows or rocks falling represent a true danger, which sometimes cannot be predicted and, consequently, avoided. What was pictured when thinking about this kind of unsafe conditions is reported in Figure 1.9, where something similar to a debris or a landslide coming from the side of a mountain has fallen on the road, making it not accessible and also very dangerous. It would then be very important

Figure 1.9: Example of a dangerous landslide happening on a road (picture taken from [1])

to know in advance if anything of this kind is about to happen, since it could not only damage and block a street, but also trample cars or civilians in the worst case. The goal of this Thesis is based exactly on this concept: try to give some sort of alarm (acoustic, visual) when a dangerous situation is sensed. In this way, for example, roads could be blocked a few meters early with respect to the dangerous area (by means of a traffic light, for example), and maybe civilians could be stopped from accessing an area that would be, after the alarm, labeled as unsafe. To pursue this goal, Optical Sensing based on Polarization, which has been introduced in this very introductory first Chapter, could be very suitable since, for what has been said about polarization and birefringence, fibers should really perceive mechanical stresses happening on them. In a mountain scenario, these would be placed under the soil and potentially sense vibrations generated by events, which, based on some parameters, would then be characterized as dangerous or not. A system like this would be a much simpler and cheaper solution with respect to more costly ones involving distributed sensing [2], for example: building an alternative to these kinds of more complex technology would be a very nice result, and another goal of this work. Our investigation will also consider rockfall barriers, in Figure 1.10, which

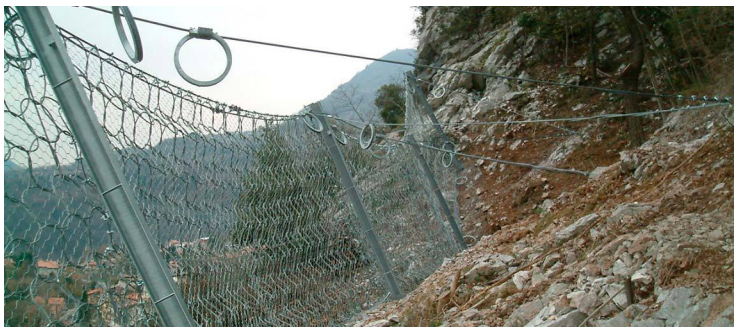

Figure 1.10: Example of a rockfall barrier (picture taken from [4])

are there to protect areas from landslides and similar events: if some fiber is wrapped around them, they could be used to monitor dangerous conditions happening over them, contributing to the monitoring and alarm system.
The very big problem of analysing these kinds of scenarios is of course the scale. It is not feasible to generate dangerous events in a real scale mountain downhill with the purpose of just analysing what happens, moreover fiber installation would be quite a problem, if the only purpose is to make tests. For this reason, an in scale model has been built, where very reliable in scale events have been generated and studied. Through this, it has been possible to develop the final alarm system with extremely thrustworthy results, which would be also valid for a real scale scenario.

1.4 Outline of the contents of the Thesis

The following Chapters will be dedicated to the actual analysis and experiments done. The workflow has been defined by the goals that, step by step, needed to be achieved, in order to see if the system that we had in mind could be working properly and, in this case, evaluate how, with all the limits and advantages. This Outline is important since it gives an idea of all the steps and goals followed, to reach the final result.

- **Chapter 2:** *Experiments.* here a very preliminary analysis has been given, explaining both the laboratory scenario where all the experiments have been carried, and the first results found by processing the data acquired. The aim of this Chapter is to make clear that the system conceived could actually work giving already very nice results, even with not optimized parameters. Once this has been understood, some ways to make the processing algorithm better and well optimized have been exposed and applied.
  - **Section 2.1:** description of how the experimental setup has been built, with all its different components.
  - **Section 2.2:** overview of the kinds of tests performed, in particular the in scale events generated on the model are described in the very details. The results obtained are shown to demonstrate that a system based on the polarization working principle to detect hazardous events could work for real.
  - **Section 2.3:** once having shown promising outcomes in the previous Section, a further characterization of the simulated events is given on the time/frequency domain. The purpose of this analysis would be to see if some further peculiarities could be found, opening up even more possibilities on the detection algorithm. Unfortunately not very interesting results came up, but this investigation has been of huge help to optimize some hardware parameters, which improved a lot the algorithm used up to now. This represents one of the most important goal of this work.
  - **Section 2.4:** up to this point the only kinds of fibers analyzed are the one buried in the soil of the model. Performances of a different kind of configuration placed over an in scale rockfall barrier are investigated. The goal of this Section is to evaluate a different way to sense events, with respect to the buried fibers one.
- **Chapter 3:** *Real time algorithm.* The algorithm version used in the latter Chapter gave interesting and promising results, but it was a post-processing version still, applied on some already acquired data. In this Chapter a fundamental step is described, the transition to an actual real time version of the system, able to give an alarm when a dangerous event happens (actually some splits of a second later), while at the same time continuously acquiring samples.

  - **Section 3.1:** a description of the working principle of the algorithm applied for a real time monitoring is given. Actually, in this Section this has been investigated as a post-processing algorithm still, simulating some sort of real time behavior, as will be described in detail. The reason behind this, is that to make the algorithm run in real time, it needed to be changed a bit with respect to the one analyzed in Chapter 2, mantaining of course the same working principle. A preliminary analysis to see if the performances of this one were equivalent to the previous one is then very important.
  - **Section 3.2:** the peculiarities of the real time monitoring system applied on the model have been here exposed, along with results from the real time application of the monitoring system on the in scale model. The goal is to show its correct functioning and also the accuracy of all the previous analysis.
  - **Section 3.3:** overview of some important problems of the system built. It is shown how the algorithm works perfectly, but would have a quite low accuracy if implemented on an actual mountain scenario. The main problem is the high weakness on the kind of noise that is not just the one generated by the fiber, but some external, spurious one: this would very easily confused with an actual dangerous event and create False Alarms. Some solutions are reported to overcome these limits, but no good ones have been found.

- **Chapter 4:** *Neural Networks approach.* The Neural Networks are a quite cutting-edge technology which could be of big help on solving the weaknesses reported in the previous Chapter. The purpose of this one is to show the general outline of this tecnique, and see how it can be applied on the system proposed, to maybe solve or mitigate the issues that came up.

  - **Section 4.1:** very general introduction to the concepts of Neural Networks, with some keywords and key concepts that will be very important to understand the analysis and tests done in the following Section.
  - **Section 4.2:** this reports the actual results of using this kind of technology. The goal here is to show how and if this approach can work in pair with the threshold algirithm exposed in the previous Chapters. Good and promising results are obtained, but no definitive ones: this analysis is treated as a possible follow-up of this whole work.

- **Chapter 5:** *Conclusions.* Main results, limits and possible open scenarios that are really worth of further investigation are here reported and summarized. A possible final scheme of an alarm system based also on the used of Neural Networks is also given, as truly final conclusion of this work.

Chapter 2

Experiments

Now that the key points to understand what the true core of the Thesis is about have been discussed, a detailed description of the experimental setup that has been used can be provided, along with all the preliminary results that have been obtained prior to get to an actual detection algorithm, able to tell if a dangerous event is actually happening.

2.1 Experimental setup

As can be guessed from the very introductory description of the key points of the adopted sensing system, given in Chapter 1, the kind of sensing which has been implemented is the one in Figure 1.2 (a), the Single point. The main reason behind this choice can be found on the working principle of polarization, explained above. Exploiting this kind of attribute of light, it is impossible to obtain sensing in space in a distributed fashion, unless adopting a Multiplexed scheme, which still would make it discrete. This one is however nonsense to be applied in this case, since the experiments and the whole analysis are carried on over a very small reduced scale model of a downhill of a mountain. Applying the Multiplexed scheme in this scenario would mean to have different sensors (each one having a quite high cost, which is not to neglect), distributed over three meters length, which would be quite useless. If the model was a bigger one, or instead the analysis was carried over a real mountain, this approach would maybe be smarter, even if it still would have problems, as pointed out in Section 1.1, but this is not the case. Distributed sensing has been excluded right away, since it has a very specific and completely diverse working principle. Also, systems based on these kind of technology would be very costly, while one of the goals of this work is to try to build a much cheaper one. The actual scheme of the setup is reported in Figure 2.1. The red edges represent the fiber connections, the black one represents an USB connection, while the light blue one represents an exchange of informations inside a PC. It's clear that, although the blocks refer to the scheme in Figure 1.1, their arrangement is not really equivalent. The main difference is that the fiber is carrying the signal only in

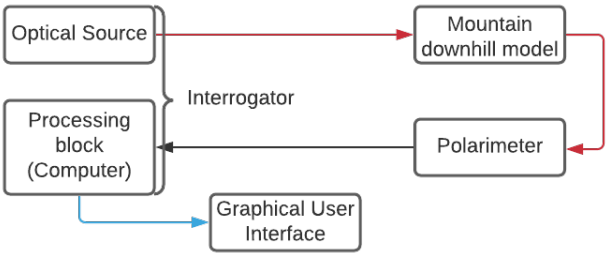

Figure 2.1: Experimental setup

one direction: from the Optical Source to the Polarimeter, and through an USB cable back to the interrogator. There is no modulation of the signal on the side of the Remote sensor, which in this case is the Polarimeter. What happens is basically that the Laser is conveying the probe light inside the fiber, which is placed inside the miniature model. Some events are happening here, which will be furtherly discussed, generating stresses on the fiber and, due to birefringence, leading to a change on the SOP of light. This will be sensed by the Polarimeter that will translate it in an electrical signal to be sent to the computer, on which processing will be executed. The results will be then showed on a screen (Graphical User Interface, GUI), connected to it. Let's now, for sake of clarity, break down the above model, explaining how every single block is made.

2.1.1 Optical Source

The optical source is a Laser emitting light at 1550 nm, the hardware used is reported in Figure 2.2. The light is then conveyed inside a Single Mode Fiber (SMF), which will

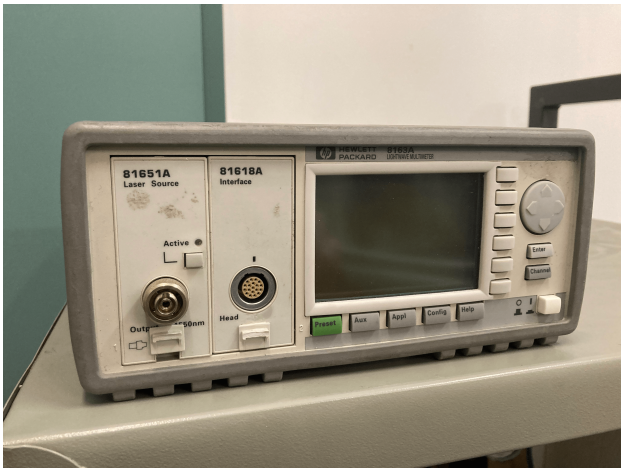

Figure 2.2: Laser used during the experiments

be inserted below the soil inside the downhill model, and act as a light probe. This instrument is placed with the computer and the Polarimeter on a table (which has been called "setup table"), where all the connections start and end. The fibers are coming out from the Laser output, and "connecting" to the slide as in Figures 2.3, 2.4. It is of course not an actual connection, they are just entering the slide from one side to get inside the soil, do some curves, and come back from the same entrance hole from which they got into, to connect to the Polarimeter. In the end, all the fibers will come out from the setup table and return back to it.

Figure 2.3: Point of view from the Laser to the slide

Figure 2.4: Point of view from the back of the slide

2.1.2 Mountain downhill model

This part of the whole set up is the most important one. The aim of this model is to reproduce in a reduced scale a side of a hill or mountain, where dangerous events could actually happen. By exploiting this very small (compared to an actual mountain) but reliable model, it was possible to reproduce some events and see how some fibers, buried inside it, reacted to them. So, first thing among all, a description of how the fibers have been placed inside the soil is needed, along with the conventions used: let's refer, in order to give a proper characterization, to Figure 2.5. In this Figure it is possible to see a first brief description of this kind of set up. In particular, the structure holding the soil is 3 meters long and 0.7 meters large, and it is inclined of about 30 degrees with respect to the horizontal plane. All the structure is made of wood, apart from the poles supporting

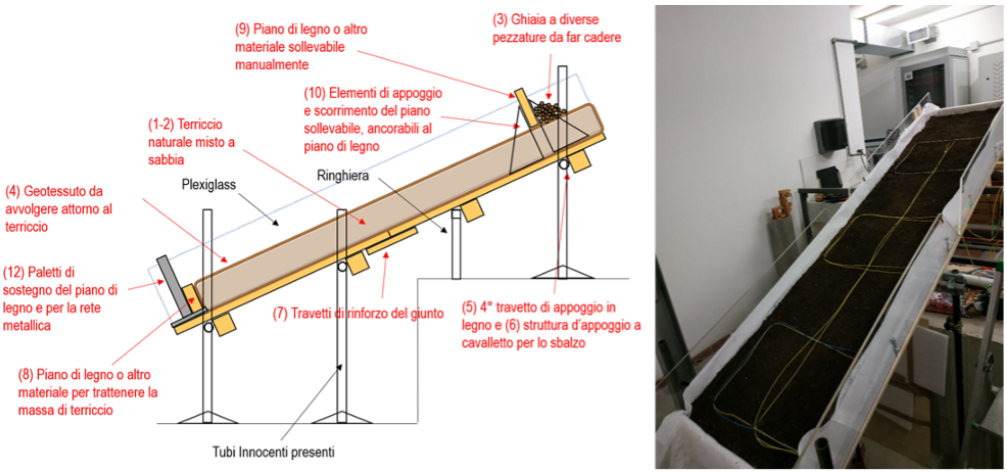

Figure 2.5: Characterization of the slide

it, which are made of metal. This is all filled by a mixture of soil and sand, to make the downhill the most reliable possible. On the left part of Fig. 2.5 it's possible to find all the details marked up to now, in a more schematic way, while on the right part of Fig. 2.5, a photo of the actual model is reported, to give a more concrete idea. Actually, the final version on which the experiments have been performed is not "naked" as in Figure, but it has geotextile covering it, which is an artificial material composed by synthetic fibres, very resistent but also very soft, often used for geotechnical applications. It has been tested that it does not modify at all the performances of the fibers, which remain the same with or without it, but it is more comfortable for a serie of several tests, since it avoids to create irregularities on the soil after each experiment, that would need to be corrected each time. On the photo on the right it's also possible to see some fibers exposed: there are not all of them, just some, and more importantly they are not that exposed but buried under some centimeters of soil. A very detailed description of the displacements of the fibers is in Figure 2.6, where they can be all recognized by means of the convention used in Table 2.1. A total of 9 fibers, differentiated by displacement and bends, is buried inside the soil. The aim of placing the fibers like described in Table 2.1, is to test and find a configuration which is able to react in a good way to the solicitations triggered through the experiments and, more in general, to observe SOP variations in different configurations. Let's look deeper at each set of them. The T fibers are the ones in Fig. 2.6 on the second plot from the left: for the T fibers, very good performances are expected, since their arrangement as a serpentine all along the soil should give a very high sensitivity and robustness against the spatial and temporal generation of the events, or where they would be the most intense. The third plot from the left in Figure shows instead the disposition of the Uc and Ul fibers: the Ul is the most lateral coil of fiber ("l" stays for lateral), while the Uc is the most central one ("c" stays for central). Their arrangement is then quite space depending, if an event is happening

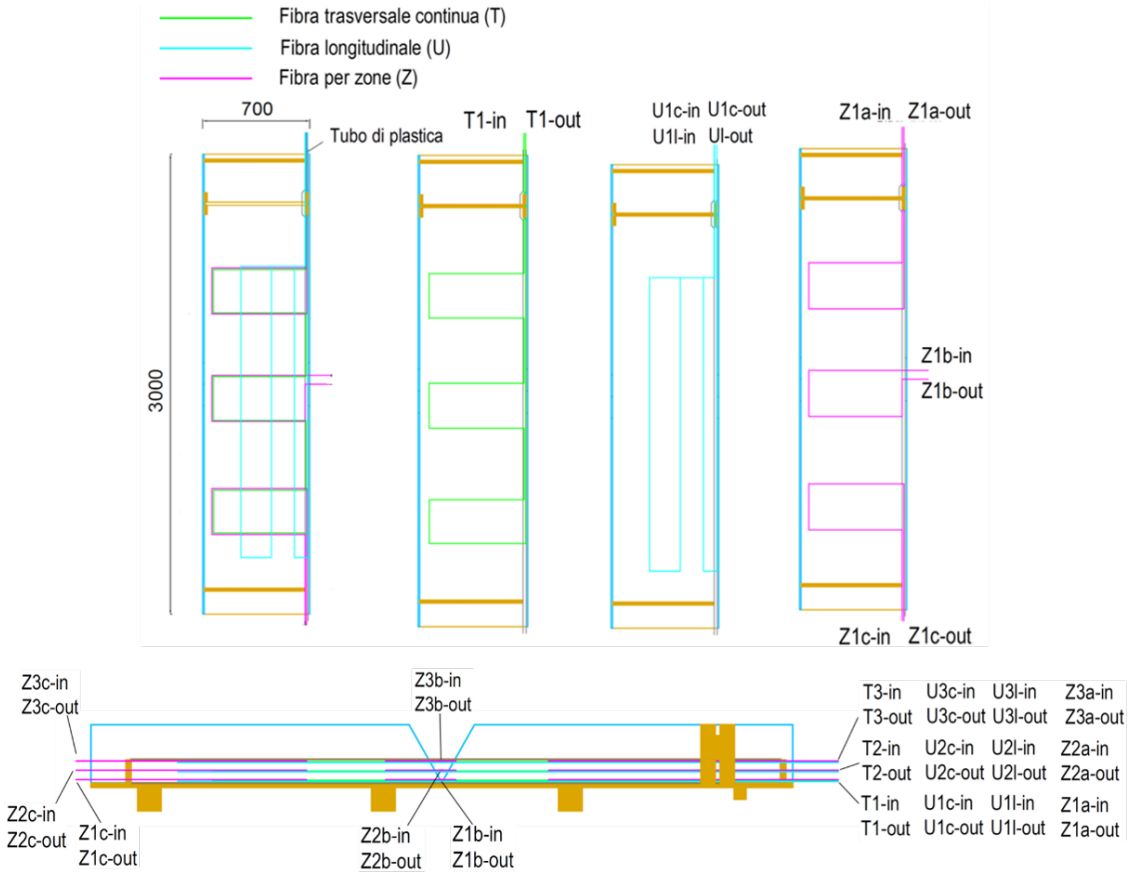

Figure 2.6: Displacement of the fibers in the model

Fiber name	*Depth*	*Configuration*
T1	9 cm	Transversal
T2	6 cm	
T3	3 cm	
U1c	9 cm	Longitudinal, central
U2c	6 cm	
U3c	3 cm	
U1l	9 cm	Longitudinal, lateral
U2l	6 cm	
U3l	3 cm	

Table 2.1: Fibers convention used

on the opposite part of the Ul fiber, it will be poorly sensed by it, and quite well by the central one. Different performances are then expected, relying on where and how intense the event is on a certain portion of the slide. Of course only one lateral fiber

was placed because of symmetry, since the performances of another one on the opposite side would be the same. The Z fibers reported in the leftmost scheme have been placed on the model, but they have never been considered during this work. They have a different aim, since their displacement is not all along the slide but considers just three small portions of it (uppermost, central, lowermost), which is to understand where the dangerous event is actually happening. In this way a quite precise space identification of where the dangerous event has happened could be provided, not just an alarm. This would simulate some sort of Multiplexed sensing, which has not been considered for the reasons exposed above. The analysis carried on during this Thesis work will mostly focus on the T fibers, in particular for a deep characterization, since the U fibers have been already studied in a preliminary work. Another important thing to notice is the bottom scheme in Fig. 2.6: here the longitudinal displacement of the fibers is represented, from left (upper part of the slide) to the right (bottom part of the slide). By focusing on the U and T fibers it's possible to see that their input and output is on the same side, exactly as specified before. Below, in Figure 2.7, an example of the kind of fibers coming out from the slide is reported. Note that the labels are T3 and U2c, and they are coupled,

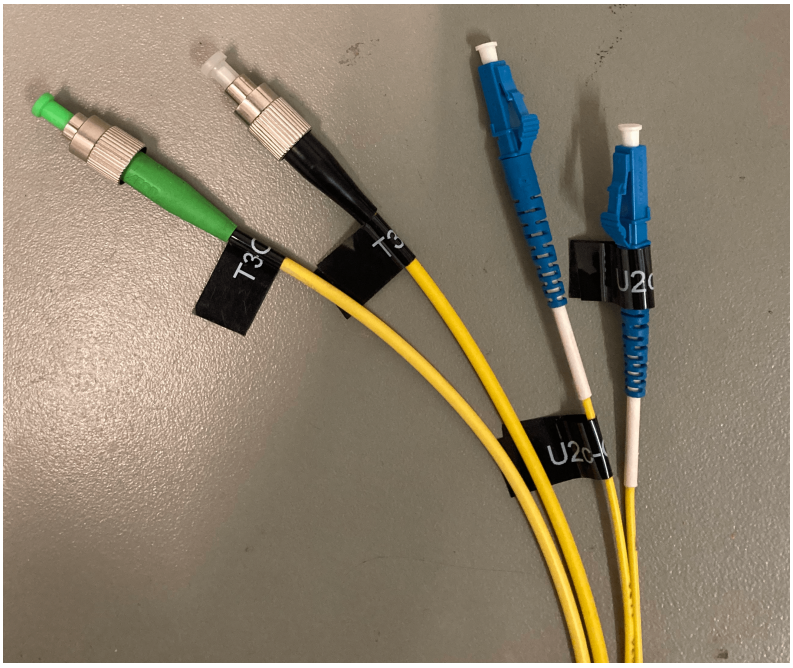

Figure 2.7: T3 and U2c fibers coming out from the soil to the setup table

since logically one of them should be for the Laser, the other for the Polarimeter. For sake of clarity, also a couple of pictures about the slide are reported in Fig. 2.8 and Fig. 2.9. The experiments simulating dangerous events happening, in the majority of cases some rocks falling, will be generated from the top of the slide in Fig. 2.9.

Figure 2.8: Bottom of the slide

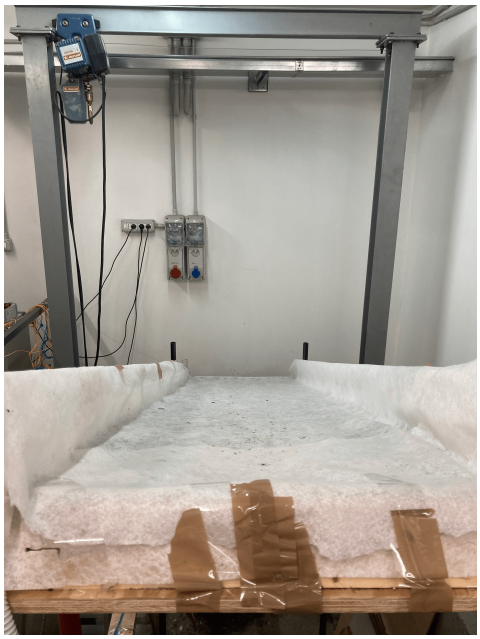

Figure 2.9: Top of the slide

2.1.3 Polarimeter

This is the core of the whole set up, since it is the sensor: this block is able to receive light from an input fiber, and give as output the SOP variations over time. The fiber received as input is the one coming out from the soil, where the probe light has been conveyed by the Laser. Let's dig in the detail of this very important block, reported in Figures 2.10 and 2.11 (Note: all the images have been taken from the datasheet of this instrument, provided by Novoptel [11]). The model of polarimeter used is the Novoptel PM1000. This kind of device allowed us to process the polarization state thanks to the possibility of connecting it to a computer, through a USB output on the back (Figure 2.11), which is the only kind of digital connection used. In this way, it has been possible to actually see what the polarimeter was measuring, thanks to a very easy-to-use user interface provided by Novoptel. Through this, some hardware parameters have been set, in particular the following are very relevant:

- Average Time Exponent (ATE): this is an internal averaging done by the Polarimeter after the 100 MS/s analog to digital conversion (AD). The number of samples which are averaged are 2^{ATE}, and this parameter can vary from 0 to 20. In particular, this means that the frequency at which the device is storing each sample is of

$$f_s = \frac{100\ MS/s}{2^{ATE}} \tag{2.1}$$

  which basically represents the sampling frequency. The highest achievable f_s will then be the one achieved by the AD without averaging (ATE = 0), so 100 MS/s,

Figure 2.10: Front side of the Polarimeter

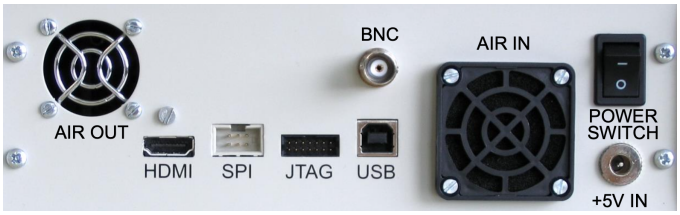

Figure 2.11: Back side of the Polarimeter

while the lowest one will be 95.4 S/s, found by applying the Equation in 2.1, with $ATE = 20$.

- Memory Exponent (ME): this defines the size of the internal memory block that is being written into, and ranges from 10 (2^{10} samples stored) to 26. This is important since, combined with the ATE, it defines the overall recording time, computed as

$$t_{tot} = 10 \cdot (2^{ME} \cdot 2^{ATE})\ ns \tag{2.2}$$

- Standard Normalization mode: this means that the Stokes vector is normalized to unit length, so that its tip will always appear on the surface of the Poincaré sphere. This is not the only operating mode provided by the Polarimeter, but it's the only one exploited in this work.

The ATE and ME parameters can be set both from the user interface provided with the device, and also by using some specific Matlab commands. Some very interesting features that the user interface provides by default are the graphical evolution in time of the three Stokes parameters, of the power and of the State Of Polarization Angular Speed (SOPAS), as can be seen in Figures 2.14, 2.12 and 2.13. These Figures are just a template to have an idea of what is meant to happen when performing the tests, but they are not taken from a real experiment. The red line in Figure 2.14 is what the tip of the $\vec{S}$ vector describes during its time evolution, and also the values of the three parameters are reported. One parameter that needs to be taken under control is the power received by the Polarimeter, showed on top of Figure 2.14. For our purposes, to get a proper good functioning of the system, it should be between about -6 dBm and -1 dBm. The other two Figures (2.12, 2.13) are instead showing the time evolutions of the

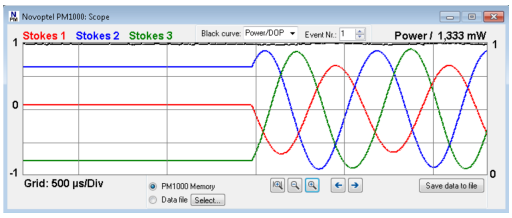

Figure 2.12: Stokes parameters and power time evolution

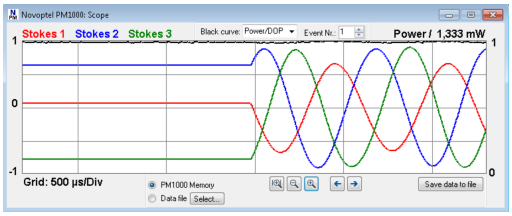

Figure 2.13: SOPAS time evolution

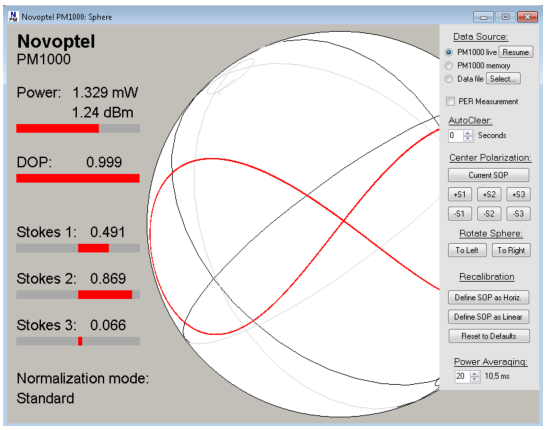

Figure 2.14: Poincaré sphere on the user interface

Stokes parameters, of the SOPAS and of the power too. In general these plot are very useful to check in real time if the acquisition was actually going smoothly, and if the system was able to do what it was meant to, but they have not been analyzed in detail: all the processing of the Stokes parameters and of the SOPAS has been done exclusively through Matlab. For example, thanks the Poincaré sphere it was possible to see if, by moving a fiber, the motion was detected by the Polarimeter as a movement of the vector, and so if it worked successfully, before starting any kind of test. The time evolution of the SOPAS was instead very useful to check if the acquisition went smoothly, by seeing the evolution displayed, and checking in real time if some irregularities showed up.

2.1.4 Processing block and GUI

This is the very last portion of the set up, the one on which the processing will actually happen and the results shown. This is extremely simple, it is just made of a computer, connected to the Polarimeter through an USB cable, which could process the input data. The GUI is actually, in our case, the computer screen on which all the results are displayed, just a way to show what is obtained after the processing. This block can work in two modes:

- Offline mode: the processing in this case is not done in real time, and the results are showed only after the data have been processed. This coincides to the post-processing versions of all the algorithms used to analyze data. In this case the computer works basically as a storage unit for the Polarimeter, which, after having set the ATE and ME, is able to save, thanks to the Novoptel user interface, the samples inside the computer as .bin files. These ones are containing the Stokes parameters discrete time evolution. Of course this is useful just to perform a

preliminary analysis on the samples acquired, in order to know better what is the kind of signal that will be processed. Further on, some different algorithms will be adopted and optimized: if the processing is not in real time, the working mode is always defined as Offline, no matter what is the algorithm applied.

- Real-time mode: this is the final version of this block, which is able to read the samples in real time from the Polarimeter, and make all the computations by means of them. To arrive at this final version, the analysis provided by the Offline mode was anyway of fundamental importance for the debugging of the algorithm.

As last consideration, now that all the fundamental blocks have been introduced, in Figures 2.15 and 2.16 the final configuration of the experimental setup used is reported, in which all the elements described up to now can be recognized but the downhill model, which is on the right and quite too big to fit inside the picture. Note the fibers that are wrapped near the set up table, which are the 9 couples of input/output, and the two connected to both the Laser and Polarimeter. The computer is then connected to this last one, as previously said.

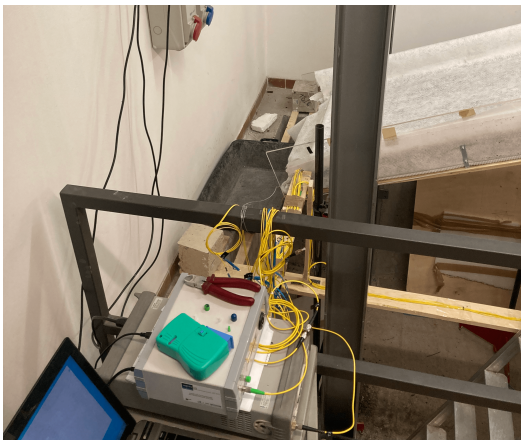

Figure 2.15: From the set up table to the in scale model

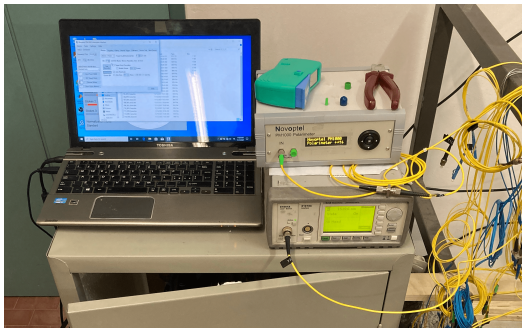

Figure 2.16: Point of view in front of the set up table

2.2 First tests and results obtained

In this Section some detailed focus on the experiments is provided, by means of the setup portrayed in the previous Section. Also, a description of a very preliminary analysis is reported, and some first results which lead to the construction of an algorithm able to perform real time detection of dangerous events.

2.2.1 Tests performed

The idea behind all the tests was about reproducing some events which could reasonably simulate some dangerous ones, and see how the fibers buried inside the soil would react, also depending on their configuration. The acquisitions performed are on the following four events:

- Single Rock (SR): this is just a single small rock, which is placed at the top of the slide, and carefully let roll up to the bottom. This is not the most dangerous event that has been simulated, but still one which is quite relatable to some real conditions. The rock used is a quite rounded up one of about 4 cm of diameter (Figure 2.17).
- Cylinder (C): this is instead a cylinder made of ceramic material, having 15 cm of length and 5 cm of diameter. This is not related to a specific "real" event that could happen in a real world like the above one, the goal was just to simulate some generic event which could be used both to see how the detection of the other ones with respect to this behaved, both how a possible event having this intensity and this evolution could be detected. In a real condition, this could be very roughly associated to the falling and rolling of a tree along the side of a mountain, but still it's quite unrealistic (Figure 2.18).

Figure 2.17: SR compared with the DF ones

Figure 2.18: SR compared with the Cylinder

- Debris Flow (DF): this event is the most realistic one among all and also the most dangerous. It is generated by the falling of 25 rocks (a bit smaller or at most comparable with the SR) along the slide. This is very intense, and it is something

that is very likely to happen in a mountain scenario, in a way that it's very similar to the one that has been simulated (Figures 2.19 and 2.20).

- Noise: just background noise, what the fiber senses when no event is happening. These acquisitions are of fundamental importance since the goal is to build a detector, which is able to always correctly detect dangerous events, and not trigger when just noise is there.

Figure 2.19: Debris Flow rocks used for the tests

Figure 2.20: Debris Flow rocks during the test

From the pictures reported of the single events, is very clear how the size of the SR and C events are extremely small compared to the DF, so it is quite reasonable for them to be on a different scale of intensity. The Single Rock and the Cylinder are actually very comparable in these terms, although their physical shape and dimension is quite different. Now that the whole scenario has been exposed, let's talk about the post-processing algorithm that has been used to elaborate the data stored in the .bin file. Just to make it perfectly clear, the steps performed in this preliminary phase are in the Offline mode, and consist in:

1. Set the ATE and ME through the user interface provided by Novoptel.
2. Start the acquisition, which will last for t_{tot} provided in Equation 2.2.
3. Generate the events.
4. Check the evolution of the parameters (Figures 2.12 and 2.13), to be completely sure that nothing went wrong.

5. Save the samples acquired inside the computer as .bin files.

For the very first experiments, the parameters have been set as reported in Table 2.3 and, once started, only one event has been acquired, among the ones mentioned above. In

Event	*# tests*	*Fibers*
SR	10	T1, T2, T3, U1l, U2l, U3l, U1c, U2c, U3c
C	10	
DF	10	
Noise	3	
Total	33	

Table 2.2: Tests performed per fiber

particular ten acquisitions per fiber have been made, for one single event, and three acquisition per fiber for Noise, as reported in Table 2.2. Note that the following preliminary results are exclusively on the T fibers, since the other ones have been already previously characterized in an even earlier stage. The total number of acquisitions done were then 33 per fiber, leading to a total of 99 measurements characterizing the T configuration.

ME	ATE	f_s [kHz]	t_{tot} [s]
20	11	48.83	21.5

Table 2.3: Preliminary parameters set

2.2.2 Preliminary analysis of the data acquired

Before proceeding with showing some plots and discussing what is their meaning, a small explanation of how they have been obtained is necessary, looking in detail at the processing chain. Up to now, the focus has been on the Stokes parameters, and another important quantity which can be retrieved from these ones, the SOPAS. The main starting idea was to exploit the Stokes parameters to detect polarization change over time. This is quite difficult to do, since the change is happening on a three dimensional space, where the Poincaré sphere is defined. Moreover, the change happening to the Stokes parameters when an event happens is extremely small. In Figures 2.21, 2.22 and 2.23 the evolution of all the four Stokes parameters along time is reported, for all the three different events. It's quite clear that other than being quite difficult to evaluate in a three dimensional space for a simple computational algorithm, the polarization change of the single S_i is quite ridiculous, and would be not easy to identify and process. Note that these first tests are also generated considering the most external fiber, that should be the most sensitive: only the DF seems to have some quite reasonable effect on the Stokes parameters, the other events are quite rough to detect in this way. For these reasons, the

vector $\vec{S}$ is actually only used to compute the SOPAS parameter, which is solving the three dimensional problem, since in this case the evaluation is only about an angular speed, so on just one dimension. Moreover, this is much more sensitive to polarization changes than the single Stokes vector component.

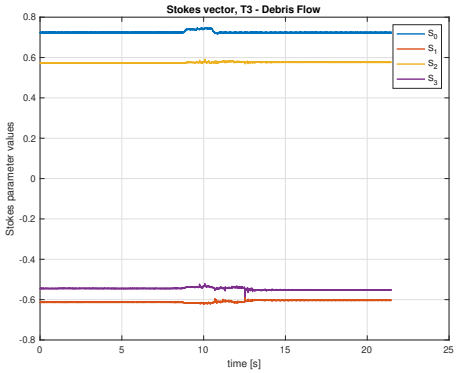

Figure 2.21: $\vec{S}$ time evolution on T3, DF

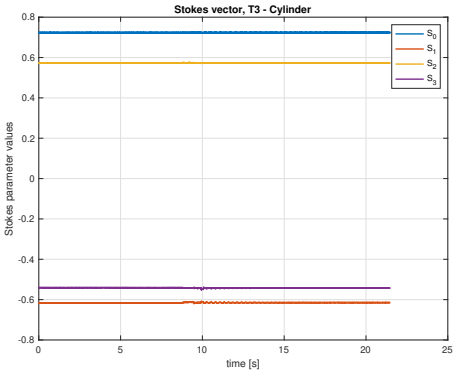

Figure 2.22: $\vec{S}$ time evolution on T3, C

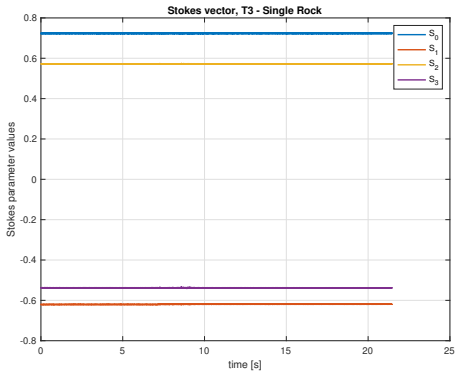

Figure 2.23: $\vec{S}$ time evolution on T3, SR

The angular variation of $\vec{S}$ is computed as in the following Equation 2.3

$$\begin{aligned}\Delta\theta(i) &= \arccos\left(\frac{(\vec{S}^i, \vec{S}^{i-1})}{\left\|\vec{S}^i\right\| \left\|\vec{S}^{i-1}\right\|}\right) = \\ &= \arccos\left(\frac{(S_1^i - S_1^{i-1}) \cdot (S_2^i - S_2^{i-1}) \cdot (S_3^i - S_3^{i-1})}{\sqrt{(S_1^i)^2 + (S_2^i)^2 + (S_3^i)^2} \cdot \sqrt{(S_1^{i-1})^2 + (S_2^{i-1})^2 + (S_3^{i-1})^2}}\right)\end{aligned} \tag{2.3}$$

where $\vec{S}$ is in this case the Stokes vector made of only three components, S_1, S_2, S_3, since the S_0 is not relevant to compute the speed, and just a normalization factor of the

other three parameters. The parameter i is instead identifying the i-th sample of the vector. Of course, the result of Equation 2.3 is in radiants: $\Delta\theta(i)$ needs to be divided for the sampling period to get an angular speed, as in 2.4.

$$\Delta\omega(i) = \left| \frac{\Delta\theta(i)}{T_s} \right| \tag{2.4}$$

The quantity expressed in 2.4 is what actually is computed and considered for all the next tests, and represents the absolute value of the SOPAS, sample by sample.

2.2.3 First steps into SOPAS processing

Now that all the reasons behind the computation of the parameter in Eq. 2.4 have been explained, the results regarding the tests performed can be shown. Remember, and this is of fundamental importance, that all the results that will be reported from now on, unless further notice, have been computed under the conditions reported in Table 2.3. These parameters are only valid for these results, they will be further changed in order to get the best out of the acquisition, but this is a point that will be discussed in detail later on.
The very first (Offline mode) algorithm that has been used to get some results consisted in a very simple one, following the steps in Figure 2.24. Two important operations are

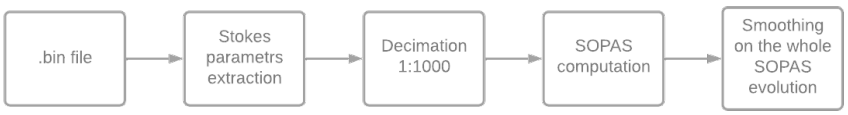

Figure 2.24: Initial Offline mode algorithm

reported in the block scheme and will determine a lot about the following choices, so they are worth of further discussion:

- Decimation: this operation consists in taking a discrete time signal as input, and give as output the same signal, but in which only one sample every D is considered. In this first analysis, D is considered to be equal to 1000, and it will actually be left untouched also for the following discussions. This technique is useful to reduce the computational time of the blocks which follows (less samples to manage), despite it actually deletes samples, so informations. In this way the sampling frequency is reduced and, because of the Nyquist theorem, this could bring aliasing problems, which could be mitigated using a Low-Pass filter, not applied in our algorithm in Fig. 2.24.

- Smoothing: this operation consists in taking a discrete time signal as input, but instead of deleting samples, some operation is performed. The signal is considered

window by window, shifting it each time of one sample along the time evolution. The notation used to express the window size in seconds is T_{mov}, and each time a window is considered, the mathematical average is performed as follows, in Eq. 2.5,

$$\Delta\omega_s(i) = \frac{\sum_{j=i}^{m} \Delta\omega(j)}{m} \tag{2.5}$$

where $\Delta\omega_s(i)$ is the output value of the smoothed sample, while $m = \lfloor \frac{T_{mov}}{f_s} \rfloor$, which is the number of samples inside the window. The idea of smoothing is then to give as output the ratio between the summation of all the values inside the window and the length of the window in samples. The advantage is that Noise oscillations are for sure mitigated (more and more increasing T_{mov}), even if the evolution over time spreads with the window size. This operation could be very easily implemented by a FIR filter, with the disadvantage of introducing delay in the processing chain: to output a $\Delta\omega_s(i)$ sample, a buffer with all the samples contained by the window needs to be filled, so a delay of T_{mov} seconds will always be introduced.

Note that decimation is applied on the Stokes parameters, while smoothing on the SOPAS discrete time evolutions. At this stage it is computed through the Matlab function `smooth()` on the whole vector of SOPAS, so of course in post-processing. The results over 10 runs of all the events and 3 runs of Noise only are reported in Figures 2.25, 2.26, 2.27 and 2.28 for fiber T3. The other configurations behaviors are reported in Appendix A, since they are very similar to the ones reported here. What gets to be quite clear

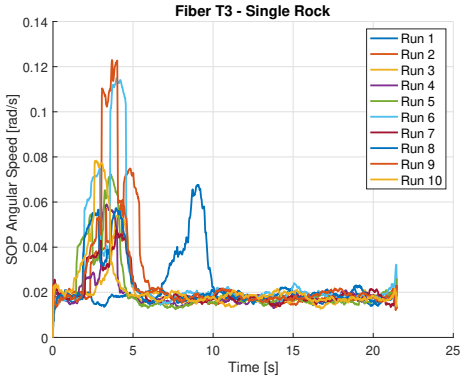

Figure 2.25: T3, SR runs

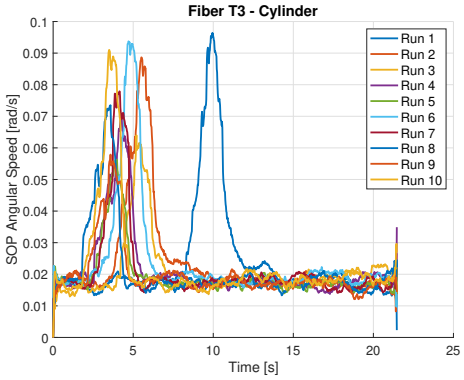

Figure 2.26: T3, C runs

from these results is first of all the fact that now the change of SOP when some stress occurs on the fiber can be seen absolutely perfectly. The choice of adopting the SOPAS metric makes the occurrence of each events over time much more evident respect to what was obtained only with the Stokes parameters, in Figures 2.21, 2.22 and 2.23.
About the single events, focusing on just this fiber (not important which one, this concept can be made in a general way, since the behavior is quite the same for all of them), it's

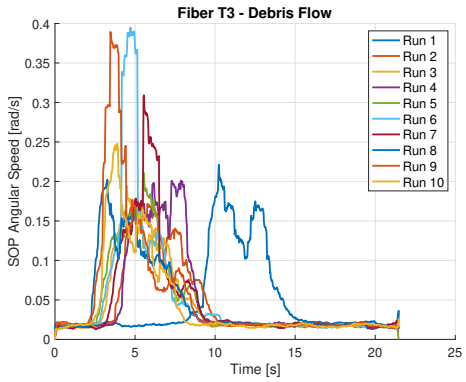

Figure 2.27: T3, DF runs

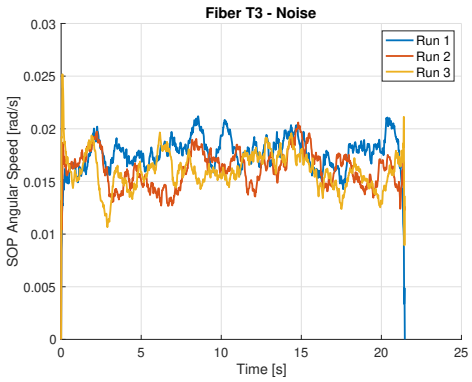

Figure 2.28: T3, Noise runs

possible to see that the evolution of C and SR is always shorter than the DF, which usually lasts at least one or two seconds more. Moreover, this one is always the most intense of all, no matter what fiber is taken into account. These plots are useful also to see the effect of Noise on the acquisitions, which is very related to the sensitivity of the fiber itself. By looking for example at all the evolutions of the Single Rocks (Figures 2.25, A.1, A.5), it's absolutely clear how the oscillations due to Noise when the event is not happening (so after or before it), are increasing with the depth (distance from the soil surface, see Table 2.1) of the fiber. This trend is also confirmed by the three Noise evolution acquired in Figures 2.28, A.4, A.8. This means that if an event is happening on T1 this fiber will be more likely, in this specific situation and exploiting this specific algorithm, to misunderstand it and confuse Noise with Single Rock event, which of course is fundamental to avoid.
Let's now take a look at what happens for the single events in different fibers, and compare their behaviors. Since basically the most important point is the maximum value

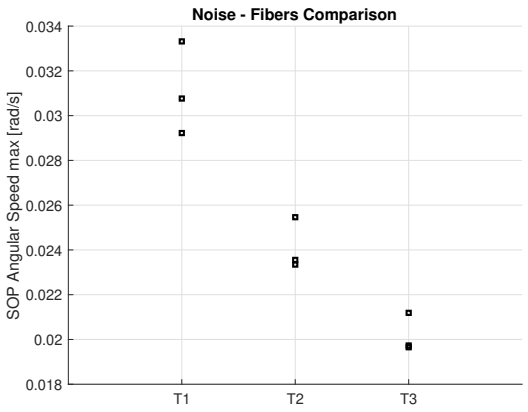

Figure 2.29: Noise, max comparison

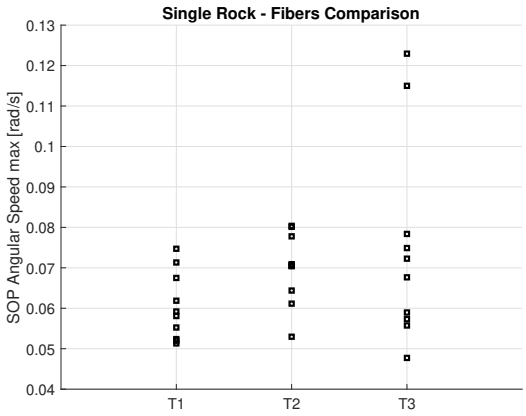

Figure 2.30: SR, max comparison

assumed by the SOPAS, which can be in some way set as an indicator of the sensitivity

of both the post-processing algorithm and the fiber that is used, some plots showing the trend of the maximum of the SOPAS have been reported for all the ten acquisition per fiber, fixed a single event. This analysis is reported in Figures 2.29, 2.30, 2.31 and 2.32. It is very clear how for the DF and SR events the sensitivity of the fibers decreases

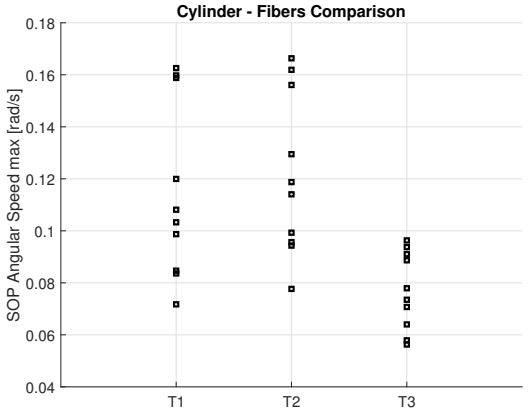

Figure 2.31: C, max comparison

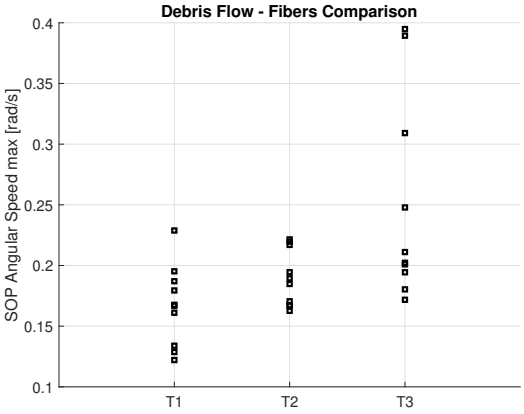

Figure 2.32: DF, max comparison

with their depth. Looking at Figure 2.30, all the maxima are inside the range around 0.05 rad/s and 0.08 rad/s, for both T1 and T2 fibers, while for T3 they spread even until up to something more than 0.12 rad/s, in a couple of tests. This situation translates in an enhanced sensitivity of T3, and in an almost identical sensitivity of T1 and T2, which have their dots almost in the same range. A very similar situation is repeated in Figure 2.32 for the Debris Flow, even if the maxima are much more intense going from 0.12 rad/s to 0.4 rad/s, overall. The sensitivity of T1 and T2 seems to be always quite identical, while the one of T3 is basically doubled (this is the one reaching for some tests 0.4 rad/s). The Cylinder (Figure 2.31) is following the same trend but for T3: in this case the range of values is shrinked, this fiber really seems to sense very badly this kind of event. It could be just a case, or it could be that, being the Cylinder a completely different event with respect to the other two, the vibrations transferred to the soil are much better perceived by deeper fibers with respect to more superficial ones. Noise is also extremely important, in Figure 2.29. First thing to notice is the range of values, which is on a lower, non intersecating, range with respect to the events maxima. This is a good thing, since it means that there is a separation between the Noise level and the event peak, which was already evident in Figures from 2.25 to 2.27. Furthermore the Noise level seems to decrease with the depth of the fibers, which also is a very nice result, and should improve the performances of the most superficial fibers. To get a further characterization of the sensitivity of the different fibers to the events that have been generated, an average of the SOPAS maxima of all the events happened on a single fiber is reported in Figure 2.33. From this Figure it is very clear how the sensitivity decreases by going from T3 to T1, and still the Cylinder seems to be not very well sensed by T3. What this plot highlights even more strongly than before, is again the fact that the

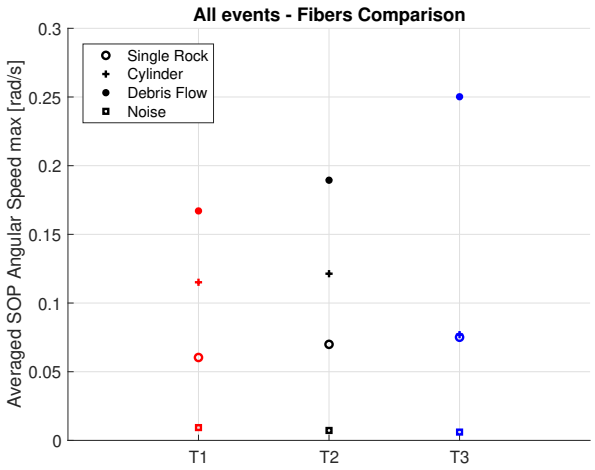

Figure 2.33: Comparison between all events and all fibers

Debris Flow is always much more intense than the SR and C, and that the level of Noise is, in the worst case, much lower in SOPAS compared to the true events. This translates in a probably good capability to detect events correctly of all the fibers considered.

2.2.4 Detection problem

With all the above preliminary analysis done about the SOPAS evolution in time, keeping the same processing approach that has been explained in the previous Section, is now possible to evaluate how much a system based on these concepts is able to correctly detect dangerous events. Three possible situations could occur:

- False Alarm (FA): no dangerous event happened, so there is nothing to detect, but the algorithm detects something as if it was dangerous.
- Missed Detection (MD): the dangerous event is not detected. This means that it actually happened, but the algorithm failed to recognize it.
- Correct detection (CD): the dangerous event is actually correctly detected by the algorithm. This means that no Missed Detection nor False Alarm occurred instead.

Of course False Alarm and Missed Detection need to happen very rarely, while the Correct Detection needs to occur basically the 100% of the times. To perform this kind of detection, the algorithm that we developed is a *Threshold Algorithm* on SOPAS. It is very simple, and consists on just checking if the samples of the SOPAS are under or above threshold, and trigger an alarm if this is the case. This control is done as the very last step, right after the smoothing. Two parameters are at the basis of our proposed algorithm:

- $\Delta\omega_{th}$: this is the threshold that has been set on the SOPAS values, in rad/s. If it is exceeded an alarm is given since a dangerous event occurrence is very likely, as will be discussed lately.

- T_{mov}: this is the value of averaging window, in seconds. All the SOPAS computations are always smoothed through a moving average, in order to reduce oscillations and of course Noise.

After several investigations, we found that these two values need to be optimized together as a couple, since taking a big T_{mov} the actual peak of the speed evolution will assume smaller and more smoothed out values, and this would represent a problem for a proper threshold setting. By looking at Figure 2.34, this effect should be quite clear: by fixing a

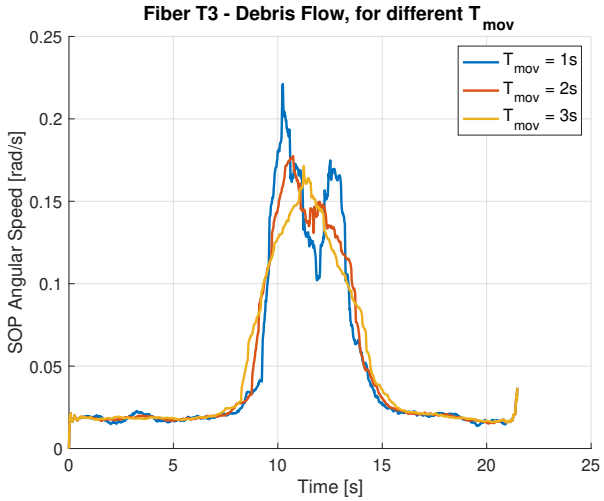

Figure 2.34: Smoothing with different T_{mov}

threshold at 0.2 rad/s which is good for the blue curve, values of T_{mov} no bigger than 1 second, if anything smaller, needs to be used. The bad effect would be an excessive smoothing over time of the SOPAS, that would lead to a behavior like the other two evolutions (orange and yellow one), which are under threshold with just 1 and 2 seconds of difference in the length of the window with respect to the first one. This means that $\Delta\omega_{th}$ and T_{mov} are very critical values that need to be set together, otherwise the detection system could not work properly. In order to get a more clear point of view on these two parameters and see if a safe area where all the events can be correctly detected exists, avoiding False Alarms or Missed Detection, some significative plots have been generated to further investigate this issue, called "detection maps". They have been obtained varying the values $\Delta\omega_{th}$ and T_{mov}, and seeing if, for a specific parameter pair, the SOPAS samples are under or above threshold. Noise is of course considered as a disturbance, so if for a certain parameter pair the angular speed evolution of Noise is above threshold, False Alarm is triggered and the detection is not correct, no matter

what is the value assumed by the SOPAS of the actual event. Unless otherwise specified, the values considered in this analysis are:

$$\Delta\omega_{th} \in [0.01;\ 0.1]\ rad/s$$

with step of 0.005 rad/s, and

$$T_{mov} \in [0.1;\ 3]\ s$$

with step of 0.05 seconds. Let's consider, fiber per fiber, how this Offline mode algorithm is robust against uncorrect detection. In particular, only one detection map considering all the three events will be commented for each fiber but for T1 for which, in order to properly understand the analysis made, the maps of the single events are also reported in Figures 2.35, 2.36, 2.37, in addition to the one in Figure 2.38. For the other two fibers only the overall performances on the three events are instead showed, in Figures 2.39 and 2.40. The single events maps are showing in the green areas the values of $\Delta\omega_{th}$

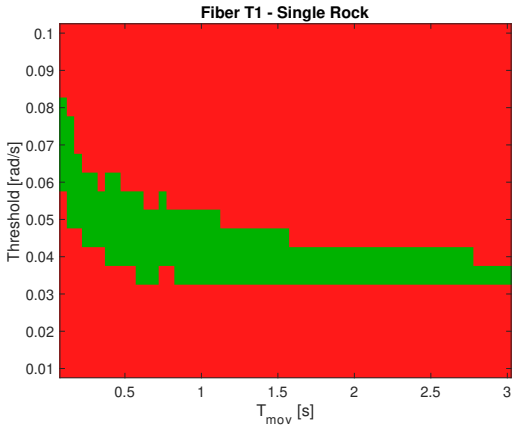

Figure 2.35: SR, T1 detection map

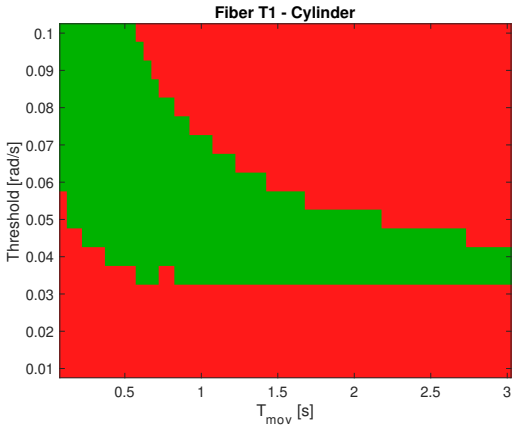

Figure 2.36: C, T1 detection map

and T_{mov} that are always giving correct detection of all the ten events considered, of one single kind. The red areas are instead some unsafe ones where either Missed Detection or False Alarm happened: even if they are represented with the same color, it is perfectly reasonable to think that the red area below the green one is caused by False Alarms, since it means that the threshold is so low that the alarm is triggered by Noise oscillations. On the contrary all the unsafe areas above the green one (usually concentrated on the upper right part of the plots) are reasonably due to Missed Detection, since the threshold is so high that for sure Noise cannot reach it, but not even the SOPAS evolutions of the actual events. Note also that the green area tends to shrink for growing values of T_{mov} and decreasing $\Delta\omega_{th}$, and viceversa. This is because by increasing the window, the averaging considers also SOPAS samples not belonging to the peak, which lowers the overall smoothed value (exactly what happens in Fig. 2.34), leading to a smaller threshold needed for correct detection. If instead T_{mov} is decreased, Noise oscillations are not smoothed, on the contrary they are in some way enhanced, so an higher threshold

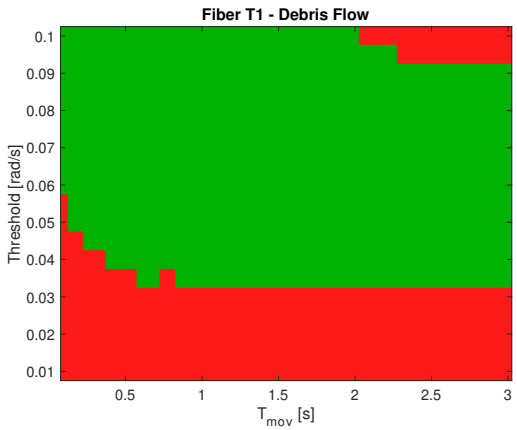

Figure 2.37: DF, T1 detection map

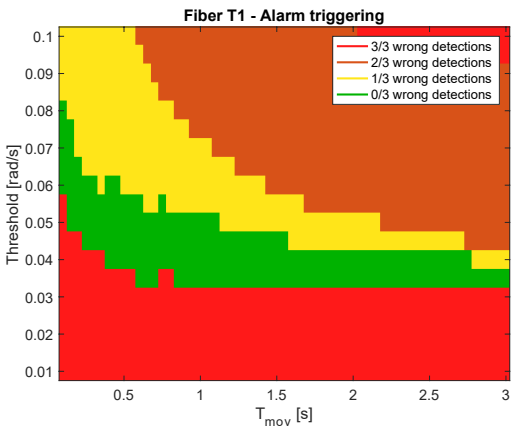

Figure 2.38: T1 detection map

is needed to avoid False Alarms. Figures like 2.38 report instead what happens if fixing one fiber and considering all the events (all the ten evolutions of DF, SR and C). They have been obtained by intersecating all the maps of a single event evolution and they summarize in a very useful way the situation for a given fiber. This means that in the green area for sure all thirty events are correctly detected, while in the orange and yellow parts at least one category got its detection failed for at least one evolution among ten. In the red one instead for sure at least one evolution per event is not detected correctly. As expected from the results obtained above, the detection performance on T1, which is the the deepest fibers between all the T, is not extremely bad, but could be better. In Figure 2.35 a quite small green area is obtained, meaning that for the majority of couples of $\Delta\omega_{th}$ and T_{mov} detection of SR fails. This trend tends to improve considering Cylinder and Debris Flow (Figures 2.36 and 2.37), which are detected much better than the Single Rock. Overall, despite the green area is not so spread, this is a good result, because it means that there are anyway not a few values that grant the correct functioning of the system. This can be seen in Fig. 2.38, where it is clear that the algorithm works for a lot of values, even if the perfectly safe area is not so big, which is still a very good result for the least sensible fiber. Also, as can be expected from the results obtained in Fig. 2.29, T1 is the most noisy fiber of all, and this is reflected on the very high red area below the green one, which as said is reasonably caused by FAs. For the same reasons in the next configurations this unsafe zone is expected to shrink. Let's now take a look at what happens on the T2 configuration: in Appendix A, Figures A.9, A.10, A.11 report what happens for the single event, with basically the same results of T1, but better. This fiber is intermediate, placed at 6 cm from the soil surface, it should then be a bit more sensible and, following the previous results a little bit less noisy. The result in Figure 2.39 is confirming this trend: the green area is wider, and the red one below seems to be smaller, meaning that the effect of Noise is reduced with respect to the previous case. For what concerns T3 instead, this is absolutely the most sensible of all the three configuration, as seen. From Figure 2.40 it's clear that the Noise floor is even lower, and

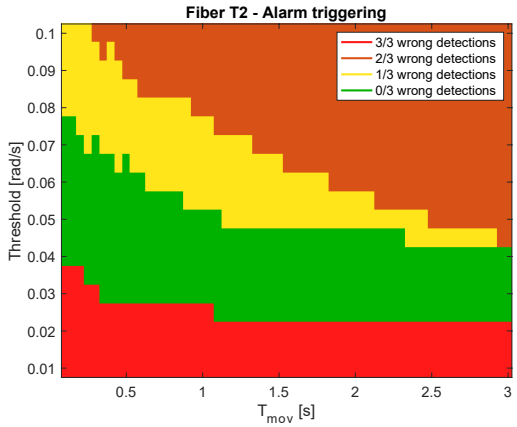

Figure 2.39: T2 detection map

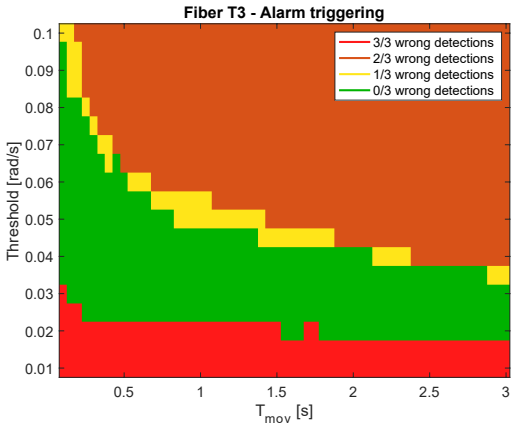

Figure 2.40: T3 detection map

the area related to False Alarms happening is actually shrinked even more. The overall performances of T3 seems to be the best one of all the three, since the green area is quite spread. The only limitation to this zone is due to the Cylinder, which probably shrinks the wideness of the safe area, due to the quite bad sensitivity of T3 on this event. Overall, what emerges from this analysis, and this is very important to underline, is the fact that the post-processing algorithm is always able to perform with great results, if $\Delta\omega_{th}$ and T_{mov} are chosen smartly. For this very early stage, this result is the main one that needed to be achieved. From now on, all these performances can only be optimized, and made better.

2.3 Spectrogram analysis, sampling frequency optimization

Before proceeding, in order to sum everything up, the main key points observed up to now are:

- Use of State of Polarization Angular Speed in place of the single Stokes parameters, since it is much comfortable to use and grants better results.
- The algorithm used for detection (Offline mode, for now) is a threshold one, that means that it checks if SOPAS samples are above one fixed threshold. If this is true, triggers an alarm.
- Smoothing is used to reduce Noise oscillations.
- The couple of threshold ($\Delta\omega_{th}$) and moving average window (T_{mov}) needs to be set carefully. Despite this, it has been demonstrated that there are a lot of values that grant a perfect functioning of the algorithm.

What is now necessary is to set the sampling frequency in an appropriate way, since the one used up to now has been considered as an initial value, to be properly discussed. The characterization in the time/frequency domain of the parameters acquired, following what has been done in [9] and [8], helped a lot in choosing the perfect f_s value. The analysis proposed, takes into account a new parameter $\Delta\vec{s}$, defined in Equations 2.6 and 2.7.

$$\begin{aligned} \Delta s_1(i) &= \bar{S}_1 - S_1(i) \\ \Delta s_2(i) &= \bar{S}_2 - S_2(i) \\ \Delta s_1(i) &= \bar{S}_3 - S_3(i) \end{aligned} \tag{2.6}$$

This vector is computed as the difference between the mean value of the discrete time evolution of each Stokes component and the actual time evolution of the component, generating something like Eq. 2.7.

$$\Delta\vec{s}(i) = (\Delta s_1(i), \Delta s_2(i), \Delta s_3(i)) \tag{2.7}$$

The actual value which will be considered for the spectrogram analysis is the one in Eq. 2.8.

$$\left|\Delta\vec{s}(i)\right| = \sqrt{(\Delta s_1^2(i) + \Delta s_2^2(i) + \Delta s_3^2(i))} \tag{2.8}$$

The main idea is to find a parameter that is able to properly show in the spectrogram a "packet" of energy having some sort of distinctive shape, when some event happens along the fiber. This would be of extreme interest, because it could give a further characterization of the events, giving even more information about them. In the papers cited above [9], [8] it has been shown how, using a very similar technique, earthquakes on the Atlantic Ocean have been detected by one submarine fiber as energy on a very precise time/frequency spot. A very similar analysis can be done exploiting $\Delta\vec{s}$, which basically

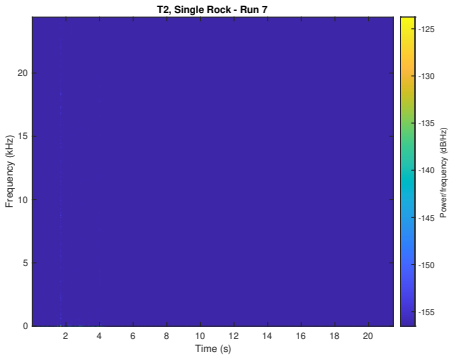

Figure 2.41: SR on T2

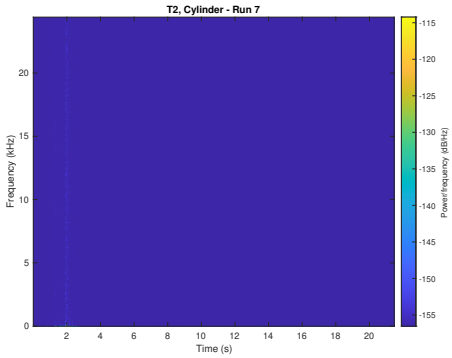

Figure 2.42: C on T2

accounts how much $\vec{S}$ deviates from its mean value, which in fact is subtracted, instant by instant (discrete time is always represented as i). The actual value represented in the spectrograms is not exactly the one in Eq. 2.8, but its value subtracted by its average, and

of which the squared absolute value is computed. This operation is very important: in the spectrogram the mean value is a continuous component around the zero frequency, which can be quite annoying since it could hide some other component, much more significative. By subtracting it, this issue can be mitigated. The first results reported in Figures 2.41, 2.42 and 2.43 are obtained through the Matlab function `spectrogram()`, on the parameter described up to now, considering the original sampling frequency, $f_s = 48.83$ kHz, on a single run of a specific event, lasting as always $t_{tot} = 21.5$ seconds, and with no decimation applied. Note that only the results on T2 are being reported since for the goal of this analysis they are enough to make the point. These representations

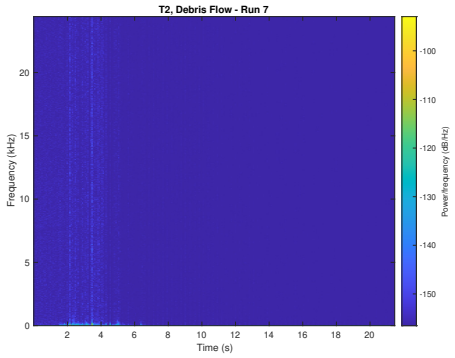

Figure 2.43: DF on T2

are extremely bad: it is very hard to see if something happens, but focusing on the very low frequencies and in the initial part of the time axis, where the event is located, it can be seen that something is going on. Everything is very zoomed out, some energy appears at the very low frequencies as it will be shown in some of the next graphs, and sometimes it extends up until the higher ones, but with negligible intensity. Note that the frequency range inside which these spectrograms are represented goes from zero to half f_s, because of the Nyquist theorem. Summing up, these plots are telling us that the actual important information contents of the events occupies an extremely small percentage of the range reported in the spectrograms. It is then quite reasonable to try to reduce the sampling frequency: in this way something could come out of the Noise floor. This downsampling operation has been performed through the `resample()` Matlab function: this allowed to try different sampling frequencies without repeating the experiments, to find out what is the actual best one, also keeping in mind that the minimum f_s at which the Polarimeter hardware can run is 95.4 S/s. After some tests, of which only the very last ones are reported, it has been found out that the main frequency components of Single Rock and Cylinder events can be found adopting an f_s being 200 time lower than the starting one. For what concerns the Debris instead, the main component is wrapped around a larger range, which can be found by using a sampling frequency 100 times lower. The new sampling frequencies obtained by downsampling

would be then equal to Equations 2.9 and 2.10.

$$f_s^C = f_s^{SR} = \frac{f_s}{200} = \frac{48.83\ kHz}{200} = 244.15\ Hz \tag{2.9}$$

$$f_s^{DF} = \frac{f_s}{100} = \frac{48.83\ kHz}{100} = 488.30\ Hz \tag{2.10}$$

The results are reported in Figures 2.44, 2.45 and 2.46. Now it's possible to have a much clearer view of what the spectrogram is showing: everything is much more zoomed in, and with an higher resolution. Cylinder and Single Rock start to lose intensity in frequency domain starting from around 60/80 Hz, while the Debris Flow is much stronger on a wider range, which starts to attenuate from around 200 Hz. The goal of this analysis

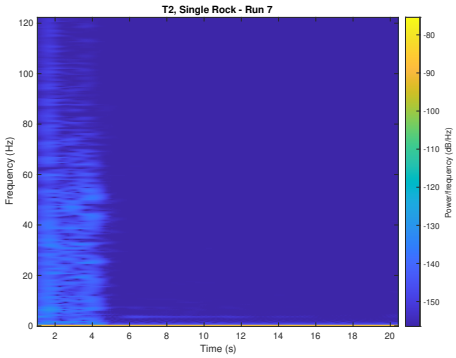

Figure 2.44: SR on T2 @ f_s^{SR}

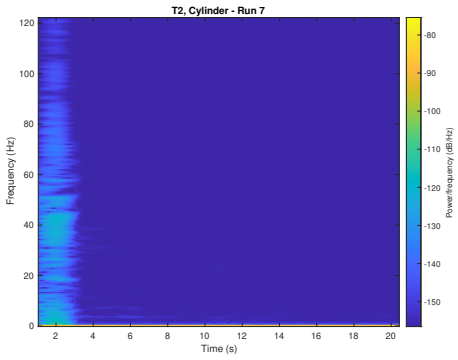

Figure 2.45: C on T2 @ f_s^C

was to find if some frequency pattern could be find in the spectrogram. Actually it seems that each time an event happens it excites a whole range of frequency, which is growing with its intensity. This is confirmed by the Debris which is made of several rocks, each one exciting in different times instant a whole frequency range. In the end, no particular pattern can be identified in a way that it could add some information on the SOPAS time evolution, further characterizing each event. The only thing that actually got our attention are two continuous "lines": in Figures 2.44 and 2.45 they are below 10 Hz, and they persist even after the event is concluded, being quite intense in particular between 6 and 10 seconds for the SR, and between 4 and 8 seconds for the C, while the events seems to end around 4 and 3 seconds, respectively. Same thing happens for the DF in Figure 2.46, in an even clearer way: two continuous lines below 10 Hz are very visible and also very constant in intensity all over t_{tot}, even after the end of the event, which happens around 7 seconds. For this specific event also some other very similar lines appear right below 200 Hz. It is very reasonable to think that they are due to a physical oscillation of the model: basically when generating the event the structure starts to vibrate and some different very specific frequencies are excited, these oscillations are perceived by the fibers which start also to vibrate, generating these strange lines in the

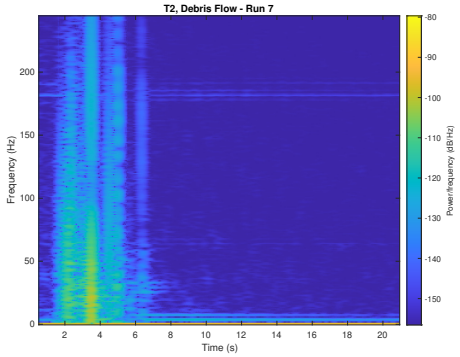

Figure 2.46: DF on T2 @ f_s^{DF}

spectrogram. This cannot be of course part of a time/frequency characterization of the events, being basically only a consequence of how the experiments have been made. All the spectrogram results of the other two fibers (T1, T3) lead to the exact same results of what has been found for T2, even with almost the same values of f_s^{SR}, f_s^C and f_s^{DF}. Despite this analysis was not successful since nothing worth of further investigation could be found in the spectrogram, it gave a fundamental idea: it could be possible to reduce the sampling frequency to obtain different, maybe better, results. This is reasonable since as we have seen the majority of the informations lies below 200 Hz circa, and it could be a good way to reduce Noise. Some tests have been made to see what happens if instead of decimating, taking one sample every one thousand, resampling of the data is applied. This has been done always by means of the Matlab function

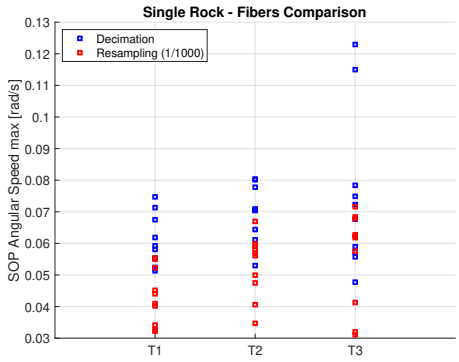

Figure 2.47: Noise, methods comparison

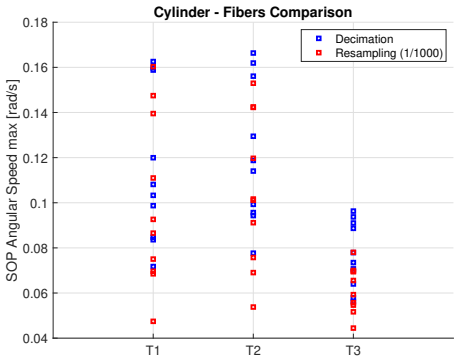

Figure 2.48: SR, methods comparison

`resample()`, of course on the Stokes parameters evolution, as decimation. This tool works very fine since it also applies a Low-Pass anti-aliasing filter which deletes artifacts coming from this operation (note that decimation does not apply this kind of filtering). The first approach to the idea has been to consider a resampling ratio equal to the

decimation one, in order to make the comparison fair, as a starting point. The results are in Figures 2.47, 2.48, 2.49 and 2.50 and are showing the maximum values assumed by the SOPAS in the case of resampling and decimation, for all the fibers and events. Note that for both the evaluations (decimation an resampling) smoothing is always applied, with $T_{mov} = 1$ second. The true achievement would be to find out that using a lower f_s instead of decimating, the Noise floor lowers, and the detection improves, and this is actually what can be recovered from the Figures reported. The use of a lower sampling frequency, which in this case has been simulated through Matlab, lowers of a very small quantity the maximum values of the SOPAS evolutions of the events, as shown in Figures. Despite of course having different values, they do not substantially diverge from the ones obtained in the decimation case. Figure 2.50 shows instead how

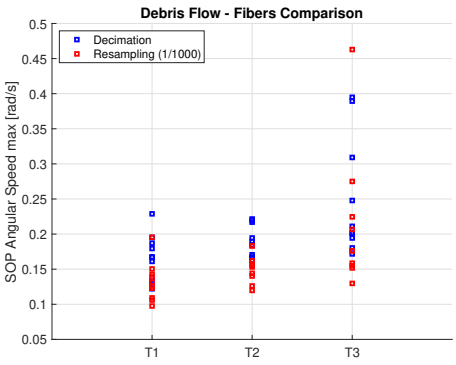

Figure 2.49: C, methods comparison

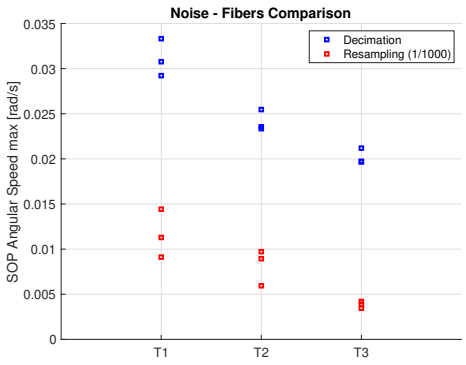

Figure 2.50: DF, methods comparison

Noise is much reduced, much more than the single events maxima. This means that by using a lower f_s, it's reasonable to get better detection performances, since events peaks are not far from the ones obtained with decimation, but Noise is lowered. The problem here is that by using a ratio of 1/1000 it's only possible to decimate, since the hardware of the Polarimeter does not allow to go under 95.4 Hz, and in this way the system would need to go at 48.83 Hz. For this reason is necessary to see what happens if resampling at a ratio which is bigger than the above one, but near to the minimum one allowed. A resampling ratio of 1/500 has been considered which grants a sampling frequency of 97.66 Hz, basically equal to the minimum one of the hardware. The results in Figures are extremely good: Figures 2.51, 2.52, 2.53 are showing that the evolutions obtained with resampling have even higher maxima with respect to decimation (note that decimation is always considered with ratio 1/1000), and Noise is instead reduced, as can be seen in Figure 2.54, even if of a smaller quantity with respect to the previous case. This means that by using this sampling frequency, or something very near to that as the 95.4 Hz achievable by the Polarimeter, the peak-to-noise distance can be improved a lot, resulting in a probably bigger safe area of choice of the $\Delta\omega_{th}$ and T_{mov} parameters, which at this point can be furtherly optimized. This is not the only advantage: a system like this, going this slow and still giving very good results, is potentially much less

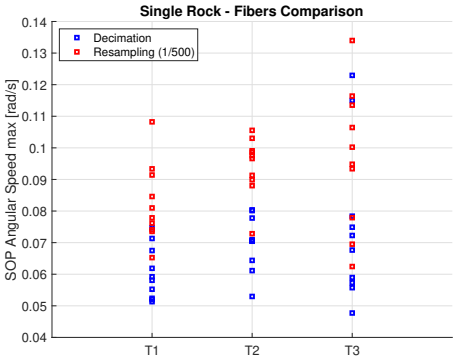

Figure 2.51: Noise, methods comparison

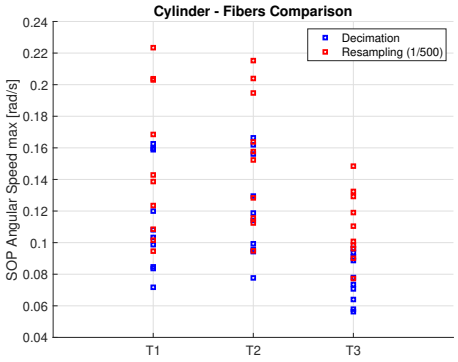

Figure 2.52: SR, methods comparison

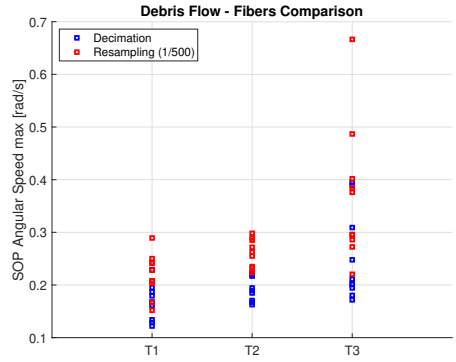

Figure 2.53: C, methods comparison

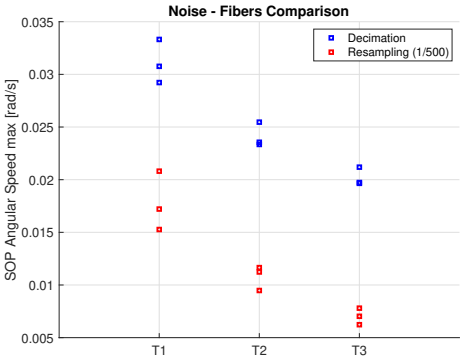

Figure 2.54: DF, methods comparison

complex to manage, and also cheaper. Detection performances should be then improved, and proofs are shown in Figures 2.55, 2.56 and 2.57. Detection maps showing the overall optimized performances for the single fibers are here reported, while the ones related to the single events are in Appendix A, labeled as "optimized maps". It is extremely clear how the detection algorithm improves a lot, since the green area is at least doubled for all the fibers, with respect to the previous non-optimized case, and considering also the yellow and orange one which are not the safest but still give decent results, the detection works fine for a very big number of couples of $\Delta\omega_{th}$ and T_{mov}. Another interesting thing to notice is the Noise floor, which is reduced a lot in particular selecting more superficial fibers: it is related to the red area below the green one, and in Figure 2.55 it is still quite consistent (perfectly in line with what has been said about Noise in Figure 2.54), while it tends to disappear from T2 to T3. Note that due to the bad detection of the Cylinder experienced by T3, the green area on the map is comparable to T2, with the advantage of having a lower Noise floor and consequenly basically no red areas. As very last thing before proceeding, in Figures 2.58 and 2.59 it has been reported the comparison between the overall performances of all the T fibers before optimization and after optimization. In

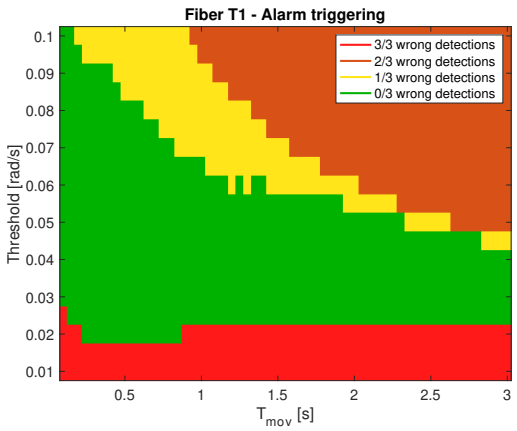

Figure 2.55: T1, optimized detection map

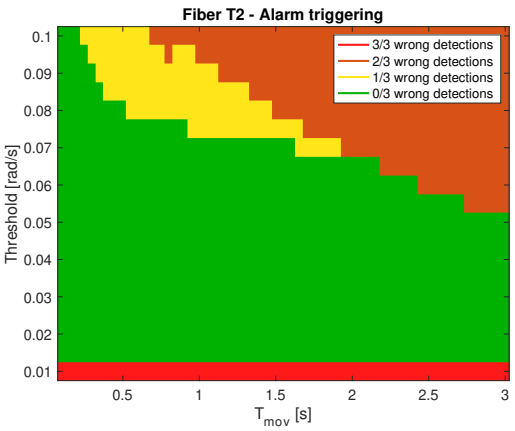

Figure 2.56: T2, optimized detection map

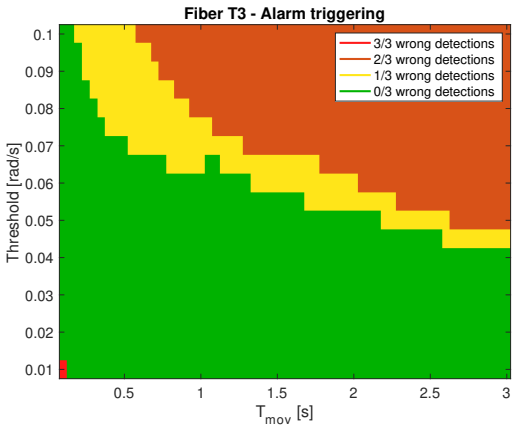

Figure 2.57: T3, optimized detection map

particular the plots have been obtained intersacating all the Figures like the one in 2.55, for all the three fibers, differentiating between the preliminary and optimized analysis. This result is extremely good and could be even identified as one of the most important result, if not even as the most important, one of the whole Thesis. The consequences are many, first of all the performances are clearly improved a lot: the yellow area in Figure is at least tripled with respect to the non optimized case, meaning that overall the events detection is made more sensible for all the fibers. This is obtained not making the system more complex but, on the contrary, making it simpler through the use of a very low sampling frequency. The processing chain in this way could go slower, which translates in simpler and cheaper hardware. Theorethically this system could grant nice detection performances and still remain not complex and cheap, which is exactly what

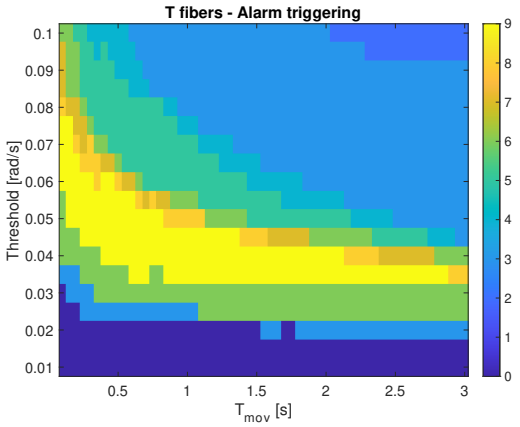

Figure 2.58: T fibers map, non optimized

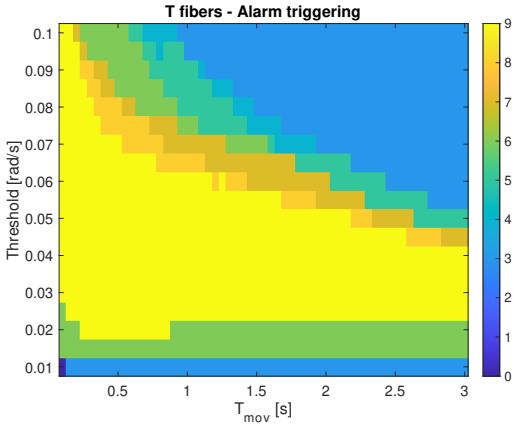

Figure 2.59: T fibers map[1], optimized

we had in mind as goal of this Thesis.

2.4 Rockfall barrier Fiber performances

Up to this point, the analysis that has been carried on was about the fibers buried in the soil. These have not been the only ones taken into consideration: a simulation of how a fiber wrapped around a rockfall barrier would react has also been tested, in order to understand if an event could be characterized in terms of danger even by this kind of protection system, not only by underground cables. They would be in some way easier to install, although much more exposed to damages generated by events of any kind. Four different configurations have been tested, named conventionally RP1, RP2, RP3, RP4. The acquisitions for RP1 and RP2 were made by using the non optimized sampling frequency (ATE = 11 and ME = 20), so resampling has been adopted, while for RP3 and RP4 the optimized choice for the f_s has been selected. The tests performed are

Event	*# tests*	*Configurations*
small SR	20	RP1, RP2, RP3, RP4
Noise	3	
Total	23	

Table 2.4: Tests performed per configuration

[1]The colours scale in Figures 2.58 and 2.59 represents in the yellow areas the regions where all the three fibers are able to correctly detect each one of the overall 30 events (so, yellow means 90 events correctly detected). One unit on the colorbar corresponds to one set of 10 events among SR, DF or C that, for a single fiber, has been correctly detected. This means that if even just one event among the set of 10 is not correctly detected, it counts as zero.

reported in Table 2.4, consisting mainly in launches of a small rocks, a bit smaller than the SR and of the rocks composing the Debris Flow, in order to be sure not to damage the structure, which is just a metallic net, able to distort in an elastic way very similarly to the real world case. One final note: the range of thresholds in the maps reported below is different from the one adopted for the T fibers, since the physical event to be monitored is quite different from the situation studied in the previous Section. This is in fact a quite different case study, in which this range seemed more suitable to make the performances of the different configurations emerge in a more meaningful way.

2.4.1 RP1 Configuration

The main idea behind this configuration was to start with a generic arrangement, just wrapping the fiber around the central part of the grid, to see how it reacted, and then changing something to that in order to possibly make it more and more sensible. The specific arrangement is reported in Figure 2.60, while Figure 2.61 shows some SOPAS evolutions along time. This configuration turned out to be the least sensible of all, and this can be seen comparing the SOPAS characterizations of the next configuration. The

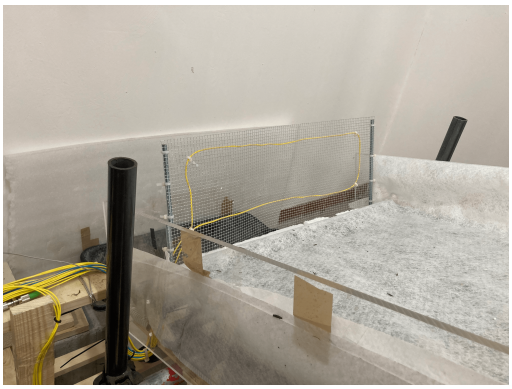

Figure 2.60: RP1 configuration

peaks of the angular speed are not that high, in particular with respect to the following ones, so a too high threshold could represent a problem. This is also confirmed by the map in Figure 2.62, where the green area is actually quite spread over the lower left part, but since the event is actually almost hitting the “naked” fiber, a much higher sensitivity should be expected, with respect for example to the T fibers case. Overall, the green area is actually big, so this fiber is in the end very sensible on the detection of the event tested, even if the scenario could be improved as shown for the configurations below. One side note: the peaks that can be seen in the final parts of the evolutions are caused by the rock bouncing on the barrier. This could also be seen in some of the following configurations.

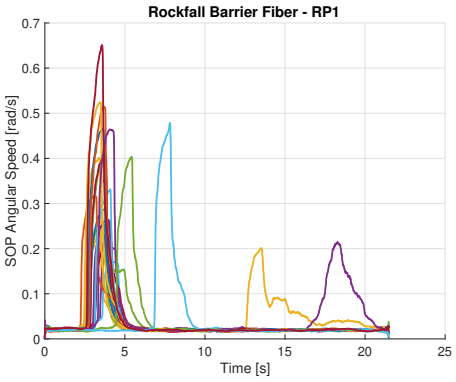

Figure 2.61: RP1, SOPAS behavior

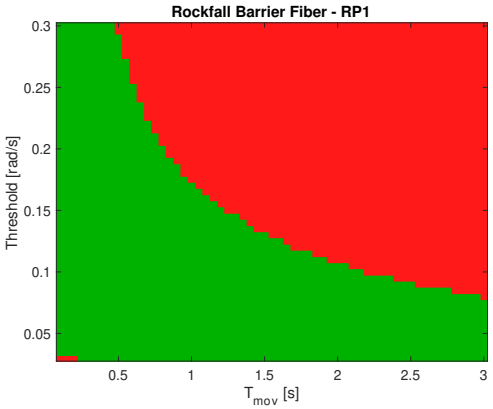

Figure 2.62: RP1 detection map

2.4.2 RP2 Configuration

The specific arrangement of this one is in Figure 2.63, and it is the result of the first adjustments made on RP1 to enhance the performances. The SOPAS evolutions are reported in Figure 2.64, and peaks are clearly much higher with respect to RP1, more than doubled. This means that the detection map should be improved, in Figure 2.65,

Figure 2.63: RP2 configuration

and actually almost all the couples of $\Delta\omega_{th}$ and T_{mov} that can be chosen are safe. This enhanced sensitivity is probably due to the fact that the majority of launches hit the central part of the barrier, or the most lateral one where two portions of the same fiber are almost overlapping: the collision would then result in an increased intensity. It has been proved that despite this very nice behavior, that happens the majority of times, if some rocks hit in the wrong part of the grid, where maybe the portion of fibers are

not overlapping or not even near one to the other, the event is very poorly detected. This could represent a quite big problem since it gives space dependency, but overall the performances are extremely good.

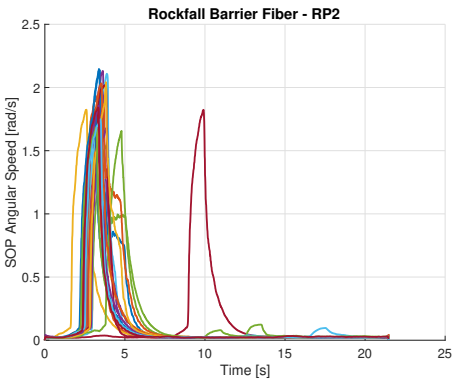

Figure 2.64: RP2, SOPAS behavior

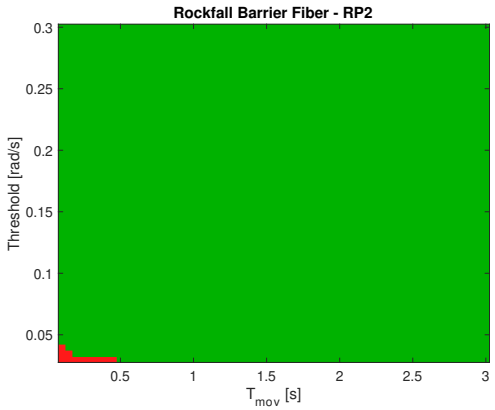

Figure 2.65: RP2 detection map

2.4.3 RP3 Configuration

For what concerns instead the RP3 configuration, in Figure 2.66, the idea was to see if the vibrations transmitted by the metallic grid to a fiber placed on its perimeter could be of any help to improve the sensitivity: maybe the vibrations on the most external part

Figure 2.66: RP3 configuration

of the barrier are stronger, so the fiber could be more sensible. The results are reported in Figures 2.67 and 2.68, where the behavior is very similar to the one of RP2 as the detection performances. The SOPAS peaks are even higher, basically doubled, even if the Noise has an higher level, as the increased lower red area shows. Due to the much higher peaks, if choosing a wider (higher $\Delta\omega_{th}$ particular) range of threshold values, the

green area on the map of RP3 should be much bigger than RP2, meaning, despite the Noise level, that this configuration is much more sensitive to these kind of events.

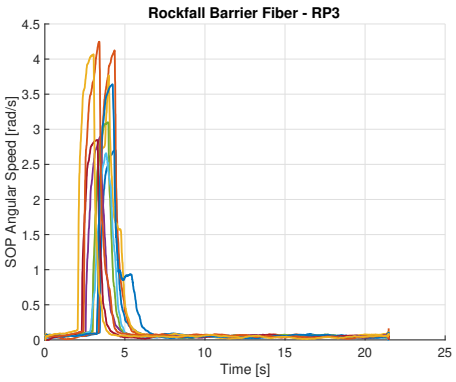

Figure 2.67: RP3, SOPAS behavior

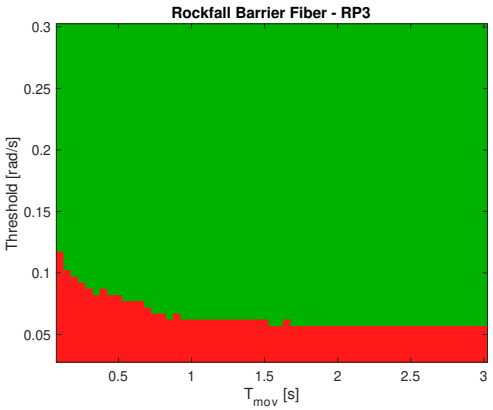

Figure 2.68: RP3 detection map

2.4.4 RP4 Configuration

As last configuration, RP4 in Fig. 2.69 has been considered. The goal of this configuration

Figure 2.69: RP4 configuration

was to see if instead of catching the vibrations transmitted from the metallic grid to the fiber, this one could be able to catch the vibrations transmitted to the barrier pole, which maybe could give an higher sensitivity. The SOPAS behavior in Figure 2.70 shows that the peaks are quite similar to the ones in Fig.2.64 in terms of intensity, a bit lower and less constant, leading to a map in Fig. 2.71 which has some reasonable red areas in

the upper right part. The Noise effect is much higher, and could be caused by the very

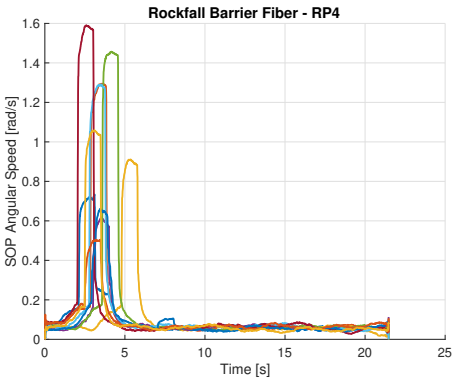

Figure 2.70: RP4, SOPAS behavior

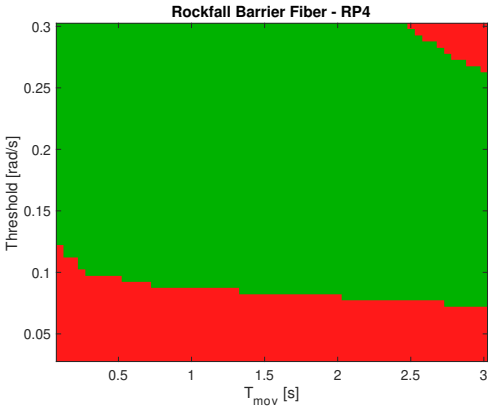

Figure 2.71: RP4 detection map

pronounced bendings that are needed on the fiber to wrap it around the pole, as Figure 2.69 shows. A very interesting thing that can be noticed from the angular speed behavior for this specific case is that the peaks are flatter, because of an extended vibration in time with respect to the one experienced by the grid. Despite this configuration is very interesting and give very good results, configurations RP2 and RP3 seem to be the most reliable of all.

Chapter 3

Real time algorithm

The previous Chapter has been devoted to the description of the main results through the usage of an Offline version af the algorithm, thanks to which it has been possible to set the optimal parameters and get the best out of the samples acquired. In this Chapter the main goal is to exploit the results obtained up to now, and apply them in a continuous time version of basically the same algorithm. In this way we get closer and closer to the generation of alarms triggered some splits of a second after the actual event occurrence.

3.1 Algorithm description and working principle

The algorithm used follows the block scheme in Figure 3.1 and it is realized through the Matlab script, reported in in Appendix C. This block scheme isn't actually that different

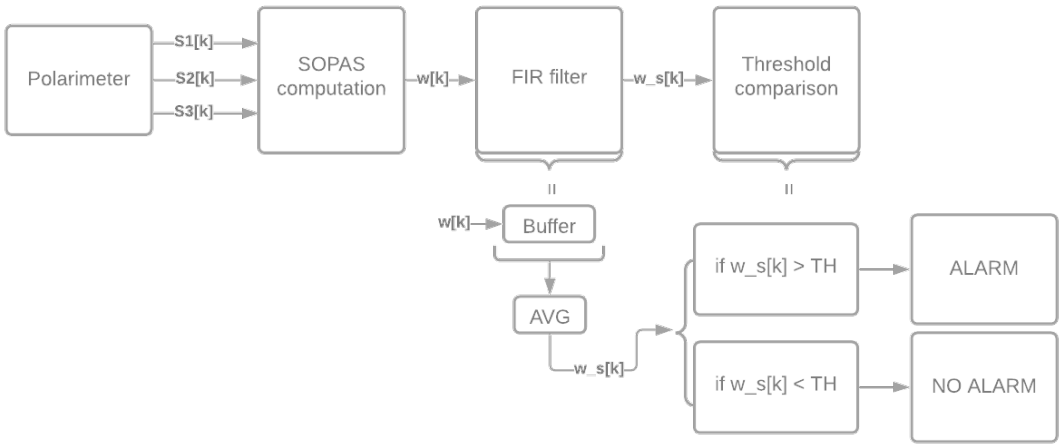

Figure 3.1: Matlab algorithm block scheme

from the ones described and optimized up to now. The main improvement is the fact

that this is specifically tailored to work in real time, but the working principle is not changed at all: smoothing on the SOPAS time evolution is always applied, along with the threshold to decide if the alarm should be triggered or not. The concepts introduced in the Offline mode analysis (such as $\Delta\omega_{th}$ and T_{mov}) are then of fundamental importance to make the detection happen correctly in Real-time mode. By looking in detail to the block scheme in Fig. 3.1, this time the real device is considered, the Polarimeter. This particular model is able to be interfaced very easily with Matlab, through which is possible to read some specific addresses containing the Stokes parameters (S_1, S_2, S_3), sample by sample, and then compute the SOPAS value at a specific time instant i. Note that since it has been demostrated that the lowest sampling frequency achievable by this instrument is also the optimal one, the Polarimeter will sample at $f_s = 95.4$ Hz, and fill the memory addresses at the same speed. The FIR filtering, which is decomposed in the scheme in Figure, is actually what does the smoothing: a buffer of length T_{mov} (in samples) needs to be filled before outputing the $\omega_s(i)$ value, that will then be compared to the threshold and give an alarm if necessary. As previously said, the FIR filtering introduces delay equal to T_{mov} seconds, since it needs to first fill the buffer, and then output the smoothed SOPAS sample.
In order to preliminarly try out the algorithm, it has not been tested on the real model right away, but some checks have been done on the already acquired data, just to be completely sure that the performances are exactly equivalent (or maybe better) to the versions of the algorithm seen in the previous Chapters, which gave optimal results. The Real-time mode version of the algorithm has been then tested as if it was meant for the post-processing, on the same dataset used in the previous Chapter for the T fibers: single events SOPAS evolutions, resampled at the right frequency. This means that, instead of having the Polarimeter as a real time samples source, a three rows matrix containing the time evolutions of the three Stokes parameters of interest has been extracted from the .bin files, and then read sample by sample, as if it was the actual device. In this way

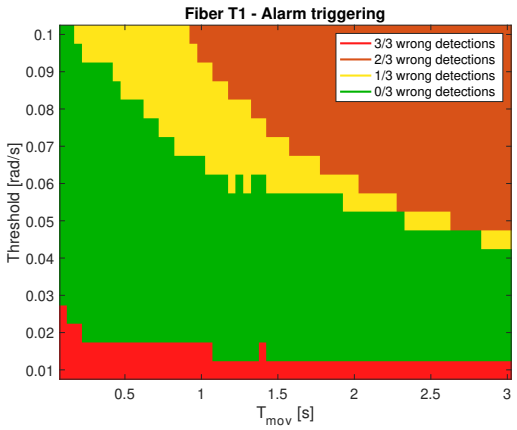

Figure 3.2: T1 fiber map, real time algorithm

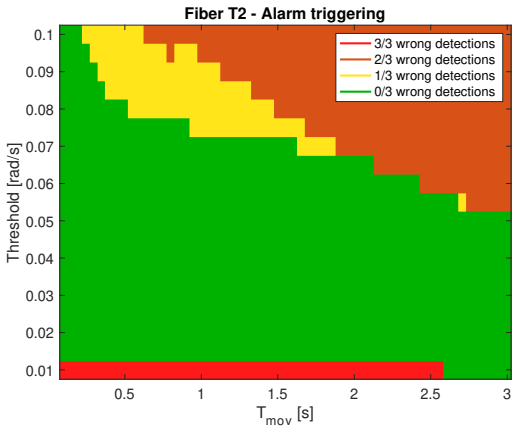

Figure 3.3: T2 fiber map, real time algorithm

the testing would be identical to actually applying in real time the algorithm, with the advantage of seeing in advance the actual performances. In Figures 3.2, 3.3 and 3.4 the performances of the algorithm applied to the evolutions of the events, fiber per fiber, are reported. From these Figures is clear how the performances are almost identical to the ones obtained with the previous version of the algorithm (Figures 2.55, 2.56 and 2.57), even a bit better since the red area below the green one seems to be a bit reduced, which translates in a lower Noise level, and less False Alarms: this is confirmed by Fig. 3.5 too, where the yellow area is even bigger or at least comparable with the already optimized case in Fig. 2.59. The two analysis are then perfectly coherent.

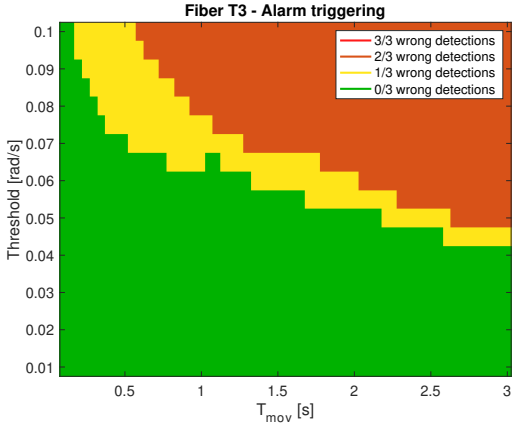

Figure 3.4: T3 fiber map, real time algorithm

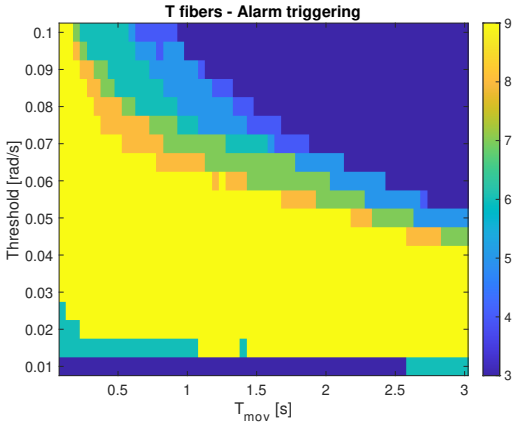

Figure 3.5: T fibers map, real time algorithm

Prior to the real time implementation, another kind of test has been carried on some long acquisitions, where a serie of different events have been generated on T, Uc and Ul fibers. The goal of this analysis was to apply the algorithm on acquisitions that would be very similar to the a Real-time mode situation, to see if the alarm signal triggered correctly, and also test if the maps obtained up to now are giving reliable results on the detection of events happening in serie. The long acquisitions on the different fibers are all identical to the one in Figure 3.6, which is the only one reported here, the others are in Appendix B. While the sampling frequency is the optimal one, $f_s = 95.4$ Hz, the total acquisition time is instead increased to 22.9 minutes. From now, the analysis will proceed only on T3 for simplicity, but the exact same results can be obtained for all the other fibers. T3 detection map in Figure 3.3 tells that the green area is a safe one, so choosing:

- $\Delta\omega_{th} = 0.04$ rad/s, $T_{mov} = 0.3$ s, Correct Detection should happen (Figure 3.7).
- $\Delta\omega_{th} = 0.08$ rad/s, $T_{mov} = 2.5$ s, Missed Detection should happen (Figure 3.8).
- $\Delta\omega_{th} = 0.005$ rad/s, $T_{mov} = 0.3$ s, False Alarms should happen (Figure 3.9). Note that in the map no red area due to Noise appears, but it's reasonable to think that,

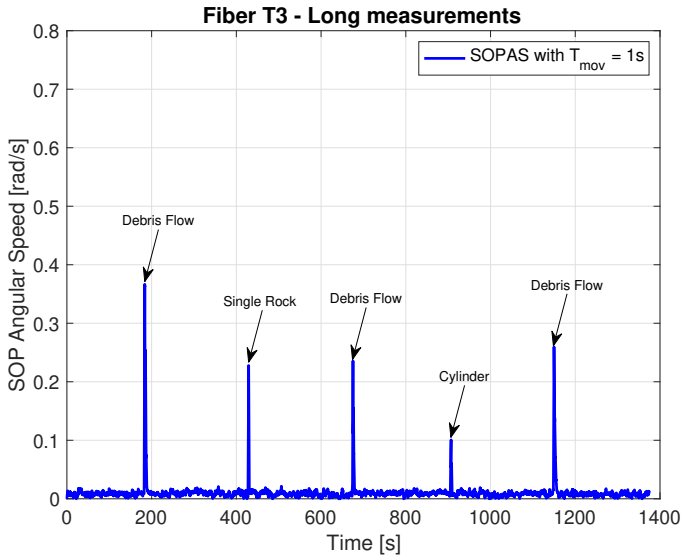

Figure 3.6: T3 long acquisition

given all the analysis done up to now, 0.01 rad/s is very near to the Noise level, so anything below that should trigger a False Alarm condition.

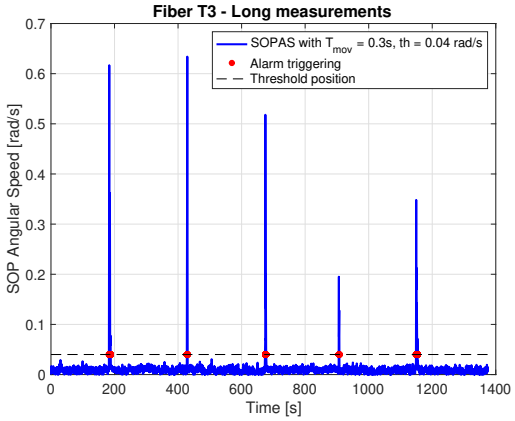

Figure 3.7: Correct Detection

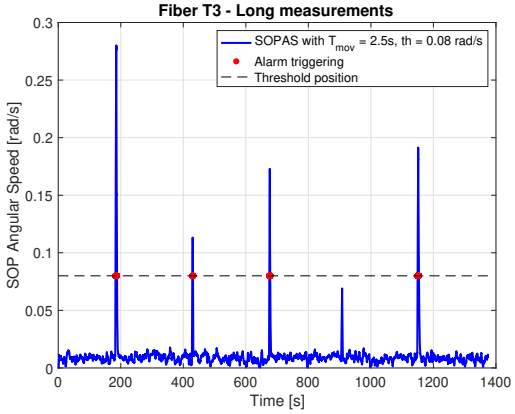

Figure 3.8: Missed Detection

From Figures 3.7, 3.8 and 3.9 is clear that the algorithm works extremely well and it is very coherent, even if not perfectly, with the detection maps obtained for the single events. For these specific long acquisitions it seems that the Noise level is a bit higher than expected, maybe due to the fact that the maps are obtained through the Matlab `resample()` function of the single events evolutions, while the long acquisitions are obtained by setting the optimal sampling frequency on hardware. Also, in Figure 3.8 the alarm should trigger only for one kind of event, according to Fig. 3.4, but instead it triggers for a Cylinder and a Debris Flow too: this could happen, the maps are based

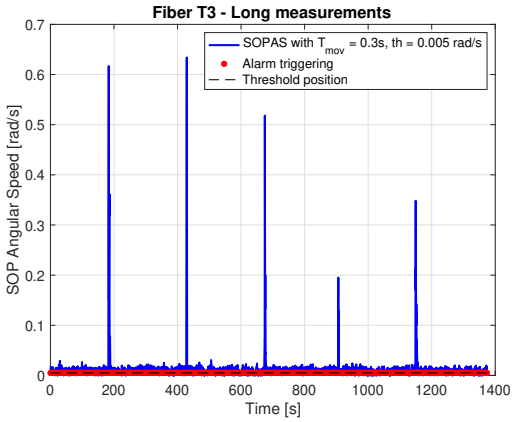

Figure 3.9: False Alarm

on a small, limited batch of events, which clearly does not account for every possible case, they are just very useful to determine if the algorithm could work or not, and in which cases it can work, to have an idea on how to set the parameters properly in the continuous time system. In order to get an even deeper analysis and see how the series of events is detected by the algorithm, another kind of map has been created, also to show for what parameters the algorithm applied to the series of events gives 100% alarm correctness. As usual, two areas can be identified: a green one and a red one. The green one means that for that specific couples of $\Delta\omega_{th}$ and T_{mov}, all the five events are correctly detected by the algorithm. The red area means instead that something wrong happened, so for that couple of parameters, not exactly five dangerous events have been detected, meaning that in the end False Alarm or Missed Detection happened. Since this

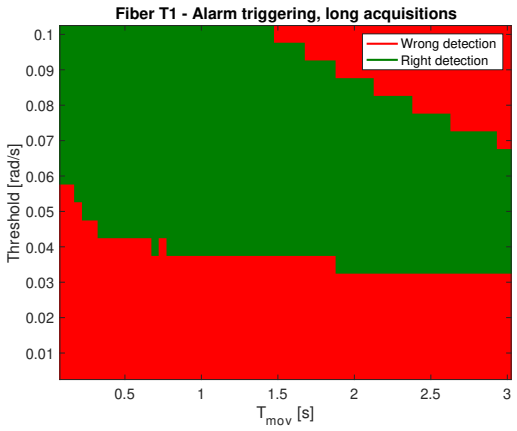

Figure 3.10: T1, Long acquisition detection map

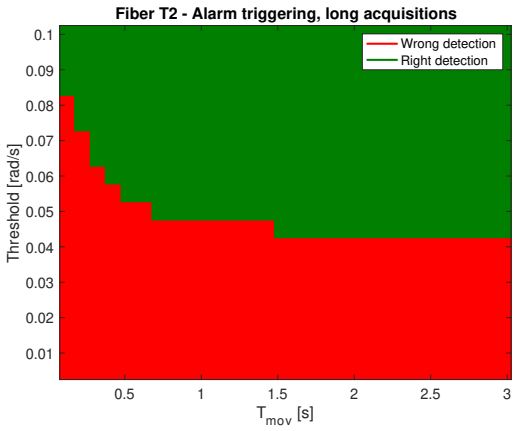

Figure 3.11: T2, Long acquisition detection map

analysis is very important, all the maps for all the nine fibers have been reported and commented, starting from the T ones, in Figures 3.10, 3.11 and 3.12. These are showing

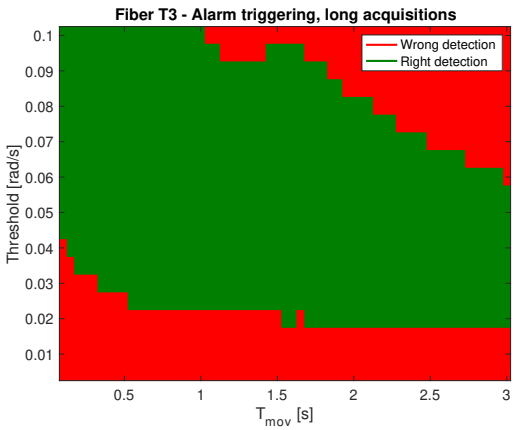

Figure 3.12: T3, Long acquisition detection map

quite different performances with respect to the single events situation, in Figures 3.2, 3.3 and 3.4. This is reasonable: as highlighted above, the number of events tested are a small batch, and the level of Noise on long acquisitions seems to be increased with respect to the resampled case, which perfeclty reflects on these maps. Furtherly, the

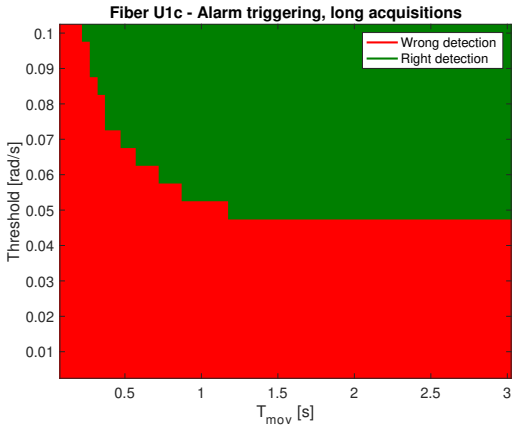

Figure 3.13: U1c, Long acquisition detection map

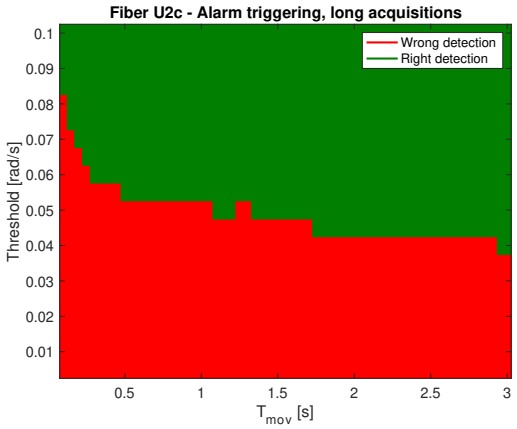

Figure 3.14: U2c, Long acquisition detection map

goal of this analysis is not to see if the single events maps are right or wrong, but deeply characterize the algorithm and find, by conveying all the different results, an optimal choice of values granting Correct Detection all the time, in a Real-time mode application. Despite this, on T fibers the algorithm seems to perform very good on the detection of all the five events: green area is spread. For what concerns the Uc (Figures 3.13, 3.14, 3.15) and Ul (Figures 3.16, 3.17, 3.18) fibers, their behavior is a bit worst with respect to the T. In particular it seems that the red area is much bigger for lower values of threshold, meaning higher Noise, but the algorithm does not perform bad at all over these ones

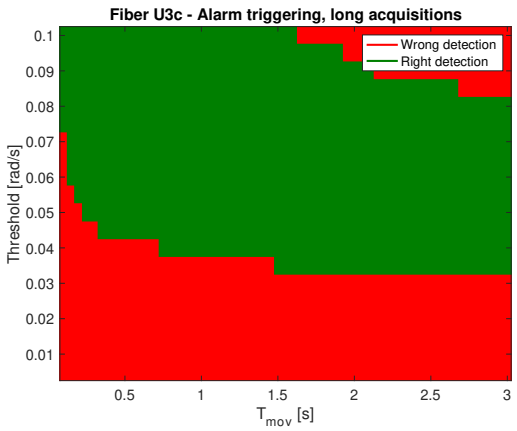

Figure 3.15: U3c, Long acquisition detection map

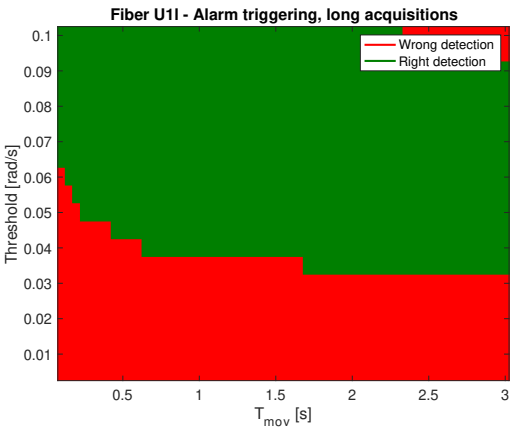

Figure 3.16: U1l, Long acquisition detection map

either: the maps are in a lot of cases comparable with the T ones. The U fibers are quite critical since they are extremely space dependent: if an event is not happening exactly along the fiber, the detection might be a lot compromised. Despite this a lot of values can be chosen in order to have a correct detection of all the five events for the majority of cases. In Figures 3.19 and 3.20 is reported an example of alarm triggering

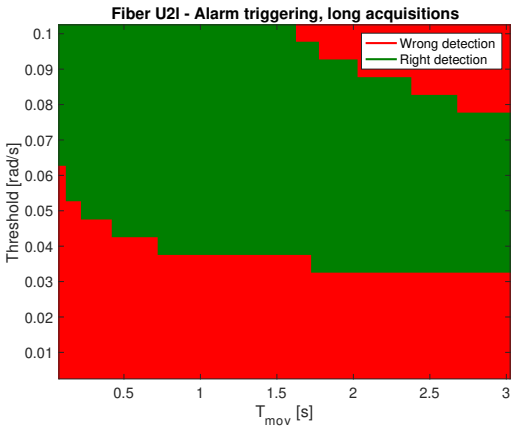

Figure 3.17: U2l, Long acquisition detection map

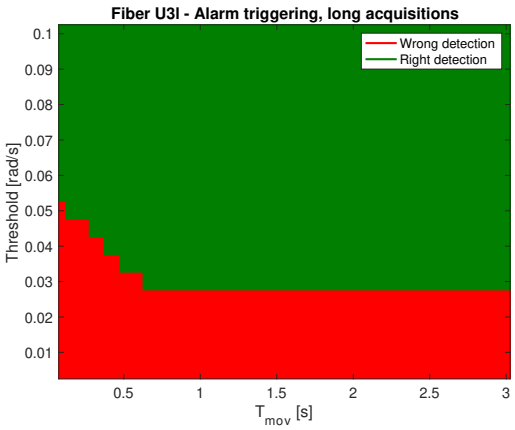

Figure 3.18: U3l, Long acquisition detection map

on U2l for a too high and too low threshold, in order to show the reliability of these maps, and also give an idea of what should happen, very similarly to the other fibers, in the Ul case too. The $\Delta\omega_{th}$ and T_{mov} values have been selected looking at Fig. 3.17, and chosing one couple over the red area and one over the green. Gathering all these different results, it has been then demostrated that the algorithm works, and a quite big set of parameters can be chosen to make it work perfectly. Now that all the key points

have been characterized, in the next Section some results about the implementation of the algorithm in the real model is provided.

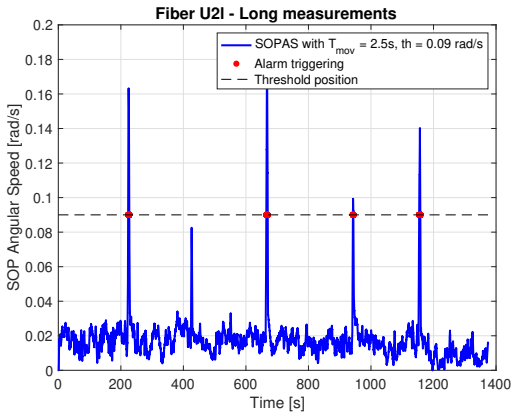

Figure 3.19: U2l, Missed Detection

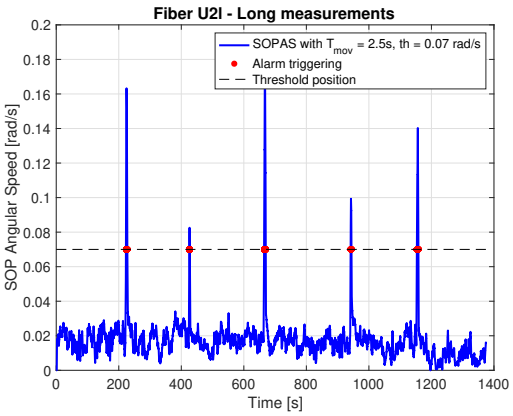

Figure 3.20: U2l, Correct Detection

3.2 Real-time mode algorithm implementation

In this Section, the results about the implementation of the block scheme in Fig. 3.1, on the actual real devices with the real downhill in scale model, so in Real-time mode, are reported. The features of the Matlab scripts implementing the real time algorithm are the following:

- Real time plotting of the SOPAS time evolution, sample by sample, of what is happening on the fiber connected to the model. When the system is in a quiet state the color of the samples is black, when instead some unsafe condition is detected, the color changes to red (basically the value of the sample is above threshold).
- Emission of a sound (similar to a “beep”) when the SOPAS goes from below to above threshold, and of a different one when the opposite case happens.
- Display of an image representing a traffic light, which is green by default and changes its colour to red for the whole period when the SOPAS stays above threshold, so in a non-quiet state.

The idea of this system is to realize something which simulates a miniature alarm system: this would trigger an acoustic alarm and a traffic light to stop cars transiting in a mountain area liable to unsafe events, and potentially save lives. Of course the real time plot of the SOPAS is useless in a real world application, but it is very interesting to evaluate the continuous time behavior. All the real time tests have been done with T3, and all with the same time serie of events: first Single Rock, then Cylinder and as last one Debris Flow. The Figures 3.21, 3.23 and 3.22 report exactly the SOPAS time

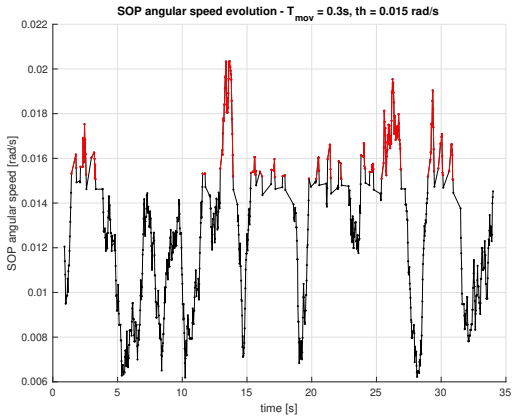

Figure 3.21: FA, real time acquisitions

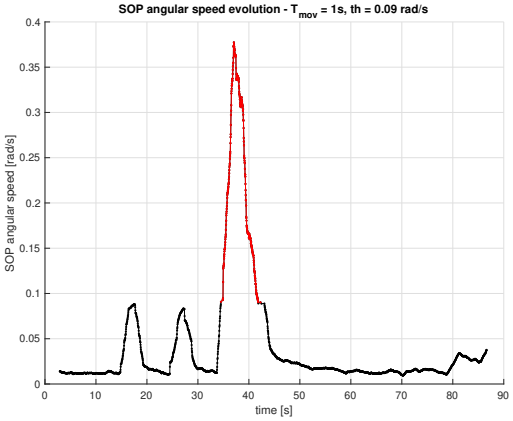

Figure 3.22: MD, real time acquisitions

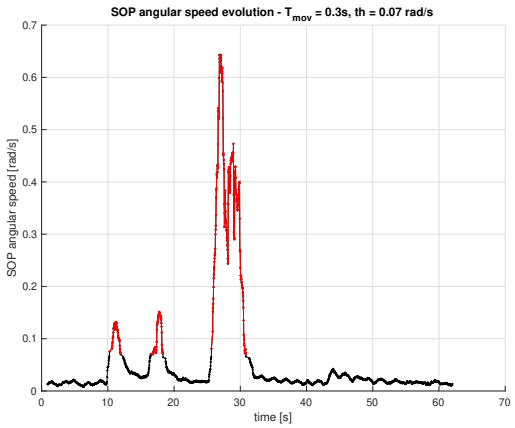

Figure 3.23: CD, real time acquisitions

evolution that Matlab shows when the script is launched, and the events are generated. Referring to the maps reported in the previous section of this Chapter, we show some important examples from which it's reasonable to say that:

- Using $\Delta\omega_{th} = 0.015$ rad/s and $T_{mov} = 0.3$ seconds, False Alarm should happen, see Figure 3.21. In this case no event is generated, since it's enough to show that the Noise oscillations are going up and down around the threshold.
- Using $\Delta\omega_{th} = 0.09$ rad/s and $T_{mov} = 1$ seconds, Missed Detection should happen, see Figure 3.22. Note that actually in Figures 3.4 and 3.12 this point is a bit on the limit between safe and unsafe area, in fact the peaks of SR and C are very near to overcome the threshold, witnessing the precision of the Offline mode analysis done previously.
- Using $\Delta\omega_{th} = 0.07$ rad/s and $T_{mov} = 0.3$ seconds, Correct Detection should happen, see Figure 3.23.

The functioning of the algorithm even on the real scale model is then confirmed, giving also a very good proof of coherency of the analysis done up to now: it is very clear that if choosing some optimal parameters as in Fig. 3.23, and there are a lot to chose from as seen, the detection happens perfectly, with no particular problems.

3.3 Limits and improvements of the real time system

One of the main constraint that needs to be taken into account for a real scale application is the delay introduced by the processing chain. Actually, this one is absolutely not complex, so it does not require a big computational effort, but still it will introduce some delay due to the FIR filtering, which would need to fill a buffer of $\lfloor \frac{T_{mov}}{f_s} \rfloor$ samples, before averaging them. Being f_s fixed, the bigger the averaging window, the longer is the time to wait before the filter outputs a smoothed sample, and as a consequence before the alarm triggering. This is anyway something that needs to be accounted, but not an unsolvable issue. A behavior that instead represents a true limitation of this

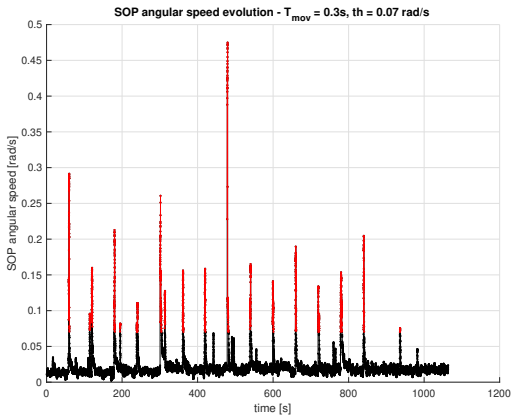

Figure 3.24: SR serie, real time acquisition

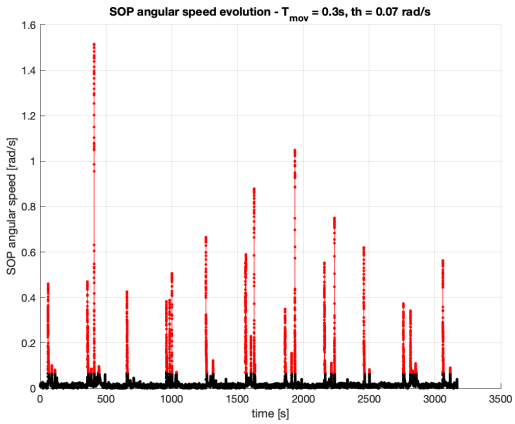

Figure 3.25: DF serie, real time acquisition

approach is the fact that up to now the analysis has been carried considering three different events labeled as dangerous, versus a non dangerous one which is the Noise generated "naturally" by the fiber: this is not an actual real life condition, a lot of non dangerous events other than Noise could stimulate the fiber, as people, animals passing on it or maybe some non dangerous natural conditions (maybe wind, rain, water flows or some material falling from trees). This other kind of disturbances are very difficoult to be simulated on the model, almost impossible, but it is not difficoult to understand that the system would be completely not robust to them. This is confirmed by what is shown in Figures 3.24 and 3.25, which report a real time acquisition of respectively a serie of SR launched one after the other every 60 seconds and DFs launched one after the other every five minutes. From this evolutions is clear that some seconds after each peak,

some spurious ones appear, not due to an event happening since each event is spaced of the same constant time, but to someone accidentally touching the structure or the fibers connected to the set up table, acting as completely random Noise not due to the fibers themselves (so conceptually similar to the external natural factors that have been mentioned above). These plots are quite comforting in a sense, since they are a further validation of the correctness of all the analysis done previously about the parameter choice and show the very good functioning of the alarm system, but also quite bad, since it's clear how by using this algorithm alone it is absolutely impossible to distinguish one kind of alarm from the other, so a lot of False Alarms would be generated and not recognized. In order to bypass this issue, a change of approach has been considered: since a Debris Flow is actually the most realistic event that has been simulated and it has an intensity on a quite different scale with respect to the other two, it's perfectly reasonable to think that this could be considered as the main event, while the other three (SR and C events, and of course Noise) as just non dangerous events generating FAs. Computing the maps with these premises, it has been obtained what is in Figures 3.26, 3.27 and 3.28. Note that the threshold range has been increased to have a better

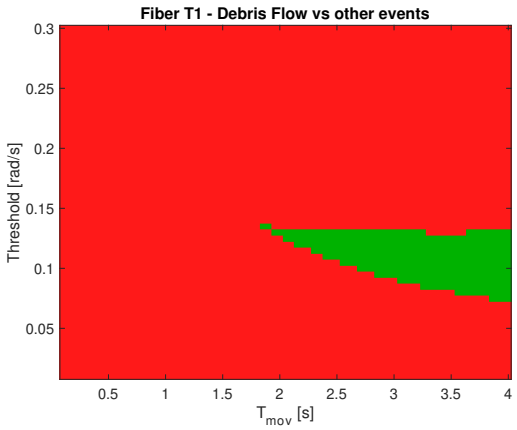

Figure 3.26: DF detection map on T1

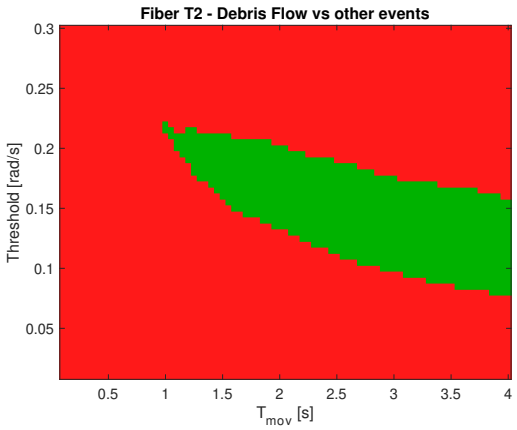

Figure 3.27: DF detection map on T2

visibility on the maps, but still it is quite clear that the algorithm still works, even if not as good as before. Of course a lot of low thresholds that were previously acceptable are now not allowed, since the intensity of Single Rock and Cylinder is much higher than just fiber Noise. T3 in particular has not big problems, even if for small values of T_{mov} the use of a quite high threshold is needed, but detection happens correctly in a lot of safe cases. Much more troublesome are T1 and T2: safe areas start at least from 1 or 2 seconds of window length, which starts to introduce in the in scale model quite a big delay between the alarm triggering and event happening. This approach could then be working quite safely, especially for T3, but still it would be very nice to have something telling if the event happening is a dangerous one or due to something else: in other words, to really make this system work properly in a real world situation, it should be able to differentiate fiber Noise with spurious noise and with events, but also events

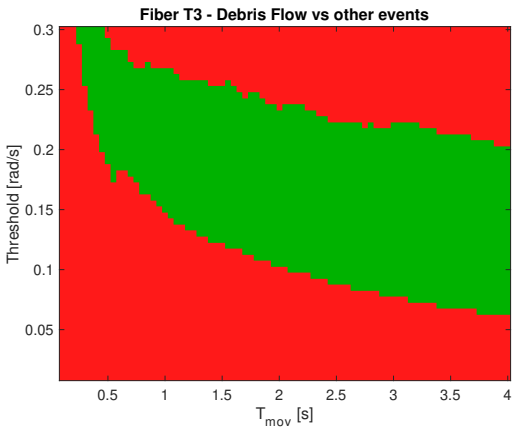

Figure 3.28: DF detection map on T3

among themselves, so tell if something that just occurred was caused by a DF, a SR or a C. This, as we said is almost impossible to do, but one very weak possibility worth of investigation could be acting on the duration time of each event, which is changing very much from one to the other. The way that has been adopted to evaluate the length of each event is to count for how much time the alarm is on. This of course depends on both T_{mov} and $\Delta\omega_{th}$ as reported in Figures 3.29 and 3.30. In these ones is shown how

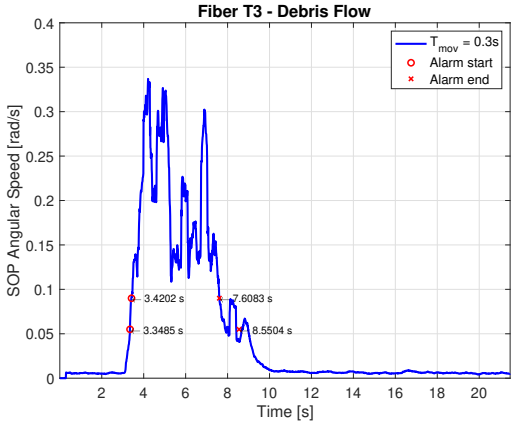

Figure 3.29: Duration shrinking for $T_{mov} = 0.3$ s

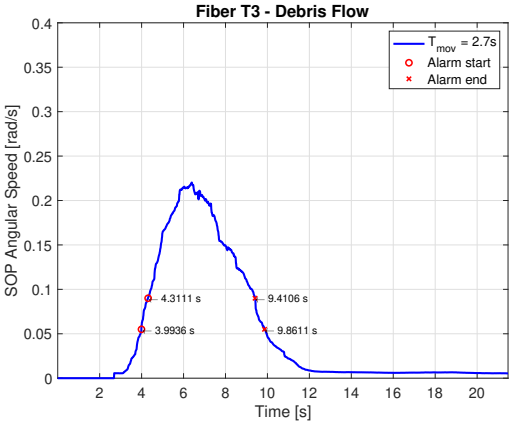

Figure 3.30: Duration shrinking for $T_{mov} = 2.7$ s

for two different thresholds (one at 0.055 rad/s and the other at 0.09 rad/s) per Figure, fixed T_{mov}, the duration changes. Intuitively, if adopting decreasing thresholds fixing the averaging window, the duration will increase since the peak gets further, and it would be possible to detect the event from the very start up to the very end, the only limit is FA triggering. Viceversa, adopting decreasing T_{mov} values and fixing $\Delta\omega_{th}$, the duration would decrease, since the effect of smoothing would spread the event SOPAS over time.

For these reasons the analysis that needs to be carried on needs to account for all the possible values of $\Delta\omega_{th}$ and T_{mov}. The duration, depending on these two values, for all the three events has been computed as a three dimensional problem, and reported in Figures 3.31, 3.32, 3.33 and 3.34. In particular the algorithm exploited to compute

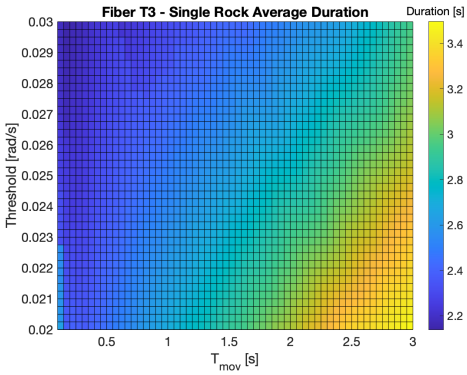

Figure 3.31: Duration map for SR on T3

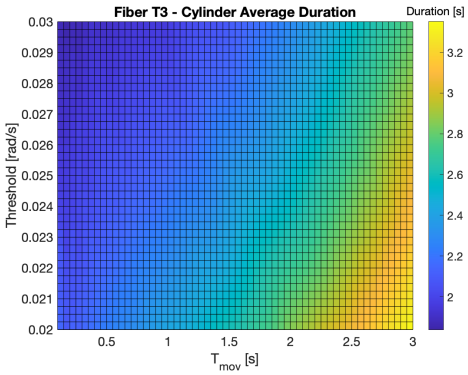

Figure 3.32: Duration map for C on T3

the duration maps in Figures provides an average duration, given the ten evolutions of the specific event selected: for each $\Delta\omega_{th}$ and T_{mov} value the duration of each event is computed, and then averaged over the total number of runs considered, which is ten. As

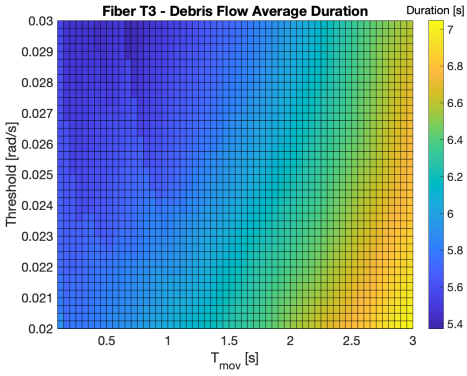

Figure 3.33: Duration map for DF on T3

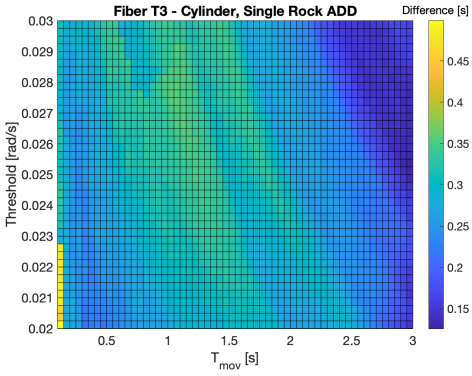

Figure 3.34: ADD, C and SR on T3

it was previously guessed the Debris Flow (Figure 3.33) has always an almost doubled duration with respect to the other two events (Figures 3.31 and 3.32), which instead are extremely similar in terms of duration: by taking the difference between their duration for each couple of parameters chosen (Average Duration Difference, ADD), Figure 3.34 is obtained, where the duration difference span goes from 0.15 seconds up to 0.45 seconds, way too small to get a good differentiation of the Single Rock an Cylinder events. As always, the Debris Flow should almost always be recognized exploiting this kind of analysis, but the other two are almost impossible to distinguish. In order to test the

precision of this approach, a Matlab script acting as an event identifier has been written. This one should be able only to distinguish DF from the other two events which are considered as if they were belonging to the same class, and from noise of any kind. The algorithm flow works as follows:

- SOPAS is given as input to the processing chain, computed with a previously chosen $\Delta\omega_{th}$ and T_{mov}.
- Depending on the $\Delta\omega_{th}$ and T_{mov} values, through the map in Fig. 3.33, a duration threshold ($d_{th,DF}$) value for the Debris Flow event is chosen, along with a margin. The margin is computed as a multiple (α) of the standard deviation (μ_{DF}) of all the ten durations of each single DF events for the couple of values selected. The same is done for the Single Rock and Cylinder, but considering them as a unique event ($d_{th,CSR}$, μ_{CSR} are the reference parameters). In particular, being the duration of these last two basically identical as shown in Fig. 3.34, a common $d_{th,CSR}$ has been chosen as the average between the two, selected from the maps in Figures 3.31 and 3.32.
- Duration is computed by counting for how much time the alarm stays on, and depending on if this value falls inside the range in Eq. 3.1, computed for the specific event, it will be classified as DF, as Cylinder or Single Rock, or as just noise. This last case is triggered if the duration does not fit into any range computed, so it should include everything other than the events labeled as dangerous.

$$[d_{th,event} - \alpha\mu_{event};\ d_{th,event} + \alpha\mu_{event}] \tag{3.1}$$

This identification algorithm has been tested in the Offline mode considering the evolutions sample by sample, fixing the parameters as described in the bullets above, in Table 3.1. No other tests are reported, since the performances are hardly improved even

T_{mov} [s]	$\Delta\omega_{th}$ [rad/s]	$d_{th,DF}$ [s]	$d_{th,CSR}$ [s]	μ_{DF} [s]	μ_{CSR} [s]	α
1.5	0.025	5.966	2.492	0.465	0.390	2

Table 3.1: Parameters set

by changing parameters. The choice of T_{mov} and $\Delta\omega_{th}$ does not change the situation that much, since the $d_{th,DF}$ and $d_{th,CSR}$ are always very different. Different values of the coefficient α have been tested, but in the end the one in Table seems to give the best results. This algorithm has been tested on some other very long acquisitions, where a serie of a lot of Debris, Rocks or Cylinder have been launched, one after the other with a constant time difference between them. The situation is very similar to the one reported in Figures 3.24 and 3.25, in which not all the spikes are events but also spurious noise exists. In Figures 3.35, 3.36, 3.37, 3.38, 3.39 and 3.36 is reported on the left, highlighted in red, what is the event peak that should be recognized versus the generic noise in black.

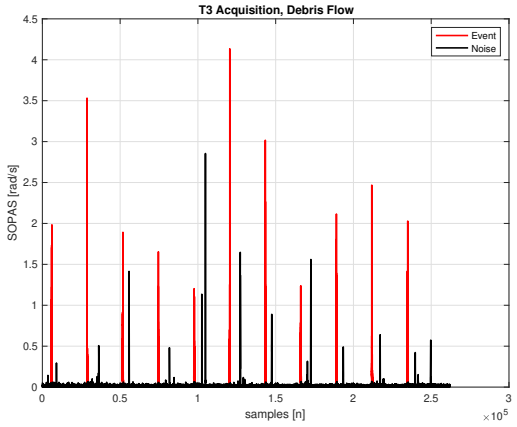

Figure 3.35: Debris Flow serie

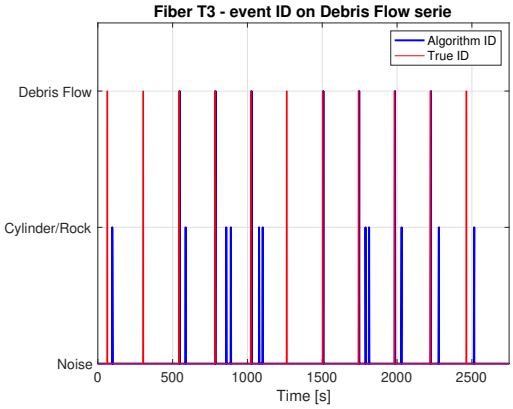

Figure 3.36: Debris Flow identification

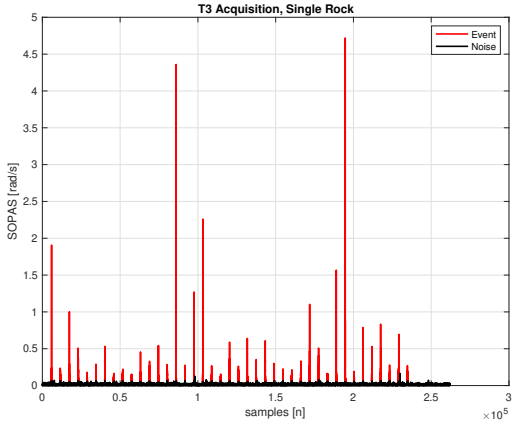

Figure 3.37: Single Rock serie

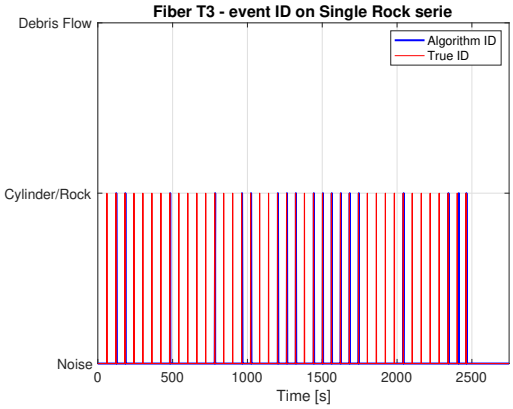

Figure 3.38: Single Rock identification

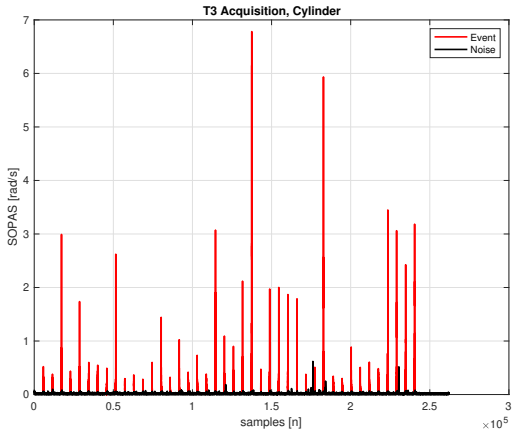

Figure 3.39: Cylinder serie

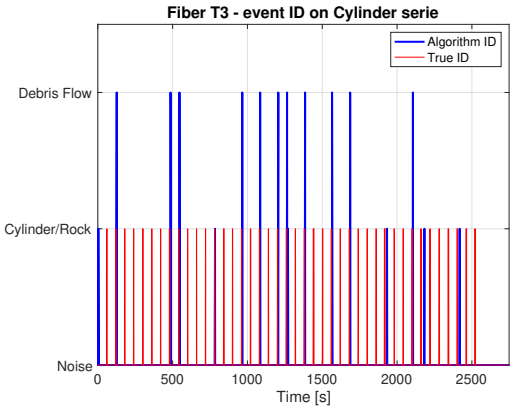

Figure 3.40: Cylinder identification

On the right instead it is showed the event classification done by the algorithm versus the correct labeling. The spurious components are due to, as said, someone accidentally touching the fibers during the experiments. In the end this algorithm is absolutely far from working. Almost each time an event happens it is almost always either confused with noise, either not recognized correctly. The spurious components are very annoying for this kind of algorithm, since in the majority of cases are confused with mainly Single Rock or Cylinder. What still works perfectly is the alarm triggering: it is just not able in any way to distinguish between actual dangerous events and spurious noise, that still remains a huge weakness of this approach.

Chapter 4

Neural Networks approach

In the last Section of the previous Chapter, we tried events identification exploiting their duration, with quite bad outcomes. The goal was to try an approach that would be able to tell apart the different events, in order to make the system more robust to False Alarms. The analysis carried in this Chapter has quite the same purpose, but considers a completely different approach, based on the usage of Neural Networks (NN). This is a new, very flexible, cutting edge technology, that proved to be suitable for pattern recognition in many fields, which is similar to our situation [6]. This could be the perfect candidate to allow an accurate event recognition and, in this very specific case, it should be able to tell if something happening on the fiber is belonging to:

- **Class 0:** Noise or spurious noise, non dangerous events.
- **Class 1:** Single Rock, dangerous events.
- **Class 2:** Debris Flow, dangerous events.
- **Class 3:** Cylinder, dangerous events.

Events belonging to Classes 1, 2 and 3 are classified as dangerous: this means that if their SOPAS time evolution exceeds the threshold, then the alarm must trigger. Class 0 deserves instead a further clarification, since here lies the core of this whole new analysis. Generally, if the NN identifies something happening on the fiber as belonging to this Class, whatever it is, it is classified as non dangerous. As reported in the bullets above, this could be due to just fiber Noise, which is the one generated naturally by the fiber when nothing happens over it (to be perfectly clear, this is the one reported in Figures 2.28, A.4, A.8, and used to generate all the detection maps reported in this work), or to spurious noise. When talking about this last one, which was already introduced previously, we refer to something happening over the fiber, not generated naturally by it. Despite this, spurious noise is not due to some dangerous event but, in our specific experimental scenario, to someone accidentally touching the structure and creating unwanted disturbances, that result as peaks on the SOPAS time evolution that could,

but should not, trigger the alarm. In a real world scenario these ones can be associated, for example, to animals or people passing over the fiber, or non dangerous natural events happening over it, so it is very important to consider them. Since these are non dangerous events that, if only using the threshold algorithm, could trigger the alarm, the NN should be able to tell these apart from real, dangerous, events belonging to all the other Classes, in order to avoid making the alarm trigger uselessly. It is also interesting to see how the NN reacts on identifying events as belonging to Classes 1, 2 and 3 which are all labeled as dangerous, but are actually completely different in terms of how they are made, how they evolve over the model and, of course, intensity. This approach could give huge advantages on the good functioning and robustness of the monitoring system, and also opening to a lot of interesting follow-ups.

4.1 Introduction to the main concepts of NN.

This Thesis has not its main focus on the NN subject, which is treated as a possible follow-up of this work. For this reason the main discussion will be about the preliminary results obtained with this approach, not much about all the details behind it. Nevertheless, an introduction to the main concepts and parameters on which to act is needed to understand the following analysis. The classification should work as in Figure 4.1. In

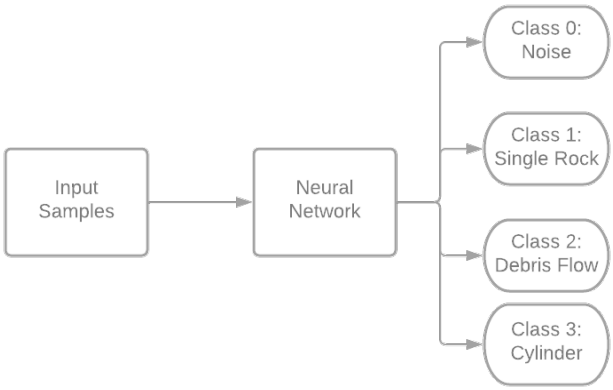

Figure 4.1: Classification problem using NN

order to work properly, a Neural Network needs to be trained. This means that, prior to use it for the events recognition, some time evolutions examples representing the events, labeled with the correct one that should be classified by the NN, are first given as input. The network will learn with a certain accuracy how to recognize properly if some samples belong to a given Class (0, 1, 2 or 3), and will be ready to take as input other ones, never seen before, and classify them. These two stages are called *training* and *testing*. It is known that, the higher is the number of evolutions given in input in the

training stage, the higher the accuracy in the testing one (in general). In the following Section some approaches based on different kind of training data and algorithms are proposed, so to evaluate the best solution. The kind of Network that has been used is called Feed Forward (FFNN), which belongs to the simplest kind of NN. For this specific case this is made like Figure 4.2, where the "circles" represent the most elementary part of the NN, called *perceptron*. Some keywords useful to understand what follows are

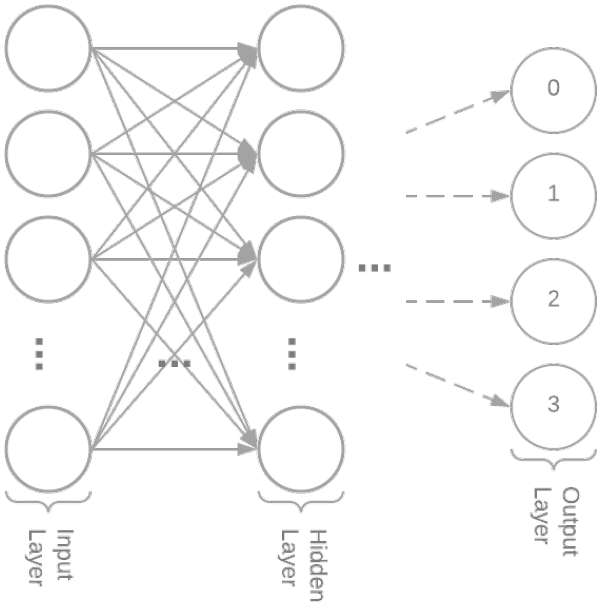

Figure 4.2: NN structure

reported below [10]:

- **Memory:** this is the number of perceptrons in the input layer. In this specific case there is only one hidden layer, which will also have this number of perceptrons.
- **Backpropagation algorithm:** this is the kind of algorithm used to train the NN. It is based on considering samples of data at the input layer on the perceptron side, which will output a value to the following layer, and so on up to the output one. The output value is based on some intrinsic perceptron's parameters called *weights* and *biases*. Once the final layer has been reached, all the two parameters of the perceptrons are updated, based on an error value that each time is computed, which needs to be decreased more and more each iteration.
- **Epoch:** each time the weights and biases are updated, it means that one epoch has passed. The bigger the number of epochs considered, the more accurate the trainig should be. Of course considering a lot of epochs means that the training takes a lot of time to end.

- **Mini-batch:** to be trained, the NN needs to consider examples representing the events to be classified. Instead of giving them all as input to the training, they are given scattered in different smaller batches, of a reduced number of examples. Each batch is changed after each epoch, in this way training should be improved.

In our specific case we have then one input layer, one hidden layer and one output layer, having respectively a number of perceptrons equal to the memory size for the first two, and just four for the output one. Each perceptron of each layer is connected to each one of the following. The number of perceptrons in the output layer is just four since this is the number of Classes over which the NN should identify events (see Fig. 4.1). The network will in fact take as input a number of samples equal to the memory size (one sample per perceptron), and output four numbers, which are summing up to one. These are representing how much the network is sure that a given window of samples belongs to one of the pre-defined Classes: usually, if the network is very confident that a set of samples belongs to a Class, the related perceptron will output a number that is very near to one, while the others will be very low, near to zero. The worst case is when all the perceptrons output similar numbers, since the uncertainty on the classification is high.

4.2 Neural Network applied on the SOPAS evolutions

The basic idea exploited to classify the events is to give as input to the NN the evolution over time of the angular speeds of the single events, through which they should be classified. Let's take a look at how the dataset is made, before proceeding. For each single event evolution, all computed with the same fixed T_{mov}, all the three detection conditions have been generated, as reported in Table 4.1, by choosing different, increasing, thresholds. In particular only one False Alarm and Missed Detection situations per event have been considered, and two Correct Detections, so four different evolutions per single event. In this way the NN would be able to see each event from different "perspectives", and maybe learn various features about them, which could be of help to enhance the event classification. Note that only T fibers have been considered, so what shown in Table is true for T1, T2, T3. For each fiber Ti, four different evolutions per

Threshold [rad/s]	*Detection*
0.02	False Alarm (FA)
0.04	Correct Detection (CD)
0.06	Correct Detection (CD)
0.08	Missed Detection (MD)

Table 4.1: Dataset conditions

event are considered. For each fiber, 33 events were measured (see Table 2.2), and thus the total number of SOPAS time evolutions is 396. Of course the time identification of

the event is not the only thing that is needed, but also the event labeling is fundamental. This is done once the alarm vector on the SOPAS time evolution has been identified, by labeling it "by hand": all the samples of the SOPAS evolution that triggered the alarm are labeled as belonging to one of the different Classes that the NN should identify, so that the training could happen smoothly.
Another concept needs to be introduced, called *k-folding*. Since the dataset is composed by three portions, one for each fiber, it could be smart to test the NN with only one of them, and train it with the other two. The k-folding concept works exactly in this way: k is set to three in this case, and first two portions of the whole dataset are used for the training, while the other one for testing, then the combination is inverted, until all the possible ones are tested. In this way $k = 3$ different results are obtained, with different performances. These ones are reported in Table 4.3, while Table 4.2 reports the parameters set for this test. Other ones have been tried, but these ones seem to work best for this case. In Table 4.3, the notation *I PA* refers to the first (*I*, of course

Epochs	*Mini-batch size* [samples]	*Memory size* [samples]
3	100	205

Table 4.2: NN parameters set for the test

same concept for *II* and *III*, which instead will refer to the second and third) k-folding combination of the dataset, while *PA* means Prediction Accuracy. Table 4.3 is saying

Class	*I PA*	*II PA*	*III PA*
0 (Noise)	99.54%	99.59%	99.60%
1 (SR)	80.75%	83.51%	79.71%
2 (DF)	51.49%	53.94%	50.07%
3 (C)	98.75%	98.66%	98.66%

Table 4.3: NN Prediction Accuracy[1]

that the NN approach has generally a quite high accuracy in recognising the event, for sure much better than the duration detection approach. The best accuracy is on the Cylinder and Noise, which are basically always identified. Single Rock works also quite good, some problems are instead there when talking about Debris Flow, which on average is identified one time every two. This is not a very good result, since the Debris Flow is actually the most dangerous event, and should be the one that is almost always correctly classified, not confused with other ones. The (interpolated) distribution

[1]Accuracy that the NN has on saying that the input samples are classified as belonging to Class 0, 1, 2 or 3, with respect to the real labeling. The higher this value, the more reliable the identification of the event done by the Network is.

plots for the three events has also been reported, which are giving a more clear idea of what is happening, in Figures 4.3, 4.4 and 4.5. These plots represent how much the

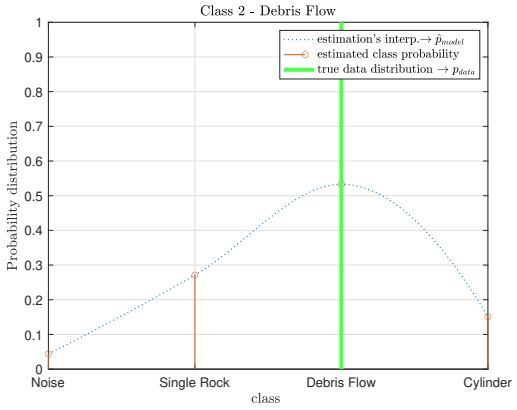

Figure 4.3: DF distribution

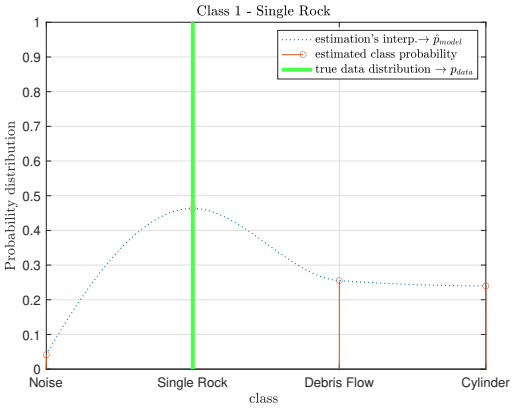

Figure 4.4: SR distribution

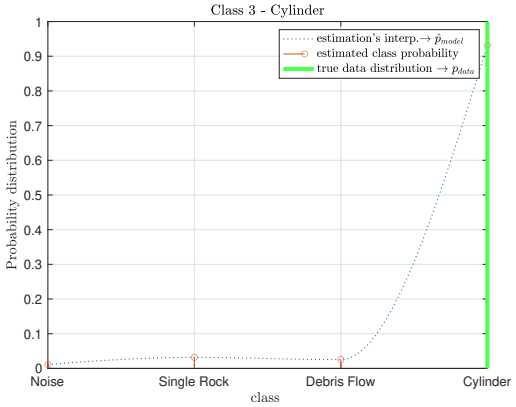

Figure 4.5: C distribution

NN is sure not to confuse one event with the other. This means that if the curve is flat over the four events, the Network has very similar probabilities to output one of them, so the classification works very bad. The higher is the peak on the specific event, the better the classification of that single one works. From the Figures is very clear how Cylinder is very easily recognized with a lot of confidence, while DF and SR have low peaks and quite flat curves: there is a strong probability for them to be confused with the other ones in the classification process, so the NN does not work so good on them. This situation is the best one that can possibly be obtained in this way, even changing the parameters in Table 4.2.

After this preliminary assesment of NN, a change of approach has been taken into consideration. The first change that has been adopted was about the way that the samples of the SOPAS were considered: for the testing of the NN, 205 samples (the memory size)

were given as input to it. In the preliminary algorithm, each sample was classified as belonging to one of the four different classes, the memory window shifted of one sample and the whole process restarted again. This would be quite useless, since one sample would be classified a lot of times, so it was decided to adopt a more "window-based" approach instead of a "sample-based" one. Note that this is all related to the testing stage, the training is left completely unchanged, but with no k-folding. Also, the dataset considered uses only the SOPAS evolution obtained with thresholds equal to 0.04 rad/s (see Table 4.1). The working scheme of this new approach is reported in Figure 4.6.

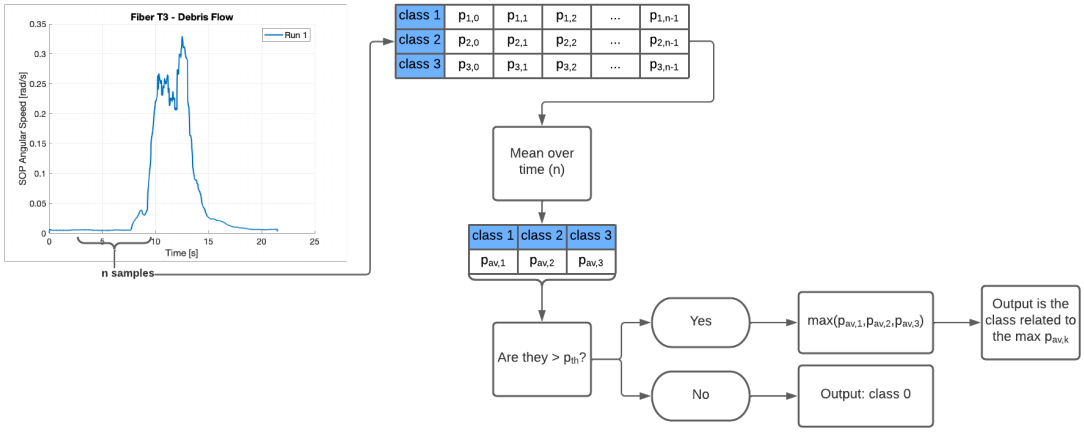

Figure 4.6: NN new approach working principle

The parameters in Figure represents:

- n: samples of window length. This parameter is a degree of freedom of the scheme, and by changing it, different performances could be achieved.
- $p_{c,k}$: represents the confidence of the Network to classify one sample as belonging to one class. The class is represented with parameter c, while discrete time with k.
- $p_{av,c}$: this is the average over time (so over the n values inside a window) per each class. As shown in Figure in the end there will be only three values, since only three classes are considered.
- p_{th}: this is another degree of freedom of the system. It represents a threshold, if any of the $p_{av,c}$ values is bigger than it, it means that on average the NN is very confident that the window contains an event. If they are smaller, it means that the NN is on average confident that the window contains noise.

Before showing the results obtained with this approach, the two degrees of freedom needs to be fixed. After some tests it has been found that the two best values granting very good performances are the ones in Table 4.4, among the others that need to be

n [samples]	p_{th}	*Epochs*	*Mini-batch size* [samples]	*Memory size* [samples]
600	0.05	5	200	205

Table 4.4: NN parameters set for this new approach

fixed each time. This NN classification algorithm has been applied on a serie of SOPAS built by combining each single event evolution used for testing, one after the other. The result in Figure 4.7 shows the performances of this algorithm versus the true labeling of the events. From this Figure, it's possible to say that these results are actually very

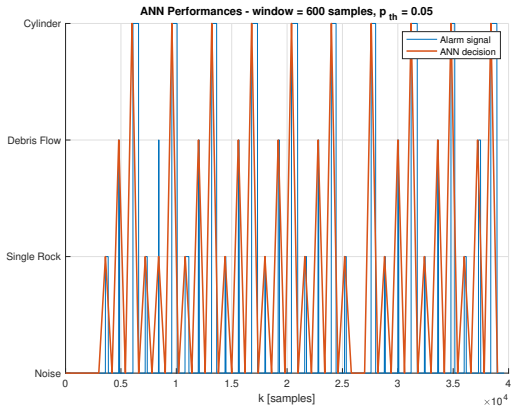

Figure 4.7: NN performances

good: every event (apart from one) is correctly classified. In this way the performances of the alarm algorithm, which is already working very well in terms of alarm triggering, would be improved a lot, since now the system could be able to differentiate among events, and consequently in some way "decide" if it is worth giving the alarm or not. This is already an extremely good result, but actually the dataset used to train and test the NN is free from spurious noise, so it's only possible to guess that this could be excluded from generating FAs, but no proof has been given up to now. In order to show the functioning of the event classification even with spurious noise, another dataset has been used, to which the evolutions in Figures 3.35, 3.37 and 3.39 belong. These

Purpose	*Debris Flow*	*Single Rock*	*Cylinder*
Training	22	22	22
Testing	8	19	20
Total	30	41	42

Table 4.5: New dataset used, T3

evolutions are basically a long serie of events generated over time, exclusively on T3, in which accidentally spurious noise has been generated too. If considering just the

single events, the total number and the ones used for training and testing are reported in Table 4.5. The algorithm used is of course the same of Figure 4.6. The result obtained

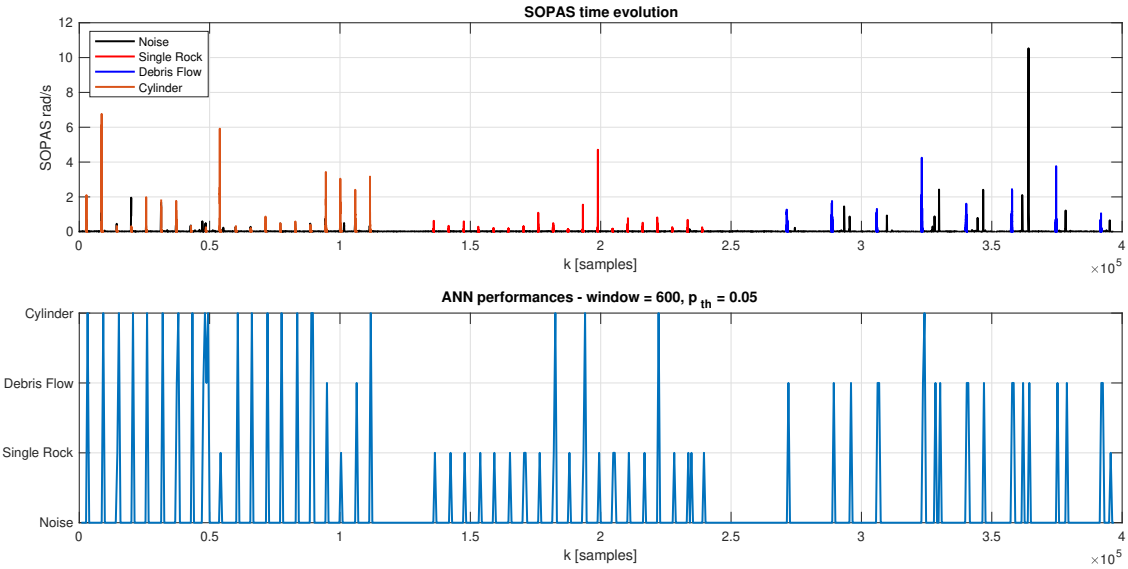

Figure 4.8: NN performances, new training

is in Figure 4.8, of which Figures 4.9, 4.10 and 4.11 report a zoom in on the single event class, since the total SOPAS time evolution has been obtained by concatenating all the single long ones. Before proceeding with commenting these very important plots, some details needs to be explained. First of all, the black lines represent everything that is not generated by dangerous events (so the ones belonging to Classes 1, 2 or 3), which the NN should classify as belonging to Class 0. All the other colours (red, blue, orange) represent the events that needs to be recognized by the Network as dangerous. Another

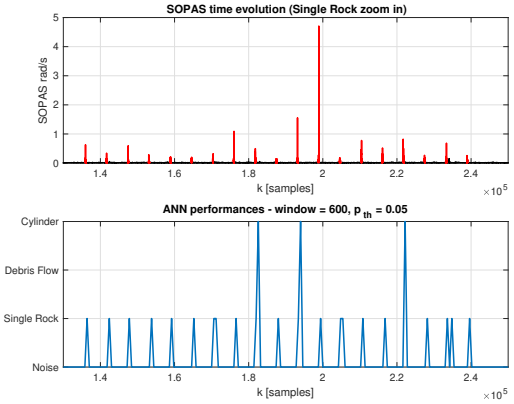

Figure 4.9: Zoom in, SR

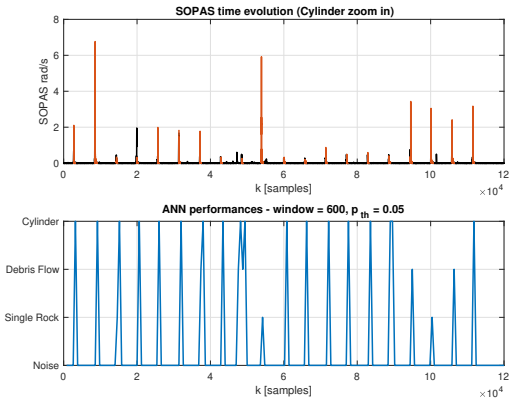

Figure 4.10: Zoom in, C

fact that needs to be discussed is the slight misalignement of the NN classification with

respect to the happening of the event, which seems to be delayed a bit: this is due to the fact that the NN is able to classify events by reading windows by windows of n samples. This means that the actual classification will be delayed of n samples, which is a constraint that cannot be avoided.
In general, looking at Fig. 4.8, all the events seems to be correctly classified in almost every case, but for some exception, in particular if there is not spurious noise, the classification seems to go smoothly. Going instead into more details in Figures 4.9 and

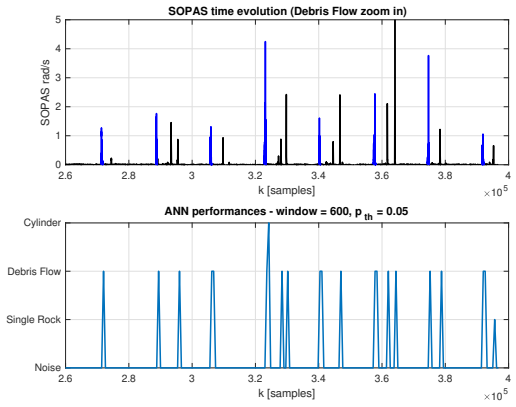

Figure 4.11: Zoom in, DF

4.10, Single Rock and Cylinders are almost always correctly classified, and spurious noise creates problems very rarely, like one or two times over the about twenty classifications done by the NN. Note that here the amount of peaks not due to an event is very low, and if they appear they do not seem to be that pronounced. Very different is instead the case of Debris Flow (Figure 4.11), in which spurious noise is very present. Here despite the classification which is done quite correctly, the spurious peaks are a lot of times confused with another Debris, for the majority of times. This could represent a problem, since would lead to an increased number of False Alarms triggered by the system. This could be maybe mitigated by using a larger number of events for training. Another test using a lower number of samples inside the window and a lower p_{th} is reported in Figure 4.12. Results are clearly worsened, so it is not worth trying to go even lower. It is instead worth to go upper: the window size is really affected by the length, in samples, of the DF which usually is around 600 and, exactly as shown, if choosing lower values performances drop significantly. With 700 and 800 samples respectively, Figures 4.13 and 4.14 are obtained, with much better performances, increasing with the window size. Enhanced events recognition is obtained, but still spurious noise is often identified as an event, with very small improvement. Note that still, despite the spurious noise weakness that is not completely solved, NN mitigates the problem a lot, and gives extremely good results that are worth of a follow-up of this Thesis.

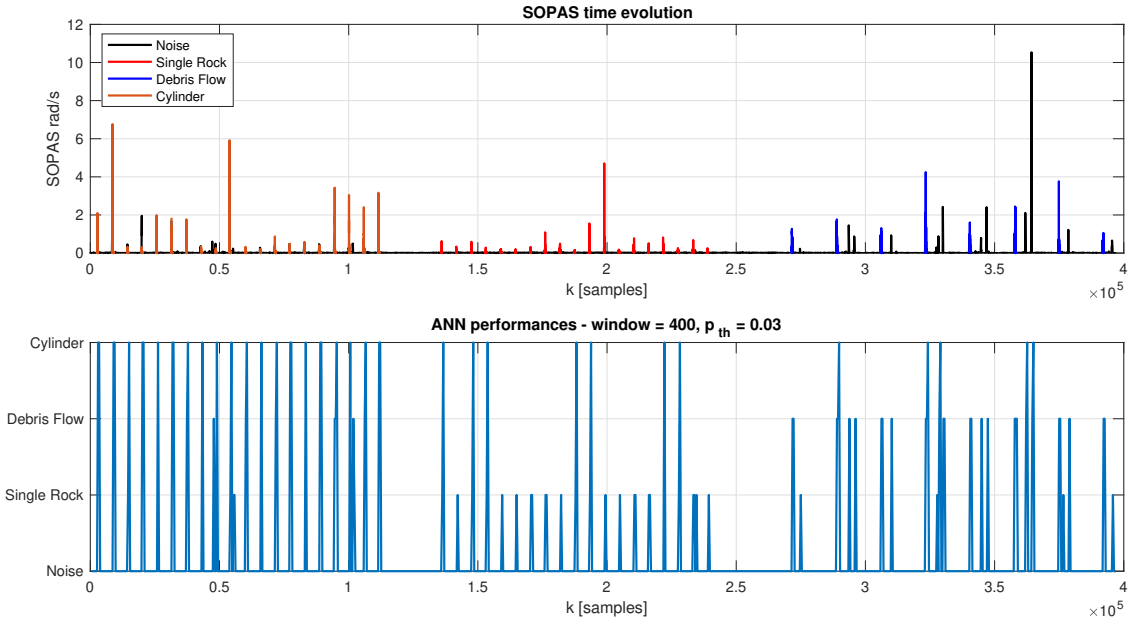

Figure 4.12: NN performances, new training, different parameters

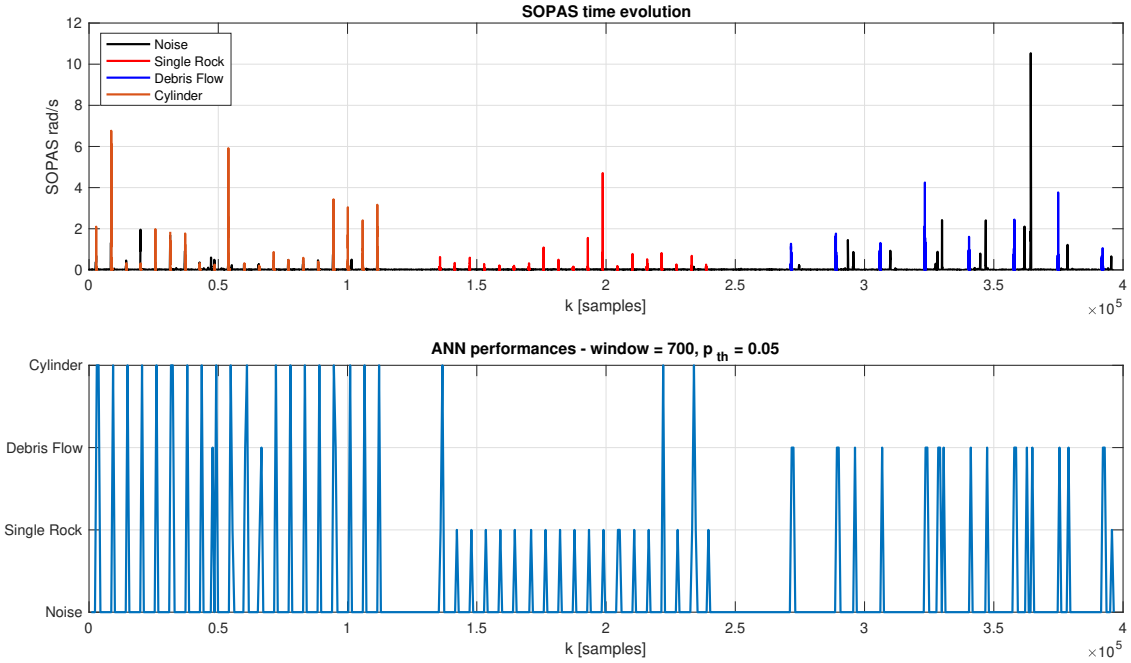

Figure 4.13: NN performances, new training, different parameters

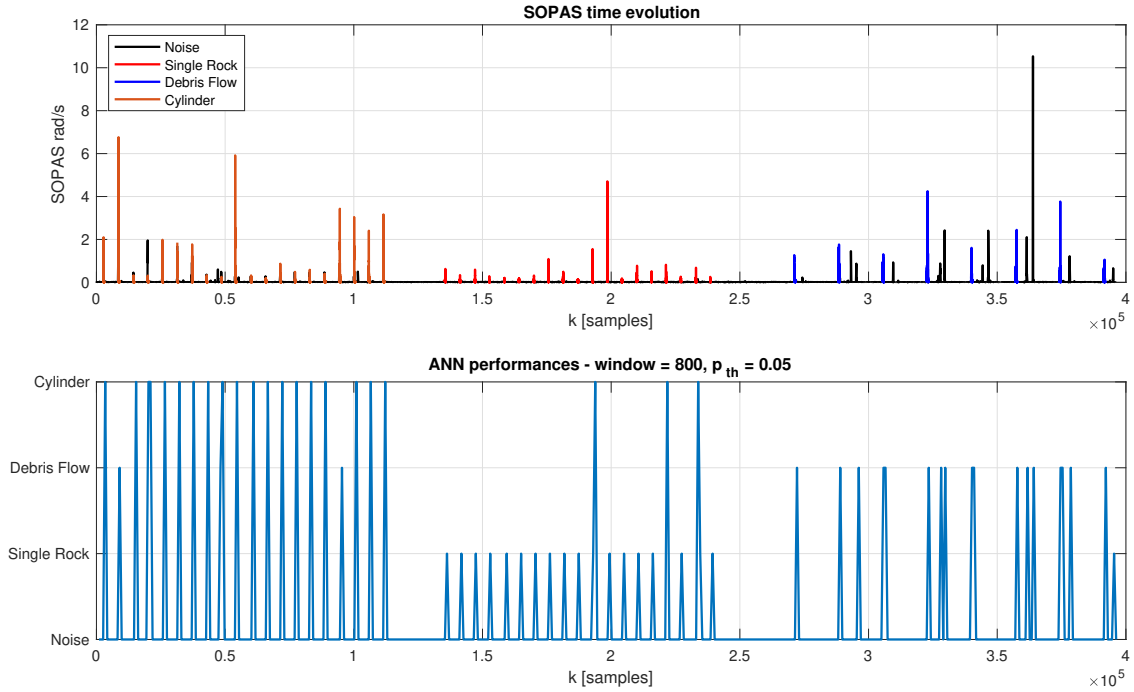

Figure 4.14: NN performances, new training, different parameters

Chapter 5

Conclusions

This Thesis work revealed that a Optical Sensing system based on polarization priciple to detect dangerous events happening on a mountain scenario, could actually work. The fact that it is just working is confirmed by the detection map, in Figure 5.1, which is the very first important result that has been obtained, with an extremely preliminary algorithm: the yellow area is not so big with respect to the blue one, to which unsafe pairs of parameters belong, but still there are a lot of pairs that can be selected to grant a correct detection. This was already a good result which, through the f_s optimization, was made a lot better, as shown in Figure 5.2. This is the map obtained applying the final algorithm on the resampled versions of the single events, and represents the core result of this whole work. Not only the sensing based on polarization is actually working by means of a quite simple threshold algorithm, but the performances are extremely good, since a lot of pairs can be selected to get perfect alarm triggering. The consequences,

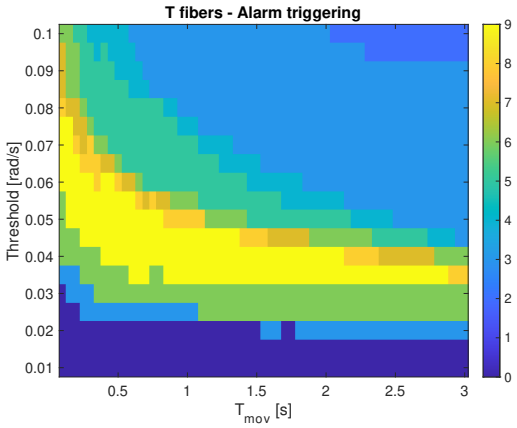

Figure 5.1: T fibers map, preliminary algorithm

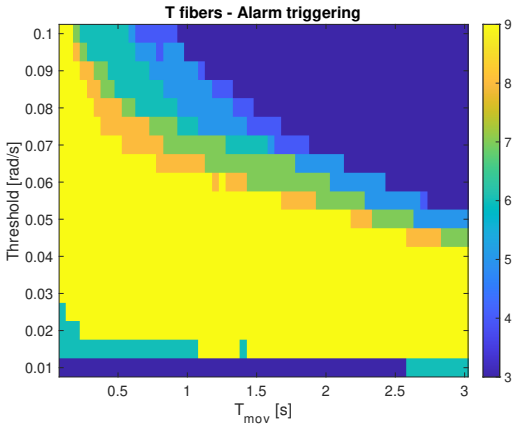

Figure 5.2: T fibers map, real time algorithm

other than the performances which are of course enhanced, reflect also on the complexity of the whole processing chain which can go very slow, at 95.4 Hz, and give perfect

results. In this way the whole system is also much cheaper, and could represent a good alternative to some of the other Optical Sensing techniques that have been shown and could work better, but are much more costly.
On the laboratory miniature model, the correct functioning of the Real-time mode algorithm and the coherency of all the detection maps and analysis done to obtain it are confirmed by the real time SOPAS plots, of which one is again reported in Figure 5.3. This shows how choosing correctly the values of $\Delta\omega_{th}$ and T_{mov}, which could be

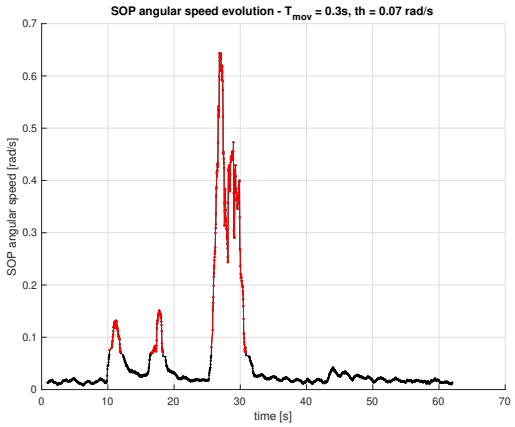

Figure 5.3: Correct applicayion of the real time algorithm

easily retrieved from the detection maps, the alarm is triggered correctly, without the generation of False Alarms (due to fiber noise), easily avoided.

The other core result is about the much more difficoult task of events identification by means of NN. It has been shown how the use of this kind of technology allows the recognition of the events, and possibly the reduction of the number of False Alarms, generated by spurious components of noise, not by events. This approach is not working perfectly, but still it's not bad either, and in particular demonstrates that the application of NN could really improve the alarm system which, left alone, would be too much weak against FAs generated by other influences than just fiber noise. This part constitutes the main follow-up of this Thesis, where for example some other kinds of NN could be adopted, and maybe the classification could consider the Stokes parameters in place of the SOPAS. Overall, the final system that should now work, given all the results obtained in the previous Chapters, could be made like the one in Figure 5.4, where the integration of NN is adopted. In this way FAs due to spurious noise should be mitigated a lot if the NN is able to perfectly identify events, since before giving an alarm, the event would be classified.

With these good results, some limits needs to be also reported, also as starting points of a successive possible work. For example, the delay generated by the cascade of FIR filter and NN could be not negligible since, as reported in the previous Chapter, the Network is

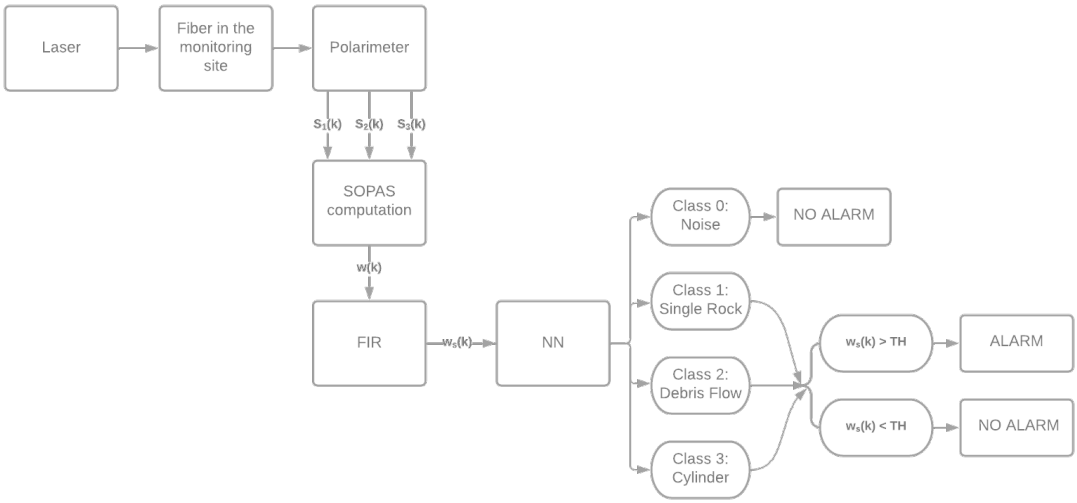

Figure 5.4: Possible complete scheme working principle

able to identify, with good results, an event only about 600 samples after its occurrence, which is about 6 seconds in our scale model. The FIR filter will also introduce a delay, and the overall effect could be a problem for an application on a real scenario. Another problem is represented by the scalability of the whole system: everything that has been tested and analyzed in the laboratory model, would in fact need to be translated in the real world. This could not be so straightforward, and bring issues releated to, for example, the installation of the system (fibers, in particular) on a quite rough scenario, and interfacing and testing it with actual real world events, which would have longer durations (even up to one minute for a Debris, for example), much bigger intensity, and most importantly would be truly dangerous.

Overall, the analysis done in this Thesis proves the good functioning of the system with good perormances and low costs. Despite the implementation issues that could come up and the possible enhancement of the NN, these results are already very exciting and promising, and could constitute intermediate steps to realize a new, advanced and reliable kind of mountain monitoring system.

Appendix A

SOPAS behaviors, Offline detection maps

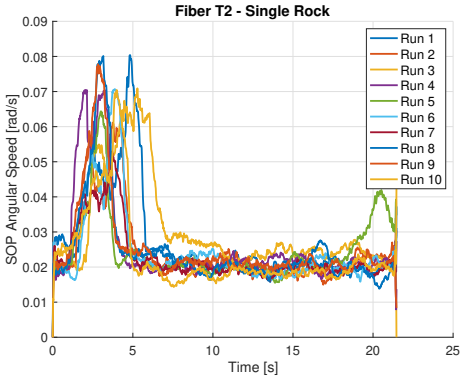

Figure A.1: T2, SR runs

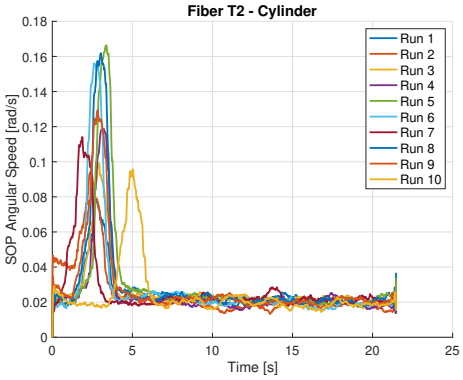

Figure A.2: T2, C runs

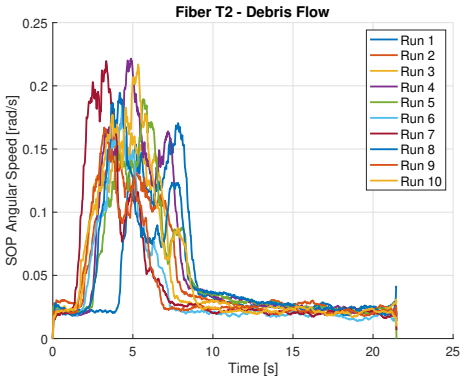

Figure A.3: T2, DF runs

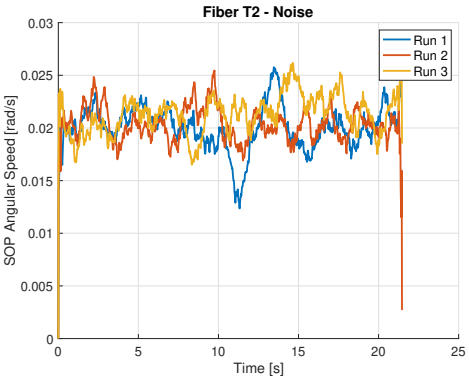

Figure A.4: T2, Noise runs

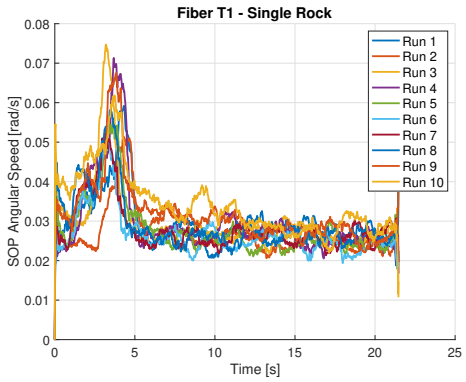

Figure A.5: T1, SR runs

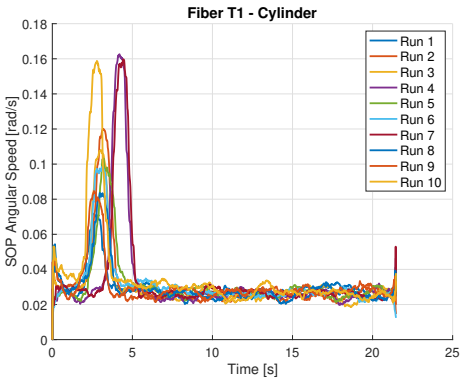

Figure A.6: T1, C runs

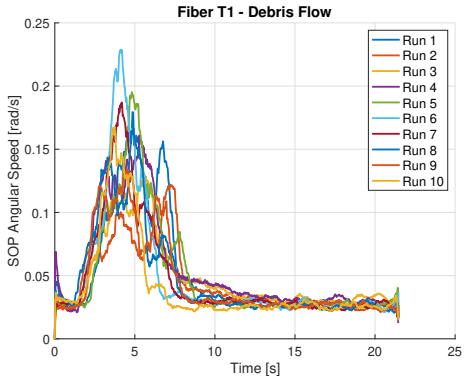

Figure A.7: T3, DF runs

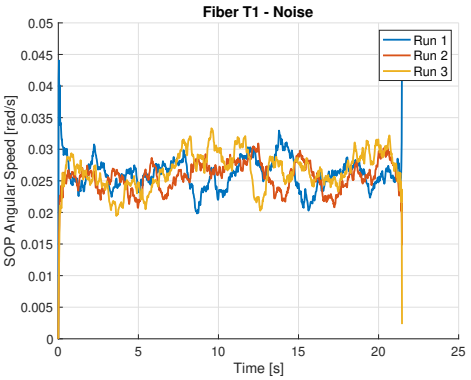

Figure A.8: T3, Noise runs

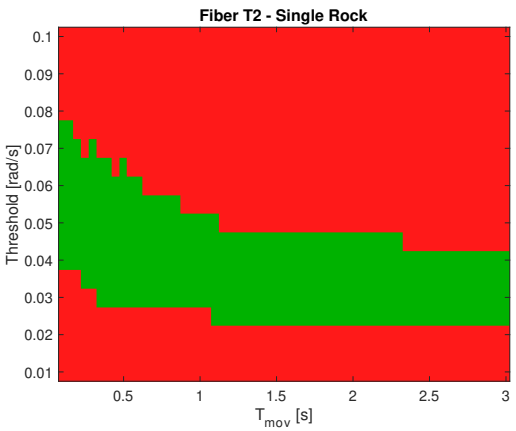

Figure A.9: non optimized map

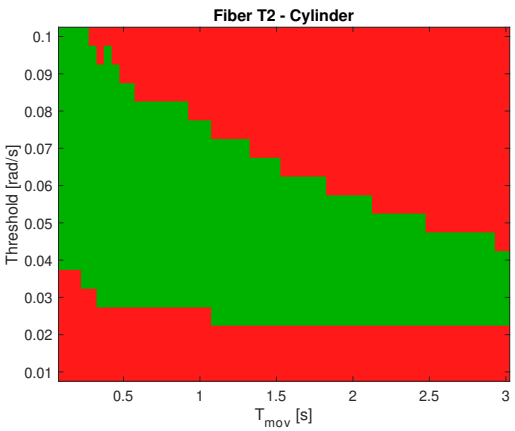

Figure A.10: non optimized map

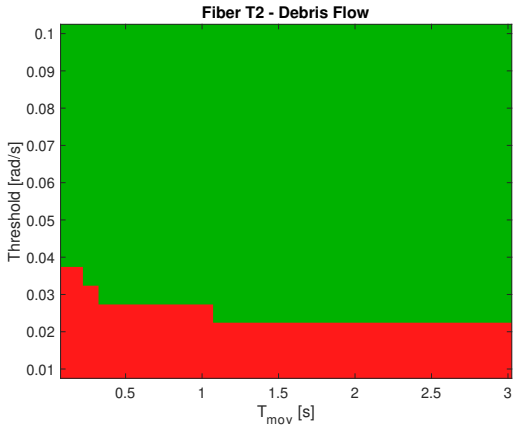

Figure A.11: non optimized map

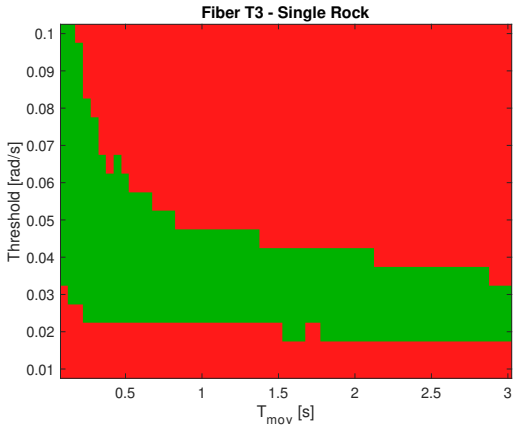

Figure A.12: non optimized map

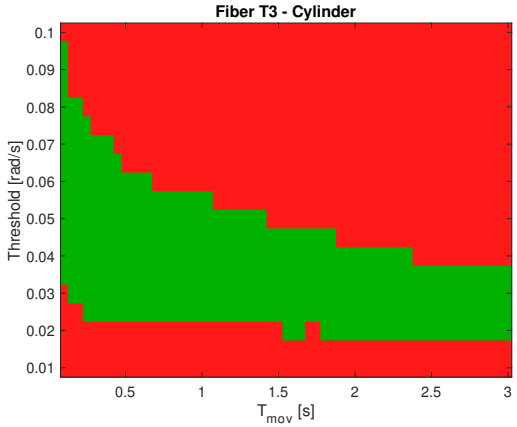

Figure A.13: non optimized map

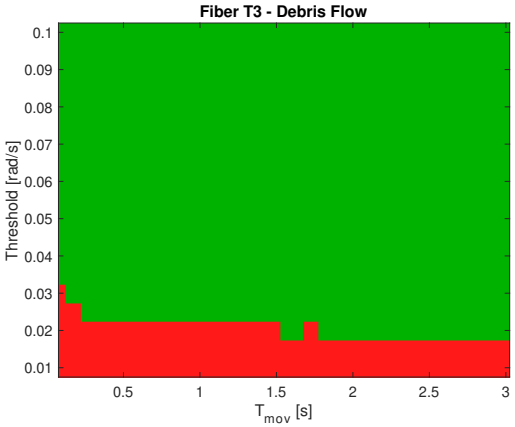

Figure A.14: non optimized map

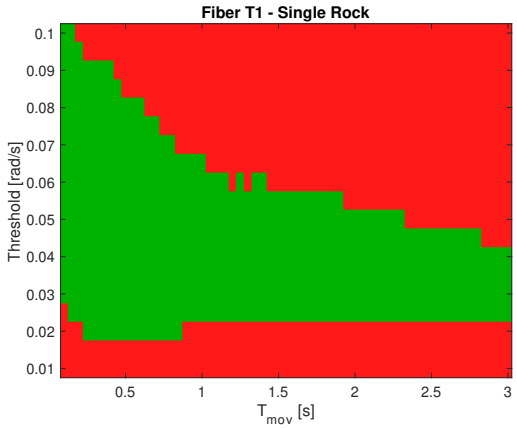

Figure A.15: optimized map

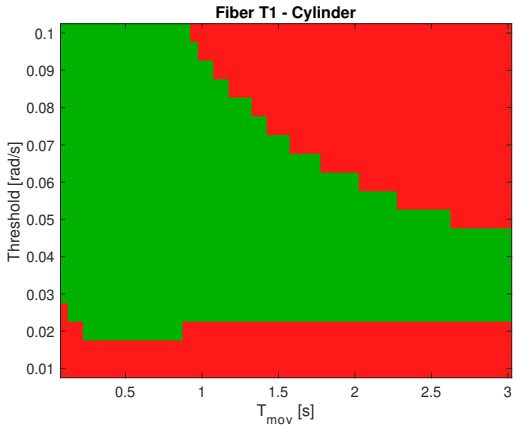

Figure A.16: optimized map

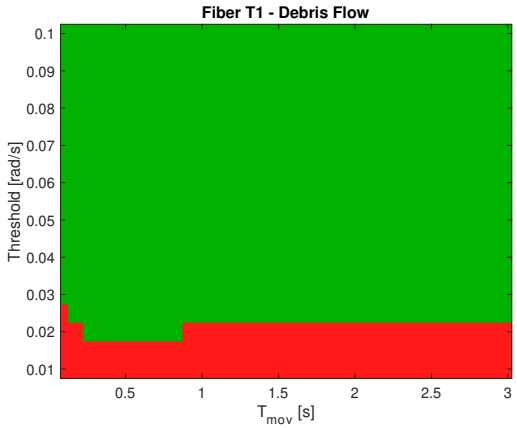

Figure A.17: optimized map

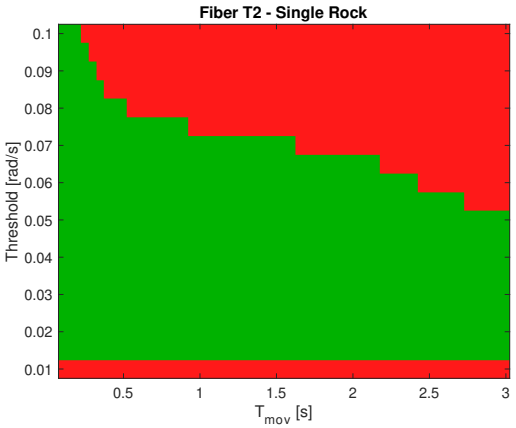

Figure A.18: optimized map

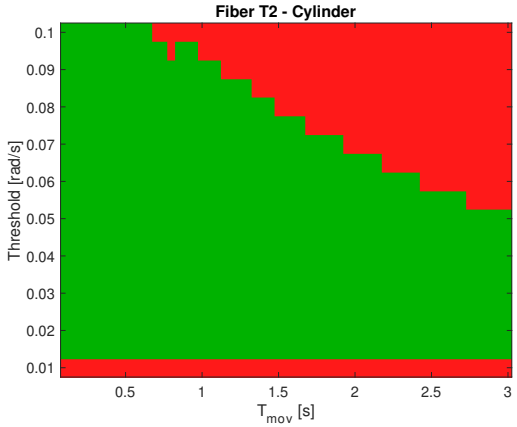

Figure A.19: optimized map

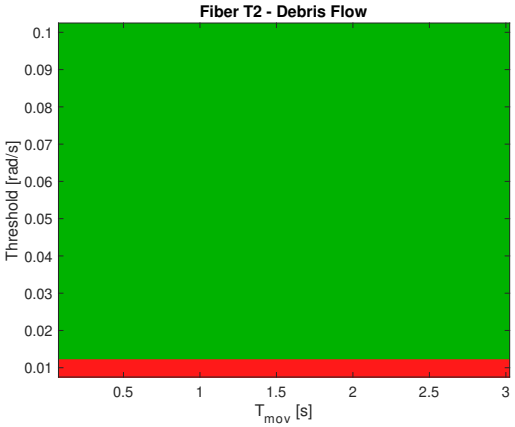

Figure A.20: optimized map

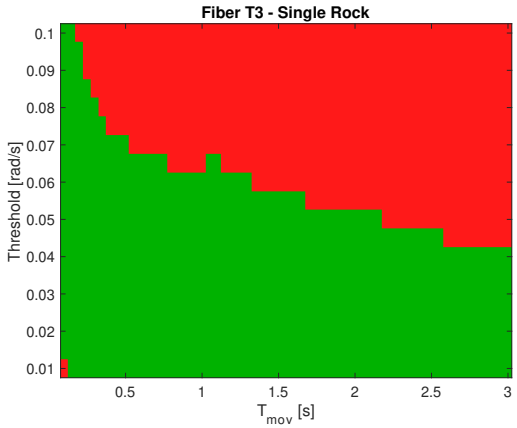

Figure A.21: optimized map

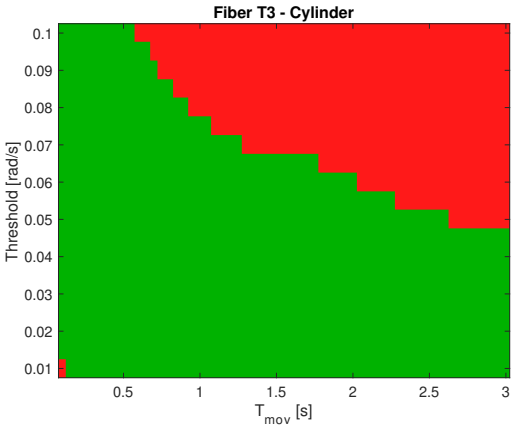

Figure A.22: optimized map

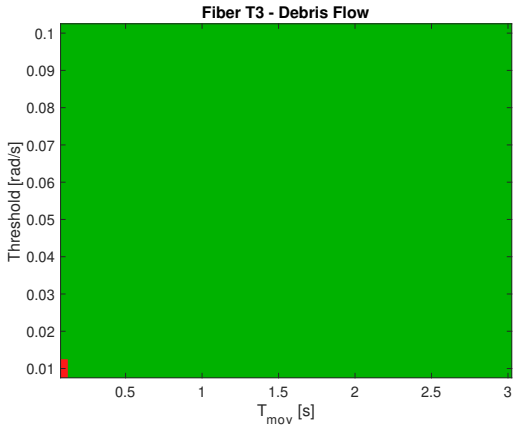

Figure A.23: optimized map

Appendix B

SOPAS long acquisitions plots

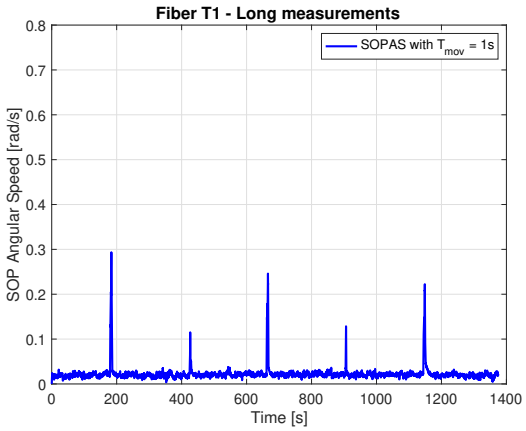

Figure B.1: T1, long acquisition

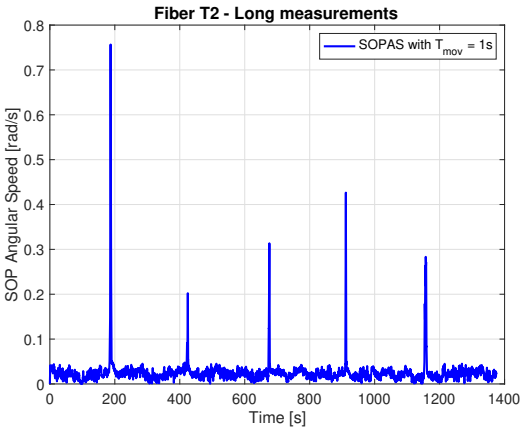

Figure B.2: T2, long acquisition

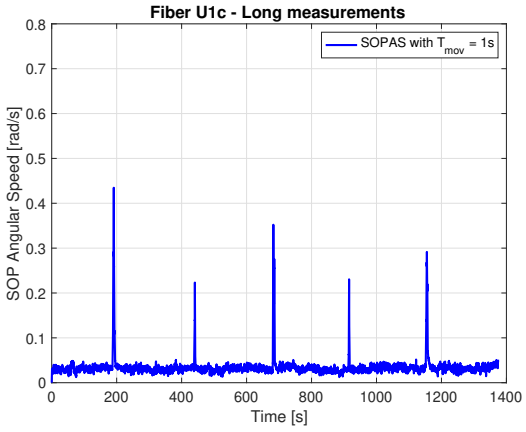

Figure B.3: U1c, long acquisition

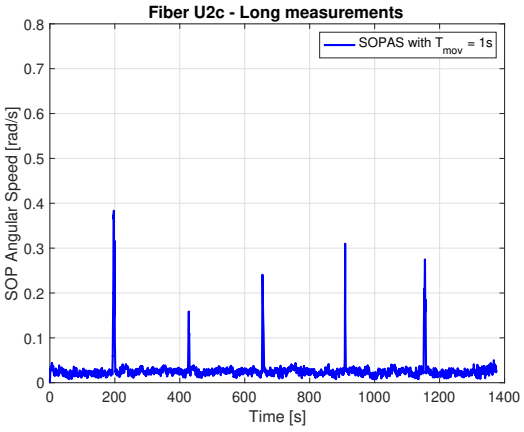

Figure B.4: U2c, long acquisition

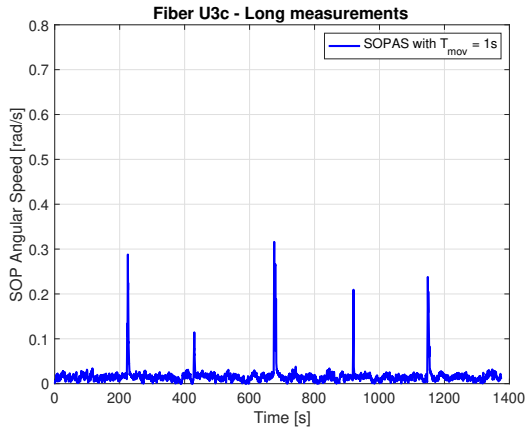

Figure B.5: U3c, long acquisition

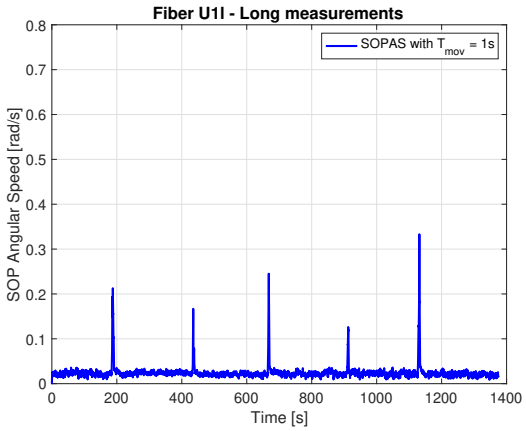

Figure B.6: U1l, long acquisition

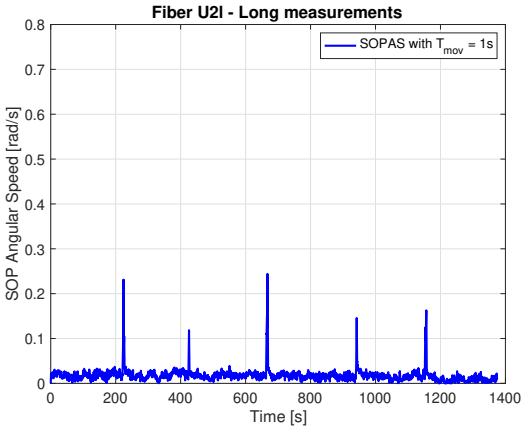

Figure B.7: U2l, long acquisition

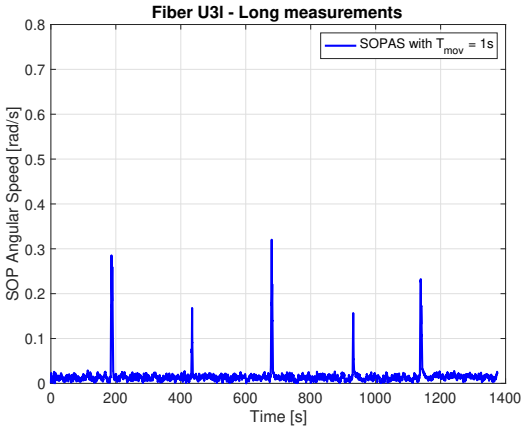

Figure B.8: U3l, long acquisition

Appendix C

Real time algorithm: Matlab script

```
clear all;
close all;

initdevice;

%set up
offset = 512;
%addresses where to write the ATE, ME to be set correctly
ATE_add = 1;
ME_add = 73;
%addresses where to read the Stokes parameters
S1_add = 25;
S2_add = 26;
S3_add = 27;
%values of ATE, ME to set
ATE = 20;
ME = 11;
sim_length = 100; %how many seconds to acquire
Tmov = 1; %moving average window size in seconds
th = 0.09; %threshold value on the SOP angular speed in rad/s
amp = 10; %intensity of the beeps
fs_beep = 1000; %sampling frequency of the beeps
%duration of the beeps
duration_a = 400;
duration_b = 4;
%frequency of the beeps
freq1 = 100;
freq2 = 80;
values_a = 1:1/fs_beep:duration_a;
values_b = 1:1/fs_beep:duration_b;
```

```
%sinusoids for the sound() function
a = amp*sin(2*pi*freq1*values_a);
b = amp*sin(2*pi*freq2*values_b);

writedevice(offset + ATE_add, ATE); %write the desidered ATE in ...
    the register
writedevice(offset + ME_add, ME); %write the desired ME in the ...
    register

fs = round(100e6/2^ATE, 1); %sampling frequency of the polarimeter
p = 1/fs; %seconds to pause the for loop
sim_samples = ceil(sim_length*fs);
Tmov_samples = ceil(Tmov*fs);
buffer = NaN(1, Tmov_samples);
started = 0;
t = 0; %time axis initialization for the plot
sounded = 0; %sound alarm flag

%semaphore setup
green = [0 0.7 0];
red = [1, 0, 0];
x = (1:10);
y = (1:10);
mat = ones(10, 10);

figure(1);
movegui('west')
imagesc(x,y,mat);
cmap = colormap(green);
title('Traffic Light');
axis off;

%real time SOPAS plotting setup
figure(2);
h1 = animatedline('Marker', '.');
h1.Color = 'black';
h2 = animatedline('Marker', '.');
h2.Color = 'red';
movegui('east')
title(['SOP angular speed evolution - T_{mov} = ', num2str(Tmov), ...
    's, th = ', num2str(th) ' rad/s']);

t0=tic;

for n = 1:sim_samples

    %Stokes parameters reading
    S1(n) = (readdevice(offset + S1_add) - 2^15)/(2^15);
    S2(n) = (readdevice(offset + S2_add) - 2^15)/(2^15);
    S3(n) = (readdevice(offset + S3_add) - 2^15)/(2^15);
```

```
78
79      if  n > 1
80
81          %SOPAS computation, sample per sample
82          angle(n) = acos((dot(S1(n), S1(n - 1))+ dot(S2(n), S2(n - ...
        1)) + dot(S3(n), S3(n - 1)))./dot(sqrt(S1(n)^2 + S2(n)^2 + S3(...
        n)^2), sqrt(S1(n - 1)^2 + S2(n - 1)^2 + S3(n - 1)^2)));
83          angle(n) = angle(n)/p;
84
85          %Algorithm
86          if (n == Tmov_samples)
87              smoothed_out = sum(buffer)/Tmov_samples;
88              if smoothed_out > th
89                  alarm(n) = 1;
90                  if (sounded == 0)
91                      sound(a)
92                      figure(1)
93                      cmap = colormap(red);
94                      sounded = 1;
95                  end
96              else
97                  alarm(n) = NaN;
98                  if(sounded == 1)
99                      clear sound
100                     sound(b);
101                     figure(1)
102                     cmap = colormap(green);
103                 end
104                 sounded = 0;
105             end
106             angle_smoothed(n) = smoothed_out;
107             started = 1;
108             time_offset = t;
109         elseif (started == 0)
110             buffer(n-1) = abs(angle(n));
111         elseif (started == 1)
112             buffer = circshift(buffer, -1);
113             buffer(end) = abs(angle(n));
114             smoothed_out = sum(buffer)/Tmov_samples;
115             if smoothed_out > th
116                 alarm(n) = 1;
117                 if (sounded == 0)
118                 sound(a);
119                 figure(1)
120                 cmap = colormap(red);
121                 sounded = 1;
122                 end
123             else
124                 alarm(n) = NaN;
125                 if(sounded == 1)
```

```
126                     clear sound;
127                     sound(b);
128                     cmap = colormap(green);
129                 end
130                 sounded = 0;
131             end
132             angle_smoothed(n) = smoothed_out;
133         end
134 
135         if (started == 1)
136             addpoints(h1, t - time_offset, angle_smoothed(n))
137             addpoints(h2, t - time_offset, alarm(n)*angle_smoothed...
    (n))
138             drawnow limitrate
139             xlabel('time [s]');
140             ylabel('SOP angular speed [rad/s]');
141             grid on;
142         end
143 
144     end
145 
146     pause(p)
147     t = toc(t0);
148 end
```

Bibliography

[1] The Weather Channel. Landslides, Flooding Clobber California, 2021. URL: https://weather.com/news/news/2021-10-24-california-live-updates-debris-flow-flooding-west-storms.

[2] Zhi-Yong Dai, Yong Liu, Li-Xun Zhang, Zhong-Hua Ou, Ce Zhou, and Yong-Zhi Liu. Landslide monitoring based on high-resolution distributed fiber optic stress sensor. In *2008 1st Asia-Pacific Optical Fiber Sensors Conference*, pages 1–4, 2008. doi:10.1109/APOS.2008.5226289.

[3] FOSCO. What is birefringence and beat length?, 2012. URL: https://www.fiberoptics4sale.com/blogs/archive-posts/95042886-what-is-birefringence-and-beat-length.

[4] Geoprotection. Barriere paramassi serie "AC", 2015. URL: http://www.geoprotection.com/it/barriere-paramassi-serie-ac-.html.

[5] Arthur H. Hartog. *An Introduction to Distributed Optical Fibre Sensors*. Series in Fiber Optic Sensors. CRC Press, 2017.

[6] D. King, W.B. Lyons, C. Flanagan, and E. Lewis. An optical-fiber sensor for use in water systems utilizing digital signal processing techniques and artificial neural network pattern recognition. *IEEE Sensors Journal*, 4(1):21–27, 2004. doi:10.1109/JSEN.2003.820344.

[7] H. Kogelnik and P. J. Winzer. Modal birefringence in weakly guiding fibers. *Journal of Lightwave Technology*, 30(14):2240–2245, 2012. doi:10.1109/JLT.2012.2193872.

[8] Antonio Mecozzi, Mattia Cantono, George C. Castellanos, Valey Kamalov, Rafael Muller, and Zhongwen Zhan. Polarization sensing using submarine optical cables. *Optica*, 8(6):788–795, June 2021. doi:10.1364/OPTICA.424307.

[9] Antonio Mecozzi, Mattia Cantono, Jorge C. Castellanos, Valey Kamalov, Rafael Muller, Zhongwen Zhan, and Shuang Yin. Seismic sensing in submarine fiber cables. In *2021 European Conference on Optical Communication (ECOC)*, pages 1–3, 2021. doi:10.1109/ECOC52684.2021.9605838.

[10] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[11] Novoptel. PM1000 Polarimeter, 2011. URL: https://www.novoptel.de/Polarimeter/Polarimeter_PM1000_en.php.

[12] Katsunari Okamoto. *Fundamentals of Optical Waveguides*. Elsevier Science, 2006.
[13] Simon Ramo, John R. Whinnery, and Theodore Van Duzer. *Fields and Waves in Communication Electronics*. Wiley, 2014.