Master's Thesis

Flare and Filament Detection in Hα Solar Image Sequences

Gernot Erich Riegler
Graz, Austria
April 2013

Thesis supervisor
Univ. Prof. DI Dr. Horst Bischof
Instructor
DI Dr. Thomas Pock

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)

Abstract

Solar storms can have a major impact on the infrastructure on earth and near earth objects. This includes satellites, astronauts, navigation systems, pipelines and power grids. Some of the related events are observable from ground based observatories in the Hα spectral line, namely solar flares and filament eruptions, which are correlated with coronal mass ejections. In this thesis we propose a new method for the simultaneous detection and segmentation of flares and filaments in Hα image sequences in near realtime. Therefore our method consists of four building blocks. In a first preprocessing step the Hα images get normalized in terms of intensity and spatial displacement. Further, a structural bandpass filter is applied to remove disturbances on a larger scale, caused by clouds and limb darkening and additive noise. In the next step we derive discriminative features per pixel and learn a classification model. This model is then used in a variational multi-label segmentation algorithm to assign each pixel to a class label. The final postprocessing step includes the identification and tracking of the solar objects in the image sequence and the computation of properties to categorize them. In an experimental evaluation we demonstrate the efficiency of our method by comparing it to expert detected and annotated events. The method will be used in the context of the space weather program of the European space agency and can further be used for the statistical analysis of flares and filaments by solar physicists.

Keywords. solar image, Hα image, filament, flare, total variation, variational methods, convex optimization, structural bandpass filter, multi-label segmentation, gpu

Acknowledgments

The master thesis marks an end and equally a beginning. It is a symbol that stands for a milestone on a journey of education, but also for the beginning of some new experiences that will follow. Therefore I want to use this opportunity to thank the people who supported my way to complete my goals and especially to finish this master thesis.

First of all I have to thank my whole family, who enabled my educational path and encouraged me all the time. In these five years of study I met many like-minded people and some of them have become real friends. I want to thank them for all the inspiring discussions and adventures we shared. Many thanks to all the members of the Institute of Computer Graphics and Vision, in particular to DI Gottfried Graber, with whom I shared an office for several months and who gave me some helpful tips on implementation.

Finally, special thanks to all the people involved in this project, Ass.-Prof. DI Dr. Thomas Pock for his guidance and answering all the questions arising during this thesis and Assoz. Univ.-Prof. Mag. Dr. Astrid Veronig and Mag. Dr. Werner Pötzi from the University of Graz who made it possible to be part of this fascinating project.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Contents

1.1 Motivation

The sun, our host star, has beyond all doubt a huge influence on our planet, however, the sun is a rather complex and active system. There exist the solar activity cycle of 11 years that can be measured by the number of sun spots that appear on the solar disk, and the magnetic cycle of 22 years [5]. Other events occur on a shorter time scale of a few seconds to several hours. These explosive events include solar flares, filament eruptions and coronal mass ejections (*CME*s).

Back in the early 19th century, the astronomer William Herschel believed to have found a correlation between the number of sun spots and the grain prices [43]. Although these findings were false, others have found a true connection between the number of sun spots and things in nature such as tree rings and rocks [43].

Over the last decades, mankind has developed many technologically advanced space systems, satellite-based services and also ground-based infrastructure that is influenced by space weather. The space weather is defined by Bhatnager et al. [12] as the "changing conditions in interplanetary space and Earth's magnetosphere controlled by the variable solar wind".

The work of Echer et al. [37] and the report by the Royal Academy of Engineering [23] give an overview of the various impacts of the solar weather on Earth's natural systems and technologies in space and on Earth. In space, this includes spacecraft charging, single bit flips in digital microelectronic circuits, and increased space radiation for astronauts. Ground-to-air and ship-to-shore communication and navigation systems can also be disturbed due to the caused changes in the ionosphere. Furthermore, the electrical power infrastructure and pipelines can be affected by the induced currents created by geomagnetic storms.

Following the Ministerial council in 2008, ESA's Space Situational Programme (*SSA*) was set in place, starting with a preparatory phase in 2009, in order to address on a European-wide level the mitigation of risks related to the conditions in space. In this context, Space Situational Awareness (SSA) is "defined as a comprehensive knowledge, understanding and maintained awareness of the (i) population of space objects, of the (ii) space environment, and of the (iii) existing threats/risks" [14]. The risks addressed to in the SSA program are basically divided into three categories: survey and tracking of man-made objects in space, surveillance of near earth objects (*NEO*s), and surveillance of the space weather. The main objective of the SSA system is "to support the European independent utilization of and access to space for research or services through providing timely and quality data, information, services and knowledge regarding the environment, the threats, and the sustainable exploitation of the outer space surrounding our planet Earth" [14].

1.2 Aims

This work is part of the SSA space weather segment in which the University of Graz is developing a service for providing near realtime Hα images and movies from the ground-based facility Kanzelhöhe Observatory for Solar and Environmental Research (*KSO*). Figure 1.1 shows the observatory and the used telescope. Additionally, this master thesis deals with the problem of a near realtime recognition of solar flares and erupting filaments in the provided Hα images. This includes the automated segmentation of filaments and flares in the image sequences and further to derive properties of flares, such as intensity maximum and mean, size and time of occurrence, and filaments, like length and time of eruption.

One major problem in ground-based observations is the influence of clouds and the generally varying sight due to the Earth's atmosphere. The poor viewing conditions make the detection more difficult, because the exposure time not only varies for the images,

Figure 1.1: The left image depicts the Kanzelhöhe Observatory for Solar and Environmental Research on the Gerlitzen. The right image shows the used telescopes for the solar observation in the doom.

which results in a differing mean brightness of the pixels, but there may also occur gaps in the image sequences. Another problem is the limb darkening meaning, that the intensity decreases significantly from the center toward the limb. This is caused by viewing the higher photospheric layers of decreasing temperature as we look closer the limb [5].

1.2.1 Hα Images

The sun consists of several layers, but only the photosphere, or solar surface, is observable by integrating the white light [5]. The layer between the corona and the photosphere is called the chromosphere, which can be seen for a few seconds during a solar eclipse due to the strong photospheric background. To observe the chromospheric features like filaments, flares, plagues and fibrils on the solar disk independently of a solar eclipse, one must use narrow bandpass filters in spectral lines that suit the chromospheric emission lines. Common filters are Hydrogen (Hα) or ionized Calcium and Magnesium based [5].

In figure 1.2, a typical Hα image is shown and describes some of the visible features. The brighter regions are called plages (French for beach) and are incandescent regions of gas with higher density. Together with sun spots, the plages define the spatial extent of active regions that have an enhanced emission over a broad spectral range compared with the quiet sun [5]. Further, we see mottled structures, which mark the boundaries of the chromospheric network [12].

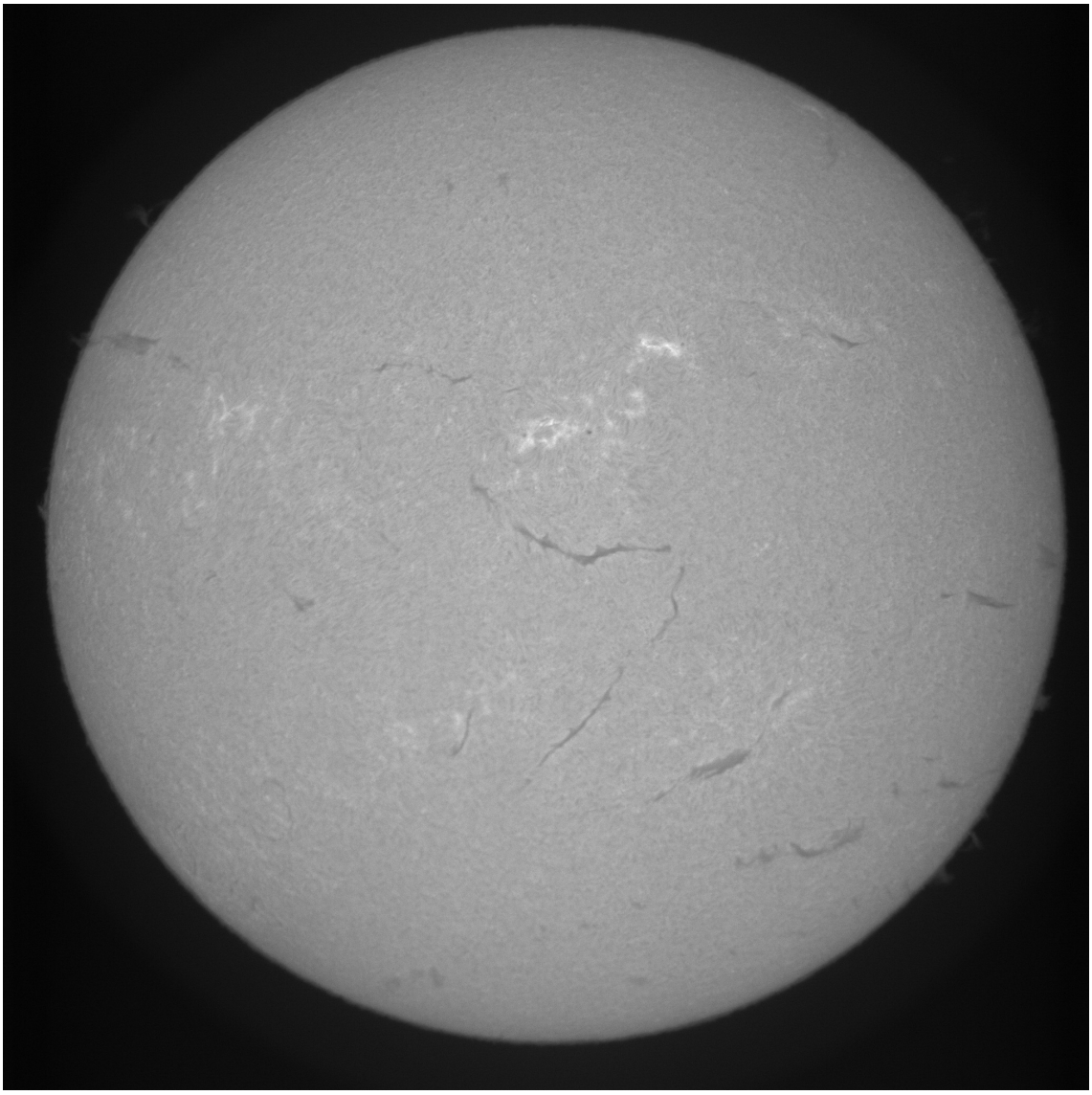
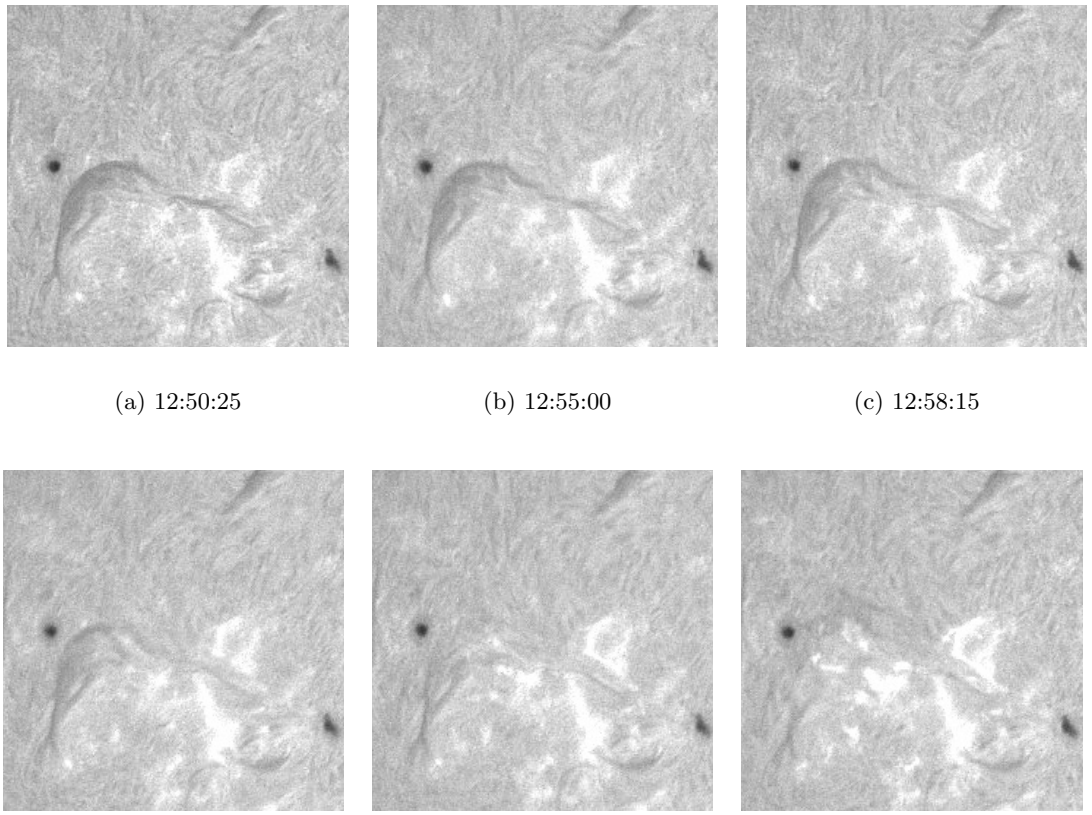


Figure 1.2: A typical Hα image of the sun from the KSO taken on 11.12.2012 at 10:42:01. Filaments appear as elongated, darker structures and can have an arbitary shape. The brighter regions are called plages and differ to flares, which appear drastically brighter compared to the solar surface. Further, one can observe tiny cirle-shaped sun spots that are even darker than filaments.

1.2.2 Filaments

Filaments appear as elongated regions of low intensity on the solar disk of Hα images. They have typically about 10% of the disk intensity, but are still brighter than the sky [91]. Filaments are among the longest-living solar features and can exist for several days. However, a filament may erupt and therefore disappear from the Hα image within several minutes. This is shown in figure 1.3. Some filament eruptions are accompanied by a flare [91] and filament eruptions also occur in a close association with coronal mass ejections (*CME*) that may again have the above mentioned impacts on Earth and earth near objects [56]. The detected location of a filament eruption therefore provides information, if the CME is heading towards Earth.



(a) 12:50:25 (b) 12:55:00 (c) 12:58:15

(d) 13:01:01 (e) 13:04:06 (f) 13:07:02

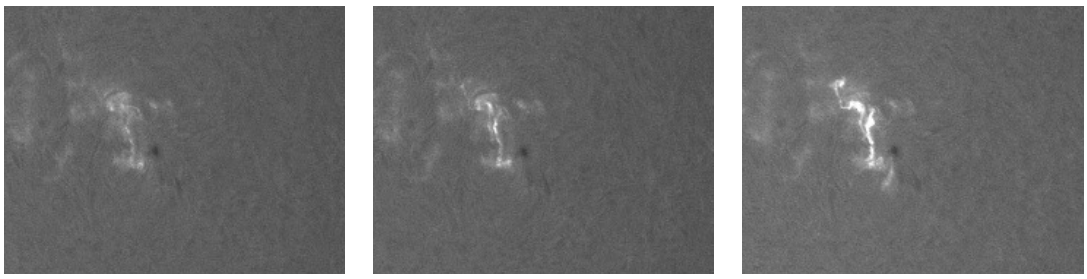
Figure 1.3: The image sequence shows an erupting filament on Hα image details. The images were taken on 09.11.2011 by KSO.

1.2.3 Flares

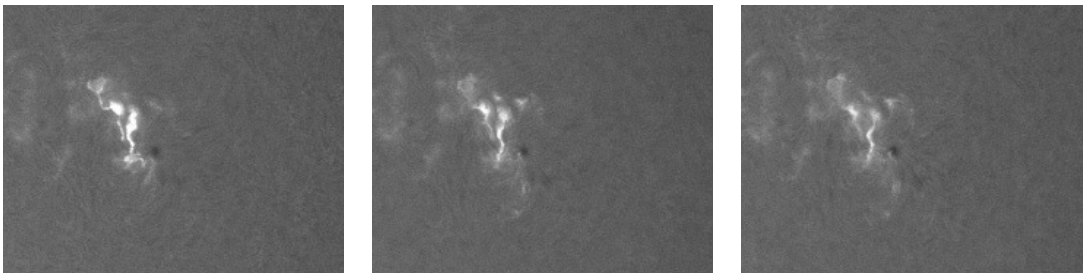
Solar flares are abrupt and enormous releases of energy in a very short time. In a relatively small area of the sun, a single solar flare can create an explosion equivalent to several billion hydrogen bombs [12]. The radiation is enhanced over the whole electromagnetic spectrum, caused by accelerated particle beams and excessive heating of the different layers of the solar atmosphere [5].

Flares are characterized in Hα images by strong brightness increases of localized solar areas that reach the maximum extent and intensity within a few minutes up to an hour [82], followed by a gradual decay of intensity. The evolution of a flare observed by KSO is illustrated in figure 1.4.

The strength of solar flares is classified by two different systems. The first scheme is based on the soft X-ray flux and the second is based on the area and brightness of the solar flare observed in Hα [12]. In table 1.1, the classification based on increasing areas of the flaring region can be found. In addition, a flare may be further classified according to the peak intensity relative to the quiet sun.



(a) 08:15:16 (b) 08:17:36 (c) 08:20:09



(d) 08:23:18 (e) 08:29:26 (f) 08:33:00

Figure 1.4: The image sequence shows an occurring flare on Hα image details. The images were taken on 27.04.2012 by KSO.

Classification	Flare area (10^{-6} of solar hemisphere)
S[ubflare]	< 200
1	$200 - 500$
2	$500 - 1200$
3	$1200 - 2400$
4	> 2400

Table 1.1: Classification of solar flares based on the area in solar hemispheres. Taken from [12].

1.3 Related Work

Image processing in connection with Hα images is quite a new topic. Back in the 1960s, where sun spots where still recorded by hand from a projection, the automated recognition of solar features got more common with the increasing computation power and amount of image data. Starting in 2000, publications were made available that dealt with solar feature recognition in the context of solar imaging, especially Hα images.

Aschwanden [6] provides an overview of image processing techniques in the context of solar physics. Independent of the type of solar image, there exist common processing steps. Before any recognition is performed, the image undergoes some preprocessing steps. Typically, this includes the removal of noise, the spatial registration and the enhancement of the solar features of interest. Next, the actual recognition is performed, followed by additional postprocessing steps. The latter involve the visualization and representation of the recognition. The last step is the computation of the solar features dependent properties from the results.

We will now focus on the research concerning flare and filament recognition in more detail.

1.3.1 Flare Recognition

The first study related to the automatic flare recognition in Hα images was conducted by Veronig et al. [84]. They first extracted the center and the radius of the solar disk by filtering the image with a large median filter and fitted a circle to the gradient image in a least squares manner. The limb darkening is removed by fitting polynomials to annuli from center to limb. For the actual flare segmentation, they selected key pixels that had twice the intensity of the solar mean intensity and from these key pixels a region growing segmentation is performed (see section 2.5) that stops if the intensity of the segmentation exceeds a certain level or reaches a Canny edge pixel [24].

Borda et al. [38] utilized an artificial neural network for the classification of a flare occurrence in Hα images. Rather than segmenting the flare in the image, they made a binary decision, if an image contained a flare or not. This is based on seven extracted features, namely:

1. Mean intensity of the image
2. Standard deviation of the intensity
3. Intensity difference of the key pixel, which is the pixel with the largest intensity difference between two consecutive images
4. Absolute intensity of the key pixel
5. Radial position of the key pixel
6. Variation of mean intensity between two consecutive images
7. Contrast between the point with maximum intensity change and its first neighbor

Features 1-4 are devoted for to recognize images with a lot of activity. With feature 5, the limb darkening is taken into account, whereas features 6 and 7 distinguish between flares and weather-related effects.

In [71], Qu et al. extend the ideas by Borda et al. and Veronig et al. by first extending the set of features with two additional features:

9. Mean intensity of a 50×50 window, where the key pixel is the center
10. Standard deviation of the pixels in the 50×50 window

The key pixel beeing the one defined in feature 3. These two additional features provide localized information about the flare. Qu et al. used a huge set of labeled training images to train an artificial neural network with sigmoidal activation function, one with radial basis activation functions (*RBF*) and one with a support vector machine (*SVM*, see section 2.6.2), whereas the SVM gained the best results on an independent test set. To segment the flares, the algorithm proposed by Veronig et al. [84] is used.

In a subsequent paper, Qu et al. [70] improved their method by adding a pre- and postprocessing step and extending the region growing algorithm. In the preprocessing step they enhanced the image by applying a median filter as well as morphological filters to reduce noise. As the region growing looses some details of the flare, especially on the

boundaries, they added an adaptive boundary method. Therefore, they calculated the first and second derivative to detect the edges of the flare, whereas the threshold for the edge is obtained by the mean and standard deviation of the 8-neighboring windows. The result is then smoothed by morphological closing, and holes in flares are filled to achieve a smoother result.

1.3.2 Filament Recognition

The existing approaches for filament detection can roughly be divided into three categories. The first approach is based on region growing from detected seed pixels [1, 44, 45, 47, 69], whereas the second approach is categorized by a pixelwise binary classification; whether a pixel of the solar image belongs to a filament or not is assigned by a previously learned classifier [89, 90]. Thresholding based on the pixel intensities is the third common method for filament detection [11, 72, 79, 87]. This approach can further be divided in global thresholding and adaptive thresholding.

Before taking a more detailed look on the different methods, the existing preprocessing strategies should be presented. All three methods rely only on the pixel intensities and the knowledge that filaments in Hα images appear darker than other solar features. However, as mentioned earlier, we observe the limb darkening, where the intensities from the center towards the limb decrease. This obviously leads to many false positive detections near the limb. Therefore, most preprocessing is dedicated to this problem.

Fuller et al. [44, 45] applied a large median filter to approximate the large-scale fluctuations and subtract the result from the original image. Another common method is fitting a polynomial g of degree ≥ 2 to the image f and subtract it [11, 79, 87]. The polynomial g is given by the solution of the least-squares optimization problem

$$g = \arg\min_{g} \sum_{x} \sum_{y} \left(f(x,y) - g(x,y)\right)^2 \tag{1.1}$$

To further enhance the filaments and discriminate them from the background, the application of a Laplacian filter is used in [1, 44, 45, 87].

The first and most simple approach to filament segmentation was done by Gao et al. [47]. They used a global threshold that was set to the half of the image median. If a pixel value is less than this threshold, it belongs to a filament and vice versa. After the segmentation, small filament parts were removed. From this initial detection, they perform region growing to merge filament parts belonging together.

A similar approach was taken by Fuller et al. [44, 45]. Instead of a global threshold to obtain initial filament pixels, they used a localized threshold. The local threshold in their method was based on the mean and standard deviation of the intensities in the local neighborhood.

Shih et al. presented in their work [79], a global thresholding method and an adaptive one. In the adaptive approach, a center pixel (x, y) in a $k \times k$ window is segmented as filament, if it is less than a threshold t, whereas the threshold in this window is defined as a clamped median

$$t = \begin{cases} c_{\text{inf}} & \text{if } \text{median}_k(x, y) \leq c_{\text{inf}} \\ \text{median}_k(x, y) & \text{if } c_{\text{inf}} < \text{median}_k(x, y) < c_{\text{sup}} \\ c_{\text{sup}} & \text{if } \text{median}_k(x, y) \geq c_{\text{sup}} \end{cases} \tag{1.2}$$

To further increase the quality of the segmentation, they applied morphological operations. Hence, they removed small segments and emphasized elongated structures.

Zharkova et al. [89, 90] used a learning approach for a pixelwise classification. Therefore, they trained a small artificial neural network with 2 hidden neurons. As input for the network, they used the 3×3 pixels around the central pixel. They concluded that with their method they achieved the same accuracy as with the region growing method by Fuller et al. [45], but in a much more efficient way.

1.4 Thesis Overview

This thesis is structured as follows: Chapter 1 provides an introduction to the related solar events visible in Hα images and states related work. The basic principles for our proposed method are presented in chapter 2. Chapter 3 describes our proposed method for the flare and filament detection in Hα images, while chapter 4 is devoted to the actual implementation of the method, where most parts are implemented on the GPU to achieve near realtime results. We present an evaluation of our method in chapter 5 and finally conclude our work in chapter 6.

Chapter 2

Basic Principles

Contents

In this chapter, the basic principles for the developed method will be introduced. First, we will present the used notation for digital images and the corresponding continuous space as well as the discretization of continuous formulations. Then we will explain, why image processing problems are inherently difficult and we will propose variational methods to handle them. To efficiently solve these problems, we will give a short overview of convex optimization and minimization algorithms. As our method relies heavily on image segmentation, we will state the general segmentation problem and proper methods to solve it. The end of this chapter is dedicated to the basic principles of machine learning and suitable machine learning algorithms for our filament and flare detection method.

2.1 Digital Image

In this section, we depict the basic model of an image. An intuitive way to define an image is in terms of the function

$$f : \Omega \to \mathbb{R}^d \tag{2.1}$$

where Ω is the image domain and has 2 dimensions for images and 3 dimensions for volumes or a temporal sequence of images e.g. a video. This function maps a location to a real valued vector, whereas $d = 1$, if the function maps only to intensity values, or $d = 3$, if we handle color images. In the remaining part of this thesis we will just use intensity images, which then reduces the mapping to a scalar.

If the domain is $\Omega = [a_{11}, a_{12}] \times \ldots \times [a_{k1}, a_{k2}] \subset \mathbb{R}^k$, where $[a_{i1}, a_{i2}]$ is a open interval from a_{i1} to a_{i2}, then f describes a continuous image. To process an image with a digital device like a computer, the continuous image gets sampled and quantized to obtain a digital image. This implies the function reduces to

$$f : \Omega \to \mathbb{N}^d \tag{2.2}$$

and the domain is restricted to $\Omega = \{0, \ldots, N_1\} \times \ldots \times \{0, \ldots, N_k\} \subset \mathbb{N}^k$. More details on image formatting can be found in the book of Szeliski [81], and more information on the mathematical formulation of images can be found in [7, 18, 77].

2.1.1 Discretization

In the upcoming section we will discuss optimization schemes and minimization algorithms in the general continuous domain. However, a digital computer can only deal with discrete images. Therefore, we present in the following a discretization scheme, especially for the gradient and divergence operator.

As already discussed, a digital image f is formed by sampling a continuous image g on discrete locations. A two dimensional digital image is usually defined on a regular $M \times N$ Cartesian grid as follows

$$\{(x_i, y_i) = (i \cdot \Delta x, j \cdot \Delta y) | 1 \leq i \leq M, 1 \leq j \leq N, i, j \in \mathbb{N}\} \tag{2.3}$$

where Δx and Δy denote to the sample interval and define the pixel size of the image.

A critical operator for the minimization is the gradient operator and we define the

discrete gradient operator at (i, j) as

$$(\nabla f)_{i,j} = (\delta_x^+ f_{i,j}, \delta_y^+ f_{i,j})^T \tag{2.4}$$

The derivatives are approximated using finite forward differences with Neumann boundary conditions:

$$\delta_x^+ f_{i,j} = \begin{cases} \frac{f_{i+1,j} - f_{i,j}}{\Delta x} & \text{if } i < M \\ 0 & \text{if } i = M \end{cases} \tag{2.5}$$

$$\delta_y^+ f_{i,j} = \begin{cases} \frac{f_{i,j+1} - f_{i,j}}{\Delta y} & \text{if } j < N \\ 0 & \text{if } j = N \end{cases} \tag{2.6}$$

We further need the divergence operator that is adjoint to the gradient operator, which means in the continuous domain $\langle -\operatorname{div} \boldsymbol{p}, f \rangle = \langle \boldsymbol{p}, \nabla f \rangle$. To fulfill this property, we define the discrete divergence operator of $\boldsymbol{p} = (p^{(1)}, p^{(2)})^T$ as

$$\operatorname{div} \boldsymbol{p}_{i,j} = \delta_x^- p_{i,j}^{(1)} + \delta_y^- p_{i,j}^{(2)} \tag{2.7}$$

whereas we use finite backward differences with Dirichlet boundary conditions

$$\delta_x^- p_{i,j}^{(1)} = \begin{cases} \frac{p_{i,j}^{(1)} - p_{i-1,j}^{(1)}}{\Delta x} & \text{if } 1 < i < M \\ p_{i,j}^{(1)} & \text{if } i = 1 \\ -p_{i-1,j}^{(1)} & \text{if } i = M \end{cases} \tag{2.8}$$

$$\delta_y^- p_{i,j}^{(2)} = \begin{cases} \frac{p_{i,j}^{(2)} - p_{i,j-1}^{(2)}}{\Delta y} & \text{if } 1 < j < N \\ p_{i,j}^{(2)} & \text{if } j = 1 \\ -p_{i,j-1}^{(2)} & \text{if } j = N \end{cases} \tag{2.9}$$

Without loss of generality, we assume for the remainder of this thesis $\Delta x = \Delta y = 1$. According to [26], the gradient operator can be bounded as follows

$$L^2 = ||\nabla||^2 = ||\operatorname{div}||^2 \leq \frac{8}{\Delta x^2} \tag{2.10}$$

2.2 Ill-Posed Problems

In the field of image processing we have to deal with inverse problems. We have given a measurement or observation, which is the image itself, and want to infer further higher level information from it. For example, we have given a function g, which is obtained by an affine transformation of u: $g = Au + n$. To obtain g from u given A and n is the direct or forward problem and well-posed. Conversely, to deduce u only given g is typically ill-posed.

The definition of a well-posed problem was first given by Jacques Hadamard [49]. According to his definition a well-posed problem has to fulfill three properties:

Existence A solution exists

Uniqueness The solution is unique

Continuity The solution continuously depends on the data

If one property is not fulfilled, the problem is said to be ill-posed.

2.2.1 Bayesian Approach

A tempting technique to solve such problems is to treat u as a random variable with some underlying probability distribution and the goal is to find a hypothesis u^* that maximizes the probability based on the observation f [52–54]. We obtain the following optimization problem, also known as maximum a posteriori (MAP) estimation

$$u^* = \arg\max_u p(u|f) \tag{2.11}$$

$$= \arg\max_u p(f|u)p(u) \tag{2.12}$$

The second equation (2.12) arises from the well-known Bayes' theorem that states

$$p(u|f) = \frac{p(f|u)p(u)}{p(f)} \tag{2.13}$$

We get back to our example from above with the affine transformation, where A is any linear operator of u and assuming that n is Gaussian additive noise with zero mean and standard deviation σ_l. Then, the likelihood for g conditioned on u is the same as for

$n = g - Au$

$$p(g|u) = \frac{1}{\sqrt{2\pi\sigma_l^2}} \prod_{x\in\Omega} e^{-\frac{(g(x)-Au(x))^2}{2\sigma_l^2}} \tag{2.14}$$

Next, we assume a Gaussian a priori probability distribution for an image u with standard deviation σ_p.

$$p(u) = \frac{1}{\sqrt{2\pi\sigma_p^2}} \prod_{x\in\Omega} e^{-\frac{\phi(u(x))^2}{2\sigma_p^2}} \tag{2.15}$$

where ϕ is a function of u like the gradient magnitude $|\nabla u|$. Using these both density functions and plug them into equation (2.12) we deduce the following MAP estimator

$$u^* = \arg\max_u \frac{1}{2\pi\sigma_p\sigma_l} \prod_{x\in\Omega} e^{-\frac{(g(x)-Au(x))^2}{2\sigma_l^2} - \frac{\phi(u(x))^2}{2\sigma_p^2}} \tag{2.16}$$

$$= \arg\max_u e^{-\int_\Omega \frac{(g(x)-Au(x))^2}{2\sigma_l^2} + \frac{\phi(u(x))^2}{2\sigma_p^2}\,\mathrm{d}x} \tag{2.17}$$

$$= \arg\min_u \int_\Omega \frac{(g(x)-Au(x))^2}{2\sigma_l^2} + \frac{\phi(u(x))^2}{2\sigma_p^2}\,\mathrm{d}x \tag{2.18}$$

However, the MAP may not always yield the desired result, because the solution with the highest probability might be very unusual as it is shown for example in [27] and also depicted in figure 2.1.

2.2.2 Variational Methods

Another approach to solve the example from above is to use the least squares approach, invented independently by Gauß and Legendre [55]. The idea is to minimize the sum of errors and can be written as follows

$$u^* = \arg\min_u \frac{1}{2} \int_\Omega (f(x) - Au(x))^2\,\mathrm{d}x \tag{2.19}$$

However, this can be problematic due to the linear transformation that may be ill-conditioned or singular in a discrete setting. We can overcome this problem, if we add a

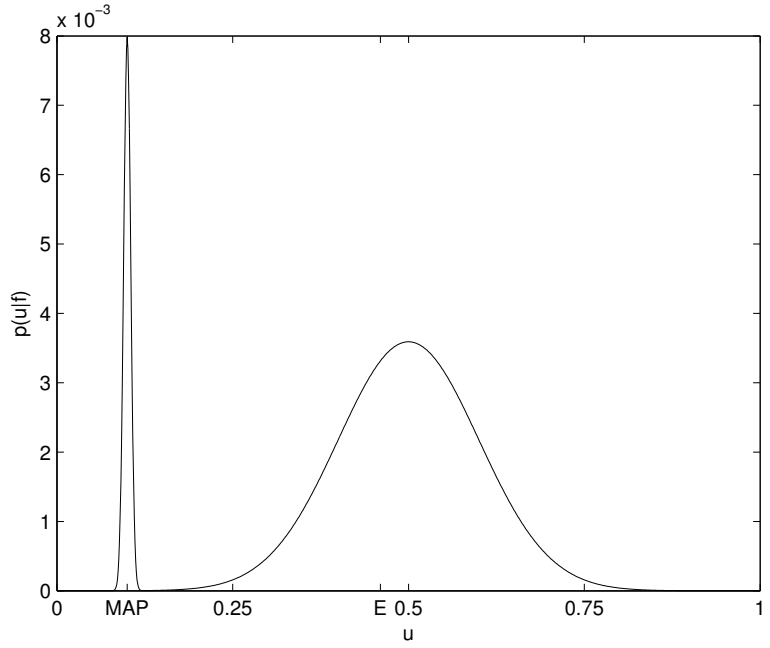


Figure 2.1: A mixture of two Gaussian distributions for $p(u|f)$. This example distribution illustrates that the MAP solution $u = 1$ may be very rare. The expectation value $\mathbb{E}(u|f) = 0.48$ leads to a better solution.

regularization term

$$u^* = \arg\min_u \frac{1}{2} \int_\Omega (f(x) - Au(x))^2 \, \mathrm{d}x + \frac{\lambda}{2} \int_\Omega \phi(u(x))^2 \, \mathrm{d}x \tag{2.20}$$

The general composite criterion was introduced by Tikhonov [83] and the basic idea behind the regularization is to introduce some prior knowledge of u. We observe the direct relationship between the MAP (2.12) and the variational solution (2.20). The quadratic regularization proposed by Tikhonov is the result of a Gaussian prior distribution.

For a more generic range of ill-posed problems, we can write the following optimization scheme

$$u^* = \arg\min_u \Phi(f - u) + \lambda \Psi(u) \tag{2.21}$$

The first term, $\Phi(f - u)$ is called the data fidelity term and it models the relationship between the observation f and the solution u. The second term, $\Psi(u)$, is the regularization term and introduces prior knowledge of the solution u. With the parameter $\lambda > 0$ we can tune the trade-off between data fit and regularization.

The challenge is now, to find a good model for this optimization scheme, so that the solution is near the desired solution. In addition, the computation of the solution should be feasible. Therefore, we will discuss in the next section properties and algorithms for objective functions that allow us a fast computation.

2.3 Convex Optimization

In the following section, we will first introduce the basic definitions and properties for convex optimization. These are mainly based on the books written by Boyd and Vandenberghe [16] and by Rockafellar [74]. Building up on these properties, we demonstrate some minimization algorithms for our models used further.

Definition 1 (Convex Set)**.** *A set C is convex if and only if for $\forall x_1, x_2 \in C$ and $\alpha \in [0,1]$ it holds*

$$\alpha x_1 + (1-\alpha)x_2 \in C$$

This means, if we imagine a line between two points x_1 and x_2 from the set C any point on this line has to be also in the set C.

Definition 2 (Convex Function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if the domain of f is a convex set, $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \operatorname{dom} f$, $\alpha \in [0,1]$ and we have*

$$f(\alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2) \leq \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2)$$

Further we say that f is a concave function if and only if $-f$ is a convex function.

This definition implies, that the line from $(\boldsymbol{x}_1, f(\boldsymbol{x}_1))$ to $(\boldsymbol{x}_2, f(\boldsymbol{x}_2))$ lies above the graph of f. We can further observe that for a convex function f all local minimums are global minimums, but the minimum is not always unique.

Definition 3 (Strictly Convex Function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is strictly convex if and only if the domain of f is a convex set, $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \operatorname{dom} f$, $\alpha \in [0,1]$ and we have*

$$\forall \boldsymbol{x}_1 \neq \boldsymbol{x}_2 : f(\alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2) < \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2)$$

The definition of strictly convex functions enforces in contrast to the convex function that there exists an unique global minimum.

Another definition that will be used later is the following

Definition 4 (Conjugate)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$, then the function $f^* : \mathbb{R}^n \to \mathbb{R}$, defined as*

$$\begin{aligned} f^*(\boldsymbol{x}^*) &= \sup_{\boldsymbol{x} \in \operatorname{dom} f} (\langle \boldsymbol{x}^*, \boldsymbol{x} \rangle - f(\boldsymbol{x})) \\ &= \inf_{\boldsymbol{x} \in \operatorname{dom} f} (f(\boldsymbol{x}) - \langle \boldsymbol{x}^*, \boldsymbol{x} \rangle) \end{aligned}$$

is called the conjugate, or Fenchel-Legendre transformation of the function f.

The conjugate $f^*(\boldsymbol{x}^*)$ of a function $f(\boldsymbol{x})$ is always a convex function and applying the conjugate twice, the function $f^{**}(\boldsymbol{x})$ is the largest lower semi-continuous envelope of the function $f(\boldsymbol{x})$, independent of the shape of f.

2.3.1 Duality

An important concept in optimization in general and especially in convex optimization, is the duality. It allows us to transform a primal problem to a so called dual problem. The solution of the dual problem $\boldsymbol{y}^*$ provides a lower bound to the solution of the primal problem $\boldsymbol{x}^*$, meaning $f(\boldsymbol{y}^*) \leq f(\boldsymbol{x}^*)$. If the primal problem satisfies some conditions, namely that the objective function is convex and Slater's condition is fulfilled, then the optimal solution of the primal problem $f(\boldsymbol{x}^*)$ is equal the optimal solution of the dual problem $f^*(\boldsymbol{y}^*)$. We call it strong duality, if $f(\boldsymbol{x}^*) = f^*(\boldsymbol{y}^*)$. For a proof see [16]. The difference $f(\boldsymbol{x}^*) - f^*(\boldsymbol{y}^*)$ is named the primal-dual gap.

2.3.1.1 The Lagrange Dual Function

The Lagrange dual is very useful if we have a general optimization problem with constraints as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f_0(\boldsymbol{x}) \\ \text{s. t.} \quad & f_i(\boldsymbol{x}) \leq 0 \quad i \in \{1, \ldots, m\} \\ & c_i(\boldsymbol{x}) = 0 \quad i \in \{1, \ldots, p\} \end{aligned} \tag{2.22}$$

where $f_i, c_i : \mathbb{R}^n \to \mathbb{R}$. We define the Lagrangian $\Lambda : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ as the objective function augmented with the constraints as follows

$$\Lambda(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \mu_i c_i(\boldsymbol{x}) \tag{2.23}$$

The vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the dual variables associated with the constraints. If we take

the pointwise minimum over $\boldsymbol{x}$ of the Lagrangian, we derive the Lagrange dual function

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\boldsymbol{x} \in \operatorname{dom} f} \Lambda(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.24}$$

The Lagrange dual is concave, independent of the shape of the problem in (2.22).

2.3.1.2 The Fenchel Dual Function

Another important duality is the Fenchel dual that makes use of the conjugate as defined in Definition 4. If $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and $g : \mathbb{R}^n \to \mathbb{R}$ is a concave function, then it holds

$$\inf_{\boldsymbol{x}} f(\boldsymbol{x}) - g(\boldsymbol{x}) = \sup_{\boldsymbol{y}} g_*(\boldsymbol{y}) - f^*(\boldsymbol{y}) \tag{2.25}$$

where g_* is the concave conjugate of g and f^* is the convex conjugates of the functions f. For a proof of this equality we refer to [74].

2.3.2 Minimization Algorithms

This section gives a brief overview over some minimization algorithms for convex optimization problems and further introduces the primal-dual algorithm by Chambolle and Pock [29] that is used throughout this thesis.

If we consider a convex optimization problem like

$$\min_{u} E(u) \tag{2.26}$$

We can utilize first order methods that only use the first order derivatives of E. The most simple one is the gradient descent method. The first derivative points to the direction that minimizes E and updates of u along this direction converges to the global optimal solution, but the method is rather slow.

Another concept includes the forward-backward methods [10, 35, 62]. They can be used, if the optimization problem can be written in the form

$$\min_{u} F(u) + G(u) \tag{2.27}$$

where F has to be a smooth convex function and continuously differentiable with the

Lipschitz constant L

$$||\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})|| \leq L||\boldsymbol{x} - \boldsymbol{y}|| \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n \tag{2.28}$$

and G has to be convex and has an easy to compute proximal operator. The proximal operator of a function $G(\boldsymbol{x})$ is defined as follows

$$\text{prox}_G(\boldsymbol{y}) = \arg\min_x G(\boldsymbol{x}) + \frac{1}{2\tau}||\boldsymbol{x} - \boldsymbol{y}||^2 \tag{2.29}$$

An efficient proximal point method is the fast iterative shrinkage thresholding algorithm (*FISTA*) [10] and is stated in algorithm 1. The algorithm is guaranteed to converge at least as fast as $\frac{1}{k^2}$.

Algorithm 1 Fast iterative shrinkage algorithm for the minimization problem $\min_u F(u) + G(u)$, with F convex and a L-Lipschitz continuous gradient and g convex and easy to compute proximal operator prox_G.

$x^0 \in X, y^1 = x^0, t^1 = 1$
$h = \frac{1}{L}$
for all $k \geq 1$ **do**
  $x^k = \text{prox}_G(y^k - hF'(y^k))$
  $t^{k+1} = \frac{1}{2} + \frac{\sqrt{1+4(t^k)^2}}{2}$
  $y^{k+1} = x^k + \frac{t^k}{t^{k+1}}(x^k - x^{k-1})$
end for

The optimization algorithm that is used throughout this thesis is the primal-dual algorithm by Chambolle and Pock [29] for non-smooth convex saddle point problems. The generic saddle-point problem is given by

$$\min_{\boldsymbol{x} \in X} \max_{\boldsymbol{y} \in Y} \langle K\boldsymbol{x}, \boldsymbol{y} \rangle + G(\boldsymbol{x}) - F^*(\boldsymbol{y}) \tag{2.30}$$

where X, Y are two finite dimensional real vector spaces and $\langle \cdot, \cdot \rangle$ denotes the inner product, $K : X \to Y$ is a linear operator and G and F^* are proper convex, lower-semicontinuous functions, whereas F^* is the convex conjugate of the function F. The general algorithm to solve this saddle-point problem is given in algorithm 2.

Algorithm 2 Primal-dual algorithm to solve the generic saddle-point problem $\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y)$.

$\tau, \sigma > 0, \theta \in [0,1], x_0 \in X, y_0 \in Y, \hat{x}^0 = x^0$
$\tau\sigma L^2 < 1, L = ||K||^2$
for all $k \geq 1$ **do**
  $y^{k+1} = \text{prox}_{F^*}(y^k + \sigma K \hat{x}^n)$
  $x^{k+1} = \text{prox}_G(x^k - \tau K^* y^{k+1})$
  $\hat{x}^{k+1} = x^{k+1} + \theta(x^{k+1} - x^k)$
end for

2.4 Denoising

In this and the next section, we will present well-known models for low-level computer vision applications. First, we will cover the image denoising problem. Assuming an image $u : \Omega \to \mathbb{R}$ is distorted with additive noise $n : \Omega \to \mathbb{R}$. The goal is to recover from the observation $f = u + n$ the original image u.

One possible solution to this problem has already been presented. We can use the Tikhonov model with the image gradient in the regularization term. This introduces some spatial prior knowledge.

$$u^* = \arg\min_u \frac{1}{2} \int_\Omega (f - u)^2 dx + \frac{\lambda}{2} \int_\Omega |\nabla u|^2 dx \tag{2.31}$$

We get an unique solution that satisfies the associated Euler-Lagrange equation and a closed form solution

$$u - f - \lambda \triangle u = 0 \tag{2.32}$$

$$u = (I - \lambda \triangle)^{-1} f \tag{2.33}$$

where $\triangle u = \sum_i \frac{\partial^2 u}{\partial x_i^2}$ is the Laplacian operator of u.

2.4.1 The ROF Model

The problem with the Tikhonov model is that it does not model sharp edge discontinuities due to the quadratic regularization term. In the work of Rudin, Osher and Fatemi [75] the quadratic regularization is replaced by the total variation norm (TV). In the original

paper the model is formulated as constrained optimization problem

$$\min_u \int_\Omega \mathrm{d}|\nabla u| \tag{2.34}$$

$$\text{s. t. } \int_\Omega (u-f)^2 \,\mathrm{d}x = \sigma^2 \tag{2.35}$$

The main advantage of the TV is that it can preserve sharp edges. If the function u is sufficiently smooth, then TV of u can be written as

$$\mathrm{TV}(u) = \int_\Omega |\nabla u| \,\mathrm{d}x = \int_\Omega \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2} \,\mathrm{d}x \tag{2.36}$$

In the work of Chambolle and Lions [28], the equality constraint is replaced by an inequality constraint $\int_\Omega (u-f)^2 dx \leq \sigma^2$. As we have discussed previously, we can write a convex formulation by using the Lagrangian as

$$\min_u \int_\Omega |\nabla u| \,\mathrm{d}x + \frac{\lambda}{2} \int_\Omega (f-u)^2 \,\mathrm{d}x \tag{2.37}$$

In the discrete setting, this can be expressed as

$$\min_u ||\nabla u||_{2,1} + \frac{\lambda}{2} ||f-u||_2^2 \tag{2.38}$$

The formulation can now be rewritten to match the saddle-point problem as equation (2.30) with $F(\nabla u) = ||\nabla u||_{2,1} = \sum_{i,j} \sqrt{(\delta_x^+ u_{ij})^2 + (\delta_y^+ u_{ij})^2}$ and $G(u) = \frac{\lambda}{2} \sum_{i,j} (f_{ij} - u_{ij})^2$. For the application of the primal-dual algorithm, we further need the convex conjugate $F^*(\boldsymbol{p})$ of the TV that is derived as

$$F^*(\boldsymbol{p}) = \sup_{\nabla u} \langle \boldsymbol{p}, \nabla u \rangle - |\nabla u| \tag{2.39}$$

$$= I_{\{||\boldsymbol{p}||_\infty \leq 1\}}(\boldsymbol{p}) \tag{2.40}$$

where I_C denotes the indicator function on the set $C = \{||\boldsymbol{p}||_\infty \leq 1\}$ and is defined as

$$I_C(x) = \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{else} \end{cases} \tag{2.41}$$

To apply the primal-dual algorithm, we further need the proximal operators of G and F^*. The proximal operator of the indicator function on a convex set is given by a pointwise

projection onto the norm ball

$$\boldsymbol{p} = \text{prox}_{F^*}(\tilde{\boldsymbol{p}}) \Leftrightarrow p_{i,j} = \frac{\tilde{p}_{i,j}}{\max(1, |\tilde{p}_{i,j}|)} \tag{2.42}$$

The proximal operator of G is simply given by the solution to the pointwise quadratic problem

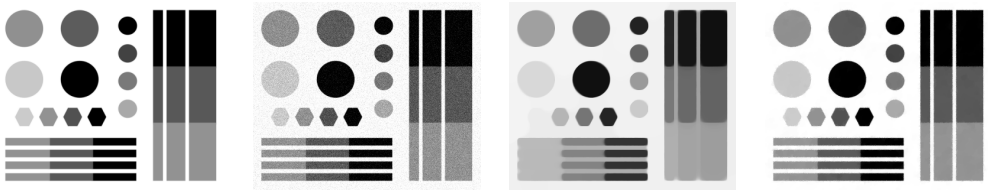
$$u = \text{prox}_G(\tilde{u}) \Leftrightarrow u_{i,j} = \frac{\tilde{u}_{i,j} + \tau f_{i,j}}{1 + \tau\lambda} \tag{2.43}$$

2.4.2 The TV-L^1 Model

We can obtain another denoising model, if we replace the L^2 norm in the data term of (2.37) with the L^1 norm [3, 4, 8, 63, 64]. This yields the TV-L^1 model and it can formally be written as follows

$$\min_u \int_\Omega \mathrm{d}|\nabla u| + \lambda \int_\Omega |f - u| \,\mathrm{d}x \tag{2.44}$$

While these changes may seem minor, they have an interesting impact on the denoising result. In the work of Nikolova [63], it is shown that this model performs better on noise with strong outliers, for example, if the image is disturbed by salt&pepper noise. Another advantage of this model over the ROF model is that it is almost contrast invariant as illustrated in figure 2.2. More properties of the TV-L^1 model are studied in the work of Chan and Esedoḡlu [30]. Especially the findings related to the scale space of an image created by varying the parameter λ will be used in our method in chapter 3.



(a) Input image

(b) Image with additive Gaussian noise ($\mu = 0, \sigma = 0.05$)

(c) Denoising result with the ROF model

(d) Denoising result with the TV-L^1 model

Figure 2.2: A synthetic example for the contrast invariant denoising power of the TV-L^1 model compared to the ROF model.

In the discrete setting, the TV-L^1 model is written as

$$\min_u ||\nabla u||_{2,1} + \lambda||f - u||_1 \tag{2.45}$$

It should be noted that the problem is still convex, but not strictly convex anymore. Therefore, the solution of the optimization problem in (2.45) is not unique. However, we again can utilize the primal-dual algorithm to compute a solution. The problem (2.45) can be rewritten to match the saddle-point formulation in equation (2.30). The function $F(\nabla u) = ||\nabla u||_1$ remains the same and we can use the proximal operator previously defined. For the data term, we now have $G(u) = \lambda||f - u||_1$, where we need to compute the proximal operator. The solution of the proximal operator for $G(u)$ is given by the pointwise shrinkage operation

$$u = \text{prox}_G(\tilde{u}) \Leftrightarrow u_{i,j} = \begin{cases} \tilde{u}_{i,j} - \tau\lambda & \text{if } \tilde{u}_{i,j} - g_{i,j} > \tau\lambda \\ \tilde{u}_{i,j} + \tau\lambda & \text{if } \tilde{u}_{i,j} - g_{i,j} < -\tau\lambda \\ g_{i,j} & \text{if } |\tilde{u}_{i,j} - g_{i,j}| \leq \tau\lambda \end{cases} \tag{2.46}$$

2.5 Segmentation

In this section we deal with the problem of image segmentation, which is one of the most fundamental topics in image processing. In the segmentation process, pixels should be grouped together to form coherent regions or objects. Formally, we can define the segmentation of the image domain Ω into N non-overlapping regions as follows

$$\Omega = \bigcup_{l=1}^{N} \Omega_l, \quad \Omega_i \cup \Omega_j = \emptyset \ \ \forall i \neq j \tag{2.47}$$

In the simplest case, we deal with $N = 2$ regions, which can be seen as the segmentation into foreground and background, where the foreground, for example is an object and the background anything else but the object. Even in the simplest case, the task is clearly highly ambiguous and usually prior knowledge is incorporated. There exists a wide range of possible approaches, which differ mainly by the image features that are used.

One straight forward technique is thresholding based on the image intensity values. For the binary image segmentation task, we can segment an image f into background

(value 0) and foreground (value 1) as follows

$$s(x,y) = \begin{cases} 0 & \text{if } f(x,y) < t \\ 1 & \text{if } f(x,y) \geq t \end{cases} \tag{2.48}$$

where s is the segmentation result and t the used threshold, which can be a fixed value, or depend on the actual image. If the threshold is the same for all pixels in the image domain, we call it global thresholding. In contrast, if the threshold depends on the pixel position and is computed from a local neighborhood, we call it local or adaptive thresholding. The extension for $N > 2$ segments is trivial by using $N - 1$ thresholds. These approaches can be efficiently implemented and parallelized, but share the same problem with overlapping intensity ranges of the objects that should be segmented.

Another segmentation technique we briefly introduce is a region-based one. In a bottom-up approach - the region growing, we start from some initial pixels, and add neighboring pixels, which fulfils a similarity criterion [2]. The criterion can be simply the pixel intensity, but gradient information is often very useful, too. The top-down method starts with the whole image as an initialization and recursively splits the regions until the results are coherent.

A completely different approach to tackle the image segmentation problem is by stating again a regularized optimization schemes. In the rest of this section we will concentrate on solutions of the minimal partition problem that is given as

$$\begin{aligned} &\min_{\{\Omega\}_{l=1}^{N}} \frac{1}{2} \sum_{l=1}^{N} \mathrm{Per}(\Omega_l) + \sum_{l=1}^{N} \int_{\Omega_l} q_l(x)\, \mathrm{d}x \\ &\text{s.t. } \Omega = \bigcup_{l=1}^{N} \Omega_l, \quad \Omega_i \cap \Omega_j = \emptyset \; \forall i \neq j \end{aligned} \tag{2.49}$$

In this formulation q_l represents the unary potentials that indicate, how likely the pixel x belongs to the segment l. In the regularization term, we penalize the perimeter of the segment l.

This continuous formulation also has a discrete counterpart, namely the Pott's model [68]. However, the Pott's model is known to be NP-hard [17] and therefore an optimal solution is not tractable to compute.

The Pott's model is further related to the piecewise constant Mumford-Shah problem. The Mumford-Shah problem [60] is a well-known image segmentation tool and is given as

follows

$$\min_{\Gamma,u} \int_{\Omega\setminus\Gamma} |\nabla u|^2 \,\mathrm{d}x + \lambda \int_{\Omega} (u-f)^2 \,\mathrm{d}x + \mu \mathcal{H}^{n-1}(\Gamma) \tag{2.50}$$

where Γ denotes to the edge set, which defines the border between the segments $\{\Omega\}_{l=1}^{N}$, $\mathcal{H}^{n-1}(\Gamma)$ is the $n-1$ dimensional Hausdorff measure and counts the number of points in Γ. This implies that the first term minimizes the length of the contours, the second term controls the quality of the approximation of f by u, and the third term controls the smoothness of u itself. The parameters λ and μ control the trade-off between the three terms and create a scale space.

We can simplify the problem, if we assume that u_l is locally constant with value c_l. This implies that the integral $\int_{\Omega\setminus\Gamma} |\nabla u|^2 = 0$ and the problem reduces to the piecewise constant Mumford-Shah problem

$$\min_{\Gamma,\{c\}_{l=1}^{N}} \lambda \sum_{l=1}^{N} \int_{\Omega} (c_l - f)^2 \,\mathrm{d}x + \mu \mathcal{H}^{n-1}(\Gamma) \tag{2.51}$$

It is now trivial to observe that the problem in equation (2.49) is related to the problem in equation (2.51) by setting $q_l(x) = (c_l - f(x))^2$.

By now, we have presented the minimal partition problem and the relation to other well-known optimization problems. In the following, we focus on a convex approximation of the problem in equation (2.49) and the algorithmic realization.

For the convex relaxation, the approach by Zach et al. [88] is used, where the minimal partition problem is rewritten as

$$\begin{aligned} \min_{\{u\}_{l=1}^{N}} \sum_{l=1}^{N} \int_{\Omega} \mathrm{d}|\nabla u_l| + \sum_{l=1}^{N} \int_{\Omega} u_l q_l \,\mathrm{d}x \\ \text{s.t. } u_l(x) \geq 0, \quad \sum_{l=1}^{N} u_l(x) = 1 \end{aligned} \tag{2.52}$$

where $u_l : \Omega \to [0,1]$ is the labeling function and q_l the weighting function for the segment l. Further, the regularization term has been replaced by the TV, which is validated by the coarea formula [39]

$$\int_{\Omega} \mathrm{d}|\nabla u| = \int_{\infty}^{\infty} \mathrm{Per}(\{x : u(x) > \gamma\}, \Omega) \,\mathrm{d}\gamma \tag{2.53}$$

For a binary function $u = I_U$, whereas I_U is defined as follows

$$I_U(x) = \begin{cases} 1 & \text{if } x \in U \\ 0 & \text{else} \end{cases} \tag{2.54}$$

the TV is equivalent to the perimeter of the set $U \subset \Omega$.

The relaxed optimization scheme can again be solved by the primal-dual algorithm [29] we have already presented. Therefore the problem in equation (2.52) is rewritten as saddle-point problem as follows

$$\min_{u=\{u_i\}_{l=1}^N} \max_{\boldsymbol{p}=\{\boldsymbol{p_l}\}_{l=1}^N} \sum_{l=1}^{N} \langle \nabla u_l, \boldsymbol{p}_l \rangle + \sum_{l=1}^{N} \langle u_l, q_l \rangle + I_U(u) - I_P(\boldsymbol{p}) \tag{2.55}$$

where we can identify $G(u) = I_U(u)$ and the convex set U is pointwise defined as

$$U = \left\{ \{u\}_{l=1}^N \;\middle|\; (u_l)_{i,j} \geq 0, \sum_{l=1}^{N} (u_l)_{i,j} = 1 \right\} \tag{2.56}$$

and the proximal operator is an orthogonal projector onto the unit simplex defined by the convex set U and can be computed by the method described in [59].

We can further identify the function $F^*(\boldsymbol{p}) = I_P(\boldsymbol{p})$ and the set P is defined as

$$P = \left\{ \{\boldsymbol{p}_l\}_{l=1}^N \;\middle|\; ||\boldsymbol{p}_l||_{2,\infty} \leq \frac{1}{2} \right\} \tag{2.57}$$

and the proximal operator is again the pointwise projection onto the unit ball as in equation (2.42)

2.6 Unary Potential Learning

In the previous section, we used weighting function q_i for the segmentation, but left over how to derive them. In this section, we describe some common methods to learn these functions by inferring unknown information from known. This is covered by the well established field of machine learning and pattern recognition. We will focus especially on supervised learning and classification. In the field of supervised machine learning, we have provided a set of N training examples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ with $x_i \in \mathcal{X} \subseteq \mathbb{R}^n$ and in the classification task $y_i \in \mathcal{Y} = \{1, \ldots, C\} \subset \mathbb{N}$, whereas in the regression task $y_i \in \mathcal{Y} \subseteq \mathbb{R}^m$. The goal of the supervised learning is to find a correct mapping $f : \mathcal{X} \to \mathcal{Y}$ for previously

unseen elements $x \in \mathcal{X}$.

Most algorithms adopt the ideas and tools from the probability theory. These methods try to learn a probability distribution of the form $p(y|x)$. If we assume an identical loss on each wrong prediction, then the optimal mapping is given by $\arg\max_y p(y|x)$ according to the decision theory.

Using again the Bayes' theorem $p(y|x) = p(x|y)p(y)$ we can further distinguish between two categories of methods. In the discriminative model, we learn the posterior probability distribution $p(y|x)$, whereas in the generative model, we learn the likelihood $p(x|y)$ and the prior probability distribution $p(x)$.

In the following, we will cover three methods for supervised classification. For a more detailed overview we refer to the machine learning books [9, 13, 50, 61].

2.6.1 Gaussian Mixture Models

If we consider the generative approach, we can try to model the likelihood $p(x|y)$ with well-known distributions. One common possibility is the normal or Gaussian distribution, which is given by

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_c, \Sigma_c) = \frac{1}{(2\pi)^{\frac{k}{2}}|\Sigma_c|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_c)^T\Sigma_c^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_c)} \tag{2.58}$$

For each class we can now fit a normal distribution by estimating the parameters $\boldsymbol{\mu}_c$ and Σ_c. The maximum likelihood estimator is simple given by

$$\boldsymbol{\mu}_c = \frac{1}{|\mathcal{D}_c|} \sum_{\boldsymbol{x}\in\mathcal{D}_c} \boldsymbol{x} \tag{2.59}$$

$$\Sigma_c = \frac{1}{|\mathcal{D}_c|} \sum_{\boldsymbol{x}\in\mathcal{D}_c} (\boldsymbol{x}-\boldsymbol{\mu}_c)(\boldsymbol{x}-\boldsymbol{\mu}_c)^T \tag{2.60}$$

where $\mathcal{D}_c$ is the set of training vectors from class c. However, a normal distribution often is not the most suitable choice. It is obvious, that any multimodal distribution is not well covered by the normal distribution, but also if the real distribution is skew. Figure 2.3 illustrates the described problems.

A possible solution to these problems is to combine several weighted normal distributions in a Gaussian mixture model. This allows us to approximate almost any probability density to arbitrary accuracy. A Gaussian mixture model with K components is defined

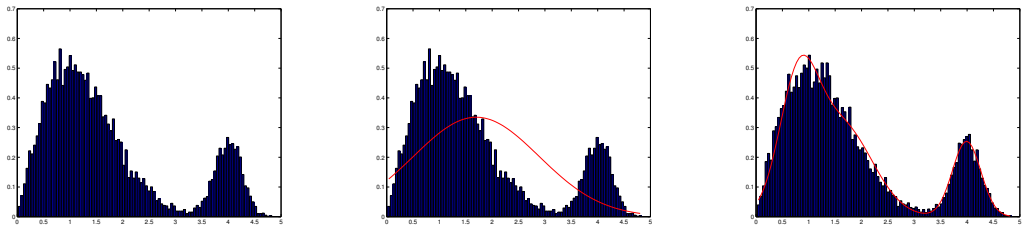
as

$$
\begin{aligned}
p(\boldsymbol{x}|\Theta) &= \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k) \\
\text{s. t. } \sum_{k=1}^{K} \alpha_k &= 1 \quad \alpha_k \geq 0 \; \forall k
\end{aligned}
\tag{2.61}
$$

where the parameter vector Θ consists of the mixing components α_k and the parameters of the normal distributions, $\boldsymbol{\mu}_k$ and Σ_k.

To estimate the parameters Θ we can again use a maximum likelihood approach. This leads to the expectation-maximization (EM) algorithm by Dempster et al. [34]. The algorithm that uses the log-likelihood as convergence criterion is summarized in algorithm 3, whereas the term $\gamma(z_{nk})$ represents the posteriori probability of $\boldsymbol{x}_n$ belonging to the component k.

The illustrated algorithm has some weaknesses, namely the covariance matrices Σ_i can get ill-conditioned or singular and it is not guaranteed that the algorithm converges in a global optimal solution. The reasons for the first problem may be that the dimension of the feature vectors is high, but we have only a few training samples, the variables of the feature space are correlated, or we tried to fit too many components. To overcome this



(a) A bimodal distribution that should be fitted

(b) The red line illustrates a fitted normal distribution

(c) A Gaussian mixture model with three components fits the data well

Figure 2.3: This example presents the usefulness of Gaussian mixture models to fit arbitrary probability distributions. The artificial bimodal distribution in figure a consists of a Rayleigh distribution with $\sigma_R = 1$ and a normal distribution with $\mu = 4$ and $\sigma_N = 0.25$. In figure b we see that a normal distribution fits the real distribution badly, because it is not able to cover the two modes and the skew of the first mode. In contrast, the Gaussian mixture model with three components in figure c covers both properties of the real distribution.

Algorithm 3 EM-algorithm for Gaussian mixture models given N training samples

init Θ randomly
repeat
  $\gamma(z_{nk})^{i+1} = \frac{\alpha_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\Sigma_k)}{\sum_{j=1}^{K} \alpha_j \mathcal{N}(\boldsymbol{x}|\mu_j,\Sigma_j)}$
  for $k = 1$ to K **do**
    $N_k^{i+1} = \sum_{n=1}^{N} \gamma(z_{nk})$
    $\boldsymbol{\mu}_k^{i+1} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n$
    $\Sigma_k^{i+1} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{i+1})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{i+1})^T$
    $\alpha_k^{i+1} = \frac{N_k}{N}$
  end for
until $\sum_{n=1}^{N} \ln \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \Sigma_k)$ converges

problem, we can regularize matrices by adding a small positive number ϵ to the diagonal of every covariance matrix.

2.6.2 Support Vector Machine

The support vector machine is a non-probabilistic binary classifier. The method was invented by Cortes and Vapnik [31] as a linear classifier that maximizes the margin between two classes in the n dimensional space. A good tutorial is the one by Fletcher [40] that states also more details and derivations.

For the binary classification we define $y_i \in \{-1, 1\}$. The goal now is to find a hyperplane that separates these two classes. A hyperplane is defined over a set of points $\boldsymbol{x}$ satisfying $\boldsymbol{w}^T\boldsymbol{x} + b = 0$. However, there are infinite possible hyperplanes and not all are equally good as illustrated in figure 2.4.

An intuitive idea is to maximize the region between the two classes, called the margin. We formulate this by two supporting hyperplanes

$$\boldsymbol{w}^T\boldsymbol{x} + b = 1 \tag{2.62}$$

$$\boldsymbol{w}^T\boldsymbol{x} + b = -1 \tag{2.63}$$

where the distance between these two hyperplanes is simply given by $\frac{2}{||\boldsymbol{w}||}$. Now we can turn the maximization of the margin into the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{w},b} ||\boldsymbol{w}|| \\ &\text{s. t. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 \geq 0 \quad \forall i \end{aligned} \tag{2.64}$$

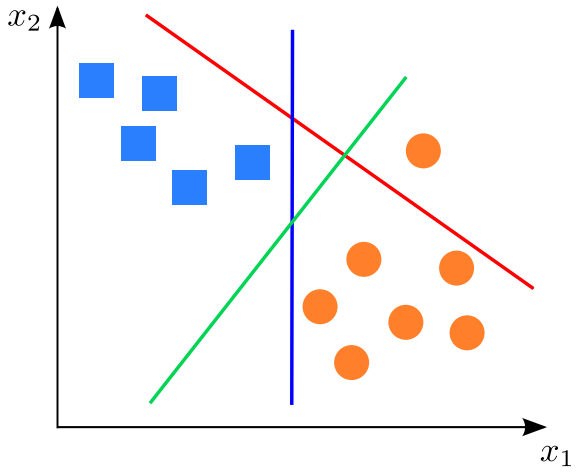


Figure 2.4: The example shows two classes depicted as circles and squares and three different hyperplanes. The red hyperplane is not separating the two classes at all, whereas the blue and the green do separate them. However, the green hyperplane may be a better choice, because it is a better generalization and maximizes the margin between the two classes.

By substituting $||\boldsymbol{w}||$ with $\frac{1}{2}||\boldsymbol{w}||^2$ without changing the solution and incorporating the constraints using Lagrangian multipliers, we derive the following primal problem that can be solved with any quadratic programming solver

$$\min_{\boldsymbol{w},b} \max_{\boldsymbol{\lambda} \geq 0} L_P \equiv \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{N} \lambda_i (y_i(\boldsymbol{w}^T \boldsymbol{x}_i - b) - 1) \tag{2.65}$$

If we take the derivative of L_P with respect to $\boldsymbol{w}$ and b and plug the results into the primal formulation, we obtain the dual problem as follows

$$\frac{\partial L_P}{\boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i \stackrel{!}{=} 0 \Leftrightarrow \boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i \tag{2.66}$$

$$\frac{\partial L_P}{b} = -\sum_{i=1}^{N} \lambda_i y_i \stackrel{!}{=} 0 \tag{2.67}$$

$$L_D \equiv \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j - \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j + \sum_{i=1}^{N} \lambda_i y_i b + \sum_{i=1}^{N} \lambda_i \tag{2.68}$$

$$= \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{2.69}$$

We can again write this result as a quadratic program

$$\begin{aligned} &\max_{\boldsymbol{\lambda}} L_D \\ &\text{s. t. } \lambda_i \geq 0 \; \forall i, \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \end{aligned} \tag{2.70}$$

So far this formulation just allows a linear separation of two non-overlapping classes. We can simply turn this classifier to a non-linear one with the Kernel-trick. If we take a look at the dual formulation in equation (2.69), we observe that the formulation requires only the dot product of the input vectors $\boldsymbol{x}_i$. The dot product $\boldsymbol{x}_i^T \boldsymbol{x}_j$ can be replaced by $K(x_i, x_j) \equiv \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$, where ϕ denotes a non-linear mapping into a higher feature dimension.

However even in the high dimensional feature space the two classes may not be completely linearly separable. Therefore, for every training vector we introduce a slack variable ξ_i that incorporates missclassifications. The optimization problem now can be written as

$$\begin{aligned} &\min_{\boldsymbol{w},b,\boldsymbol{\xi}} ||\boldsymbol{w}|| + C \sum_{i=1}^{N} \xi_i \\ &\text{s. t. } y_i(\boldsymbol{x}^T \boldsymbol{w} + b) - 1 + \xi_i \geq 0, \quad \xi_i \geq 0 \quad \forall i \end{aligned} \tag{2.71}$$

where the parameter C defines the trade-off of the slack variable penalty and the size of the soft-margin. Using the same steps as in the linearly separable case, we can derive the primal and dual formulation

$$\begin{aligned} &\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \max_{\boldsymbol{\lambda} \geq 0, \boldsymbol{\mu} \geq 0} L_P \equiv \frac{1}{2}||\boldsymbol{w}||^2 + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \lambda_i (y_i(\boldsymbol{w}^T \boldsymbol{x}_i - b) - 1) - \sum_{i=1}^{N} \mu_i \xi_i \\ &\text{s. t. } \lambda_i, \mu_i \geq 0 \end{aligned} \tag{2.72}$$

$$\begin{aligned} &\max_{\boldsymbol{\lambda}} L_D \equiv \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\ &\text{s. t. } C \geq \lambda_i \geq 0 \; \forall i, \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \end{aligned} \tag{2.73}$$

Note that in the dual formulation L_D only the constraint has changed.

The last remaining issue is the multi-class classification. Crammer and Singer [32]

proposed a method that solves the multi-class problem as a single optimization problem, but the common approach is to combine several binary SVMs. An overview of possible combinations of binary SVMs is given in [51] and [36].

2.6.3 Random Forests

The last model we take a look at are random forests. Breiman introduced the random forests classifier [21] as a successor of his bagging idea [20]. The model is very popular now in various computer vision applications, for example in pose recognition [80] and object detection [46]. Recently, Criminisi and Shotton [33] published an excellent book dedicated to random forests in the field of computer vision. Further, they gain excellent results in machine learning comparison studies [25] and are inherently multi-class classifiers, in contrast to SVMs.

The main building block is the decision tree that is a special graph, where all nodes, except the leaf nodes, represent split functions (or test functions). These split functions are binary decision functions and determine, if the feature vector is passed to the left or to the right subtree. A feature vector therefore traverses the tree until it reaches a leaf node. In the leaf nodes the class probabilities $p(c|\boldsymbol{x})$ are stored which are derived from the number of training samples, belonging to this leaf node. One example of a decision tree is shown in figure 2.5.

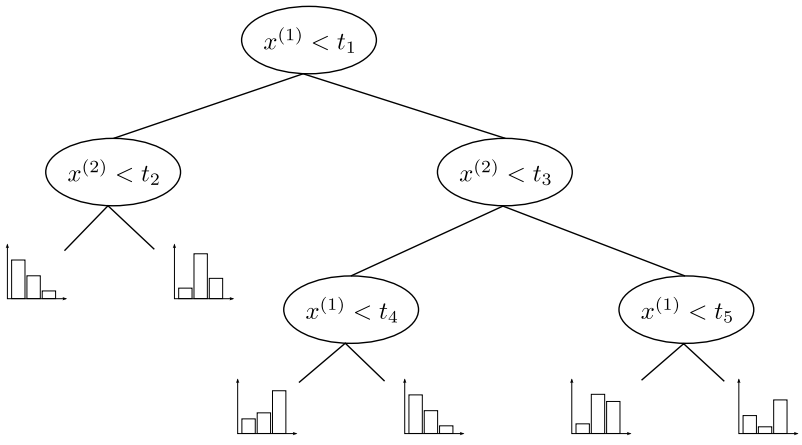


Figure 2.5: A sample decision tree for two dimensional feature vectors and linear, axis-aligned decision functions. Each split partitions the feature space into two non-overlapping subspaces. In the leaf nodes the sample distribution for three different classes is depicted.

There exist many different types of decision trees, varying by the type of split function and also differing by how the algorithm learns these functions. The most simple split

function is the axis-aligned hyperplane which is created by thresholding a single component of the feature vector $\boldsymbol{x}$, e.g. $x^{(1)} < t_i$. But the split functions can also be generic linear functions, e.g. $\boldsymbol{w}^T \cdot [\boldsymbol{x}^T \; 1]^T$, where $\boldsymbol{w}$ defines the separating hyperplane, or also any non-linear mapping, e.g. $[\boldsymbol{x}^T \; 1] \cdot \boldsymbol{W} \cdot [\boldsymbol{x}^T \; 1]^T$, where $\boldsymbol{W}$ is a matrix defining the separating conic.

The split functions are learned in a greedy manner. Therefore, for each node the split function is optimized independently of the parent and child nodes. For the optimization, an energy function is needed that usually is dependent on the purpose of the decision tree. For the classification task, the Gini index, as it is used in the Classification and Regression Tree (*CART*) [19], or the information gain $I(S)$, as it is implemented in the C4.5 [73], can be utilized. If S is the set of training samples at node i, and S^r are the samples that belong to the right child node and S^l vice versa, then the information gain is defined as follows

$$I(S) = H(S) - \frac{|S^r|}{|S|} H(S^r) - \frac{|S^l|}{|S|} H(S^l) \tag{2.74}$$

where $H(\cdot)$ is the sample entropy and the term $H(S)$ can be omitted for the optimization, because it stays the same, independent of the split.

In classic decision tree algorithms, the best parameters for each split function are found in an exhaustive search over all possible hypotheses per node. This has one major drawback: efficiency. In contrast, the random forest approach generates ρ different hypotheses and takes the best, based on the energy function. Further, as the name random forest suggests, it consists of many decorrelated decision trees, combined to form a strong classifier. The decorrelation is achieved by training each tree on a randomly chosen subset of input variables as well as a randomly chosen subset of training samples. Each tree is therefore independently and identically distributed. This fact is used by Breiman [21] in combination with the strong law of large numbers to proof that with increasing the number of trees the random forest do not overfit.

The probability for class c given the feature vector $\boldsymbol{x}$ can then be computed for T decision trees in the forest by averaging the single probabilities as

$$p(c|\boldsymbol{x}) = \frac{1}{T} \sum_{i=1}^{T} p_t(c|\boldsymbol{x}) \tag{2.75}$$

The key parameters of the random forest can be summarized as follow [33]

- The maximum depth of the trees
- The number of decision trees in the forest
- The number ρ of hypotheses per split function
- The type of split function
- The energy function for the split function optimization

For more details on random forests, especially on the efficient implementation on the GPU, we refer to the book written by Criminisi and Shotton [33].

Chapter 3

Method

Contents

In the previous two chapters we introduced the fundamentals of this master thesis. First, we presented solar activity features visible in Hα images, namely filaments and flares, and in the second chapter important concepts of mathematical image processing and machine learning were stated. This chapter builds up on that knowledge to develop our method for the detection of filaments and flares in Hα image sequences.

3.1 Problem Statement

The aim of the proposed method is to solve the near realtime detection of filament eruptions and flares in a sequence of Hα images. The KSO creates up to 10 images per minute depending on the weather and viewing conditions. At least one image per minute should be processed by our method. The images are provided in an uncompressed image format, that allows a dynamical range of the intensity values. Further, the images are rather huge in size with 2048×2048 pixels and 12 bits per pixel.

For the detection of filament eruptions and flares we need a segmentation of every image for the classes *filament*, *flare* and *background*. This allows us a tracking of the

features in the image sequence and furthermore the detection of flare occurrences and filament eruptions. The segmentation results are also used to derive important properties of the objects. For the categorization of the events, the length, the area and the intensity values of the objects are needed.

The ground based observation of the solar disk causes some possible disturbances in the images. One obvious problem can be caused by clouds. The parts of the solar disk covered by a cloud are seen as darker regions. An example is shown in figure 3.1. Another problem is the so called limb darkening, which refers to the decreasing of the intensity from the solar center towards the limb. The atmosphere of the earth has also an influence on the images. The solar disk is not always in the center of the images and also the radius can vary slightly.

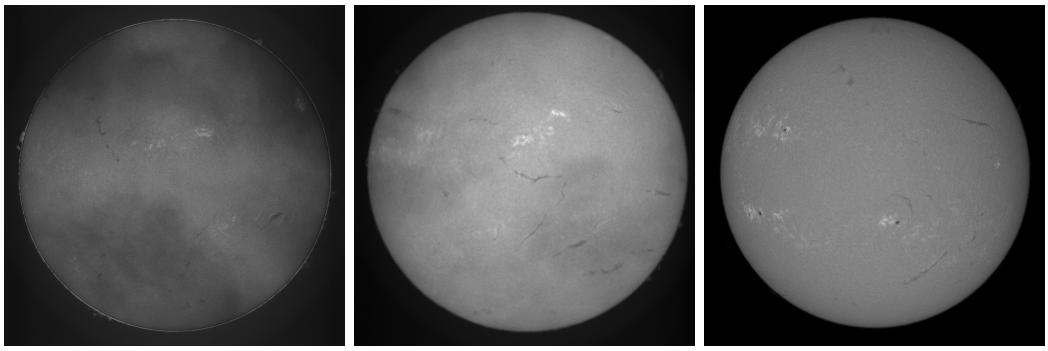


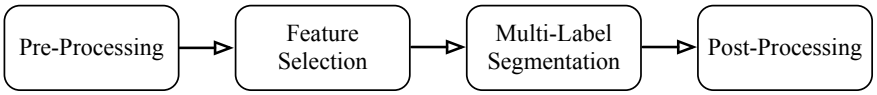
Figure 3.1: The Hα images on the left and in the middle show the effect of clouds that can be seen as darker regions over the solar disk. The limb darkening, which describes the intensity decreasing from the center to the limb, can be observed in the right image.

The problems described clearly have an impact on the image processing pipeline, but the solar events are still observable. However, if the viewing conditions are really bad, it could happen that the sun is not observable at all and so no images can be recorded. This may lead to gaps in the image sequence and should also be addressed to by our method.

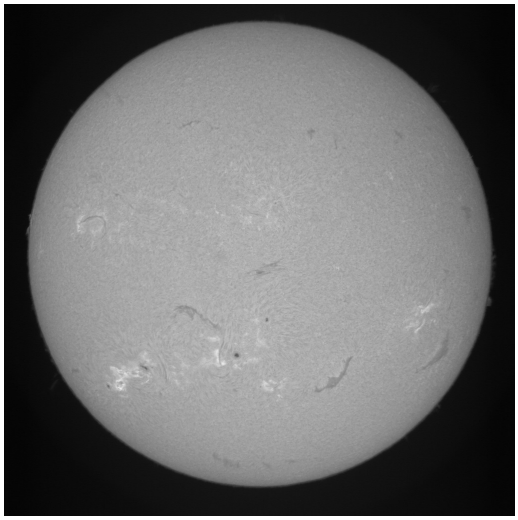
3.2 Overview

In the following, we give a short overview of our method. It consists of four main building blocks like illustrated in figure 3.2.

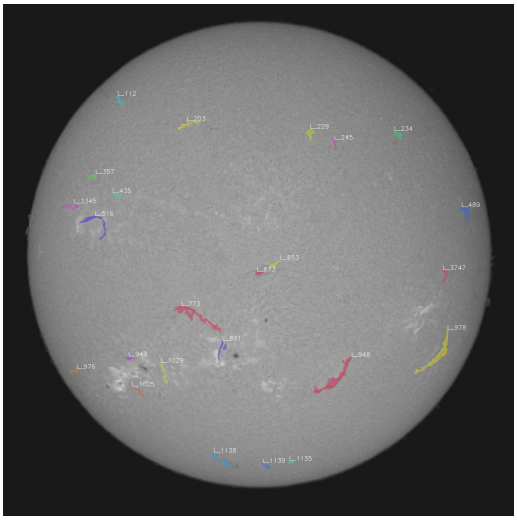
Preprocessing In the first step, the preprocessing, we address the problems such as the different intensity distributions caused by varying viewing conditions. For further



(a) Image Processing Pipeline



(b) Input Image



(c) Final Filament Recognition

Figure 3.2: Overview of our presented method. The method is sequentially provided with Hα images as shown in (b). In the first preprocessing steps the images get normalized in terms of intensity and spatial displacement and further disturbances caused by additive noise, clouds and limb darkening are removed. On the preprocessed images we extract features and learn a model to discriminate especially between filaments, flares and background. This model is then applied in the third step and used for a regularized segmentation approach. In the final postprocessing steps, the solar activity objects are identified and tracked over the image sequence. Additionally, we derive properties like the length, the area and the intensity of the flares and filaments to categorize them. An example segmentation of filaments produced by our method for the input image in (b) is presented in (c).

steps it is also essential to have the images in the sequence aligned, therefore we apply a simple registration technique. As a last preprocessing step, we apply a structural bandpass filter. This filter suppresses additive noise in the images on a small scale, but also filters large-scale intensity variations caused by clouds.

Feature selection In the feature selection step, two questions arise. First, what are the attributes of filaments and of flares - what discriminates them from the solar background? The second question is then, how can we model this efficiently? To cope with the first question, we compute a discriminative feature vector for every pixel. These vectors further will be used in a machine learning model to assign class probabilities to each pixel.

Multi-Label Segmentation After we have learned a model offline from annotated training images, the next step is to apply it to new, unseen images in realtime. The most simple case would be that each pixel gets assigned to the class with the highest probability. However, this would lead to a non-smooth, noisy segmentation. Therefore, we regularize our problem in terms of a variational optimization problem, where we take the contour length of the segmented regions into account.

Postprocessing The last step is the postprocessing, where first of all, every object gets identified with an ID. In the sequence of images the ID of the flares and filaments should stay the same, even if the sequence has gaps. Therefore, we apply a simple tracking approach by propagating the ID through the image sequence. A single filament may be divided into several parts. To address this problem, we apply a reconnection algorithm that incorporates the filament shape. Finally, we derive properties from the filaments and flares to categorize them. What is important is the length of the filaments, and the area in pixels and the intensity values for flares.

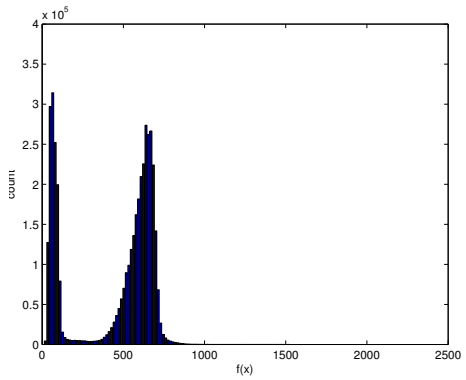
In the following, we discuss the single steps and their impacts in more detail. Each section is dedicated to a block of the workflow.

3.3 Preprocessing

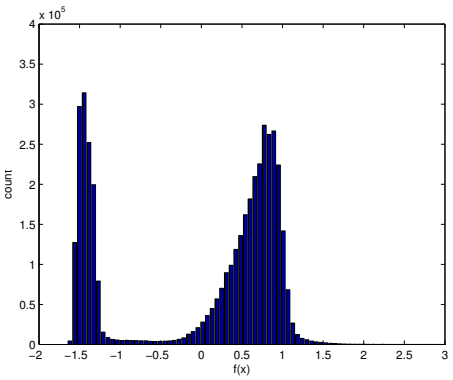
The preprocessing has two goals, namely the image normalization and feature enhancing. Across the different image sequences, but also within a single image sequence, the intensity distribution of the images is shifted and dilated. This effect should be compensated with the intensity normalization. Furthermore, the solar disk is not always in the center of the image. We use a simple, yet powerful technique for the registration of the solar disk across an image sequence. The last preprocessing step is the structural bandpass filter. As the name indicates, the filter removes large-scale structures, caused for example by clouds and limb darkening, but also additive noise on a small scale.

3.3.1 Intensity Normalization

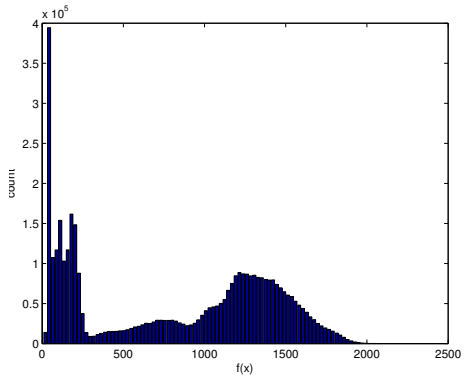
The first step of our method is the intensity normalization. The exposure time of the Hα camera is depending on the viewing conditions. But the exposure time and the state of the atmosphere have an impact on the intensity distribution of the images. The distribution can be shifted and dilated as one can see in figure 3.3.



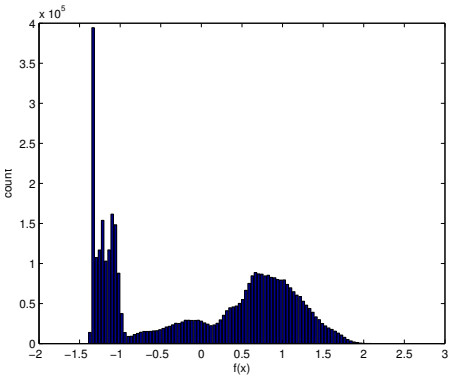
(a) Intensity distribution for a normal input image



(b) Normalized intensity distribution for a normal image



(c) Intensity distribution for an input image with heavy clouds



(d) Normalized intensity distribution for the cloudy image

Figure 3.3: The intensity distributions for two different images are shown on the left. The top left distribution belongs to an image with good viewing conditions without clouds, whereas the second distribution, illustrated on the bottom left, belongs to an image disturbed by large clouds. This causes that the distribution being dilated and the second mode being shifted. The right images present the intensity distribution of the same images that where normalized to a zero mean and a unit standard deviation. The intensity interval is now nearly the same, as well as the location of the maximum of the second mode.

As our feature selection heavily relies on the intensity, we try to minimize this problem. Therefore we normalize the image intensity to a zero mean and a unit standard deviation as follows

$$\mu = \frac{1}{|\Omega|} \sum_{\boldsymbol{x} \in \Omega} f(\boldsymbol{x}) \tag{3.1}$$

$$\sigma = \sqrt{\frac{1}{|\Omega| - 1} \sum_{\boldsymbol{x} \in \Omega} (f(\boldsymbol{x}) - \mu)^2} \tag{3.2}$$

$$f_n(\boldsymbol{x}) = \frac{f(\boldsymbol{x}) - \mu}{\sigma} \tag{3.3}$$

where Ω is the image domain, μ the sample mean and σ the sample standard deviation of the input image f, respectively. The normalized image f_n is given by subtracting the sample mean from the input image and dividing it through the sample standard deviation.

3.3.2 Spatial Registration

Based on the intensity normalized images, the center of the solar disk should stay the same over the whole Hα image sequence. For this registration task, we use the method of Lucas and Kanade [58] to compute a single translation vector $\boldsymbol{u} = (u_1, u_2)^T$ that shifts the center of the second image to the center of the first image. If $g_r = |\nabla f_1|$ refers to the gradient magnitude of the reference image and $g_i = |\nabla f_i|$ to the second image that should be registered, then we obtain the translation vector $\boldsymbol{u}$ by the solution of the following minimization problem

$$E(\boldsymbol{u}) = \sum_{\boldsymbol{x}=(x_1,x_2)} (g_i(x_1 + u_1, x_2 + u_2) - g_r(x_1, x_2))^2 \tag{3.4}$$

$$\boldsymbol{u} = \underset{\boldsymbol{u}=(u_1,u_2)^T}{\arg\min} E(\boldsymbol{u}) \tag{3.5}$$

We can linearize the problem, if we approximate the term $g_i(\boldsymbol{x} + \boldsymbol{u})$ with the linear terms of the Taylor expansion as follows

$$g_i(\boldsymbol{x} + \boldsymbol{u}) = g_i(\boldsymbol{x}) + u_1 \frac{\partial g_i}{\partial x_1}(\boldsymbol{x}) + u_2 \frac{\partial g_i}{\partial x_2}(\boldsymbol{x}) + \text{higher order terms} \tag{3.6}$$

$$\approx g_i(\boldsymbol{x}) + \boldsymbol{u}^T \nabla g_i(\boldsymbol{x}) \tag{3.7}$$

If we plug the linearizion (3.7) into the optimization problem of equation (3.5) we

obtain a linear problem

$$\boldsymbol{u} = \arg\min_{\boldsymbol{u}} \sum_{\boldsymbol{x}} (g_i(\boldsymbol{x}) - g_r(\boldsymbol{x}) + \boldsymbol{u}^T \nabla g_i(\boldsymbol{x}))^2 \tag{3.8}$$

We can further obtain a closed form solution as follows

$$\frac{\partial E(\boldsymbol{u})}{\partial \boldsymbol{u}} \stackrel{!}{=} 0 \Leftrightarrow \tag{3.9}$$

$$2 \sum_{\boldsymbol{x}} (g_i(\boldsymbol{x}) - g_r(\boldsymbol{x}) + \boldsymbol{u}^T \nabla g_i(\boldsymbol{x})) \nabla g_i(\boldsymbol{x}) \stackrel{!}{=} 0 \Leftrightarrow \tag{3.10}$$

$$\sum_{\boldsymbol{x}} (g_r(\boldsymbol{x}) - g_i(\boldsymbol{x})) \nabla g_i(\boldsymbol{x}) = \sum_{\boldsymbol{x}} \nabla g_i(\boldsymbol{x}) (\nabla g_i(\boldsymbol{x}))^T \boldsymbol{u} \Leftrightarrow \tag{3.11}$$

$$\left(\sum_{\boldsymbol{x}} \nabla g_i(\boldsymbol{x}) (\nabla g_i(\boldsymbol{x}))^T \right)^{-1} \sum_{\boldsymbol{x}} (g_r(\boldsymbol{x}) - g_i(\boldsymbol{x})) \nabla g_i(\boldsymbol{x}) = \boldsymbol{u} \tag{3.12}$$

This solution is just a local approximation, because we use only the linear terms of the Taylor expansion. The resulting translation vector can be improved, if we repeat the calculation several times. This is depicted in algorithm 4.

Algorithm 4 Iterative Lucas-Kanade algorithm for image registration. g_r refers to the reference image, g_i to the image for which the translation vector $\boldsymbol{u}_{\text{total}}$ should be computed.

Set $\boldsymbol{u}_{\text{total}} = \boldsymbol{0}$
for $k \leftarrow 1 \dots K$ **do**
  $\boldsymbol{u} \leftarrow \left(\sum_{\boldsymbol{x}} \nabla g_i(\boldsymbol{x}) (\nabla g_i(\boldsymbol{x}))^T \right)^{-1} \sum_{\boldsymbol{x}} (g_r(\boldsymbol{x}) - g_i(\boldsymbol{x})) \nabla g_i(\boldsymbol{x})$
  $g_i(\boldsymbol{x}) = g_i(\boldsymbol{x} + \boldsymbol{u})$
  $\boldsymbol{u}_{\text{total}} \leftarrow \boldsymbol{u}_{\text{total}} + \boldsymbol{u}$
end for

Further, the used Taylor approximation is only valid for small translations. Therefore we use a multi-scale approach to calculate the displacement vector at several scales similar to [15]. For the multi-scale analysis we use a Gaussian image pyramid with L levels. The image at level L is the input image and has the highest resolution. To compute the image at a lower level $l - 1$, we convolute the image at level l with a Gaussian filter and then subsample it with a scaling factor $s \in (0, 1)$. This implies, if the image on level l has a resolution of $m \times n$, the image on level $l - 1$ has a resolution of $tm \times tn$.

With this method we also can compute large translation. The final algorithm for the image registration is listed in algorithm 5.

Algorithm 5 Multi-scale Lucas-Kanade algorithm for image registration. L are the used levels of the Gaussian image pyramids, g_r refers to the reference image, g_i to the image that should be registered by shifting it with the translation vector $\boldsymbol{u}_{\text{total}}$.

Set $\boldsymbol{u}_{\text{total}} = \mathbf{0}$
Compute Gaussian image pyramid $G_r(x, y, l)$ for $g_r(x, y)$, L levels and scale factor s
Compute Gaussian image pyramid $G_i(x, y, l)$ for $g_i(x, y)$, L levels and scale factor s
for $l \leftarrow 1 \dots L$ **do**
  $g_r^l(x, y) \leftarrow G_r(x, y, l)$
  $g_i^l(x, y) \leftarrow G_i(x + u_{\text{total}}^{(1)}, y + u_{\text{total}}^{(2)}, l)$
  Compute translation vector $\boldsymbol{u}$ at scale l with algorithm 4
  $s_{\text{up}} \leftarrow s^{L-l}$
  $\boldsymbol{u}_{\text{total}} \leftarrow \boldsymbol{u}_{\text{total}} + \frac{1}{s_{\text{up}}}\boldsymbol{u}$
end for

3.3.3 Structural Bandpass Filter

As a last step in the preprocessing chain, additive noise and large-scale intensity variations, caused by limb darkening and clouds, are removed by applying a structural bandpass filter.

The core of this particular filter is the TV-L^1 model as described in chapter 2 and repeated in equation (3.13)

$$\min_u ||\nabla u||_{2,1} + \lambda ||f - u||_1. \tag{3.13}$$

where f is the noisy observation, u the solution that minimizes the optimization problem, and λ a free parameter that controls the trade-off between the data fidelity (right term) and the total variation regularization (left term).

The parameter λ has some nice properties that were studied by Chan and Esedoḡlu [30] and exploited in our structural bandpass filter. One property is the relation between the parameter λ and the size of objects that are removed by the model. To illustrate this relation, we present an example by [30].

Assume an image $f(x)$ is given by the indicator function $I_{D_r(0)}(x)$, where $D_r(0)$ is a convex set of points on a disk with center 0 and radius r. In this special case Chan and Esedoḡlu proof that for $\lambda \geq 0$ the solution of the optimization problem in equation (3.13) is equivalent to the solution of the following problem

$$\min_{c \in [0,1]} 2\pi r c + \lambda \pi r^2 |1 - c| \tag{3.14}$$

We can minimize the problem and obtain the set of solutions as follows

$$\begin{cases} \{0\} & \text{if } 0 \leq \lambda < \frac{2}{r} \\ \left\{cI_{D_r(0)}(x) | c \in [0,1]\right\} & \text{if } \lambda = \frac{2}{r} \\ \left\{I_{D_r(0)}(x)\right\} & \text{if } \lambda \geq \frac{2}{r} \end{cases} \tag{3.15}$$

which already reveals some interesting properties. First, the solution of the TV-L^1 model is, as already mentioned, not unique. In this example the solution is unique, except for $\lambda = \frac{2}{r}$, where it can have any contrast between 0 and 1. The second property relates the size of the object with the parameter λ. The disk does not change until λ is equal to $\frac{2}{r}$, then it suddenly merges with the background. This effect is demonstrated in figure 3.4.

We utilize this fact in our structural bandpass filter. First, small scaled noise is removed from the image by using a bigger λ_1. After that, we capture the large-scaled intensity variations caused by clouds and limb darkening and therefore use a smaller $\lambda_2 < \lambda_1$. The result of the structural bandpass filter is then given by the subtraction of the second image v_2 from the first image v_1. The algorithm is also listed in algorithm 6.

Algorithm 6 Structural Bandpass Filter for an input image f

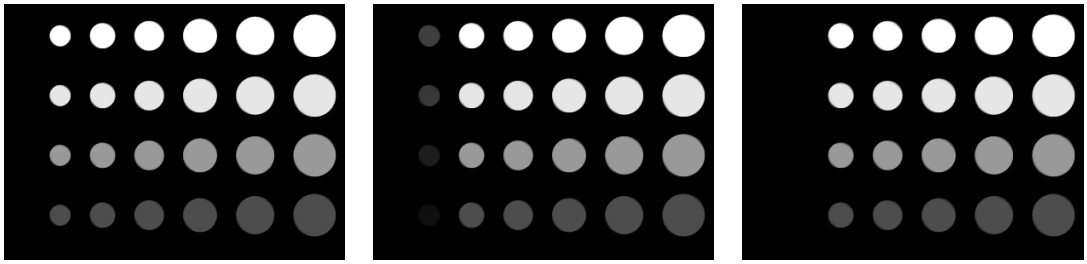
Choose $\lambda_1 > \lambda_2 > 0$
$v_1 \leftarrow \min_u ||\nabla u||_{2,1} + \lambda_1 ||f - u||_1$
$v_2 \leftarrow \min_u ||\nabla u||_{2,1} + \lambda_2 ||v_1 - u||_1$
$u \leftarrow v_1 - v_2$

First, we demonstrate the results of the structural bandpass filter on the artificial circle image. We select the parameters λ_1 and λ_2 in terms so that only circles of a specific radius remain. This is illustrated in figure 3.5. We use the demonstrated fact for the given Hα images to remove noise and large-scale intensity variations. Figure 3.6 depicts the effect of applying the structural bandpass filter to a cloudy Hα image.

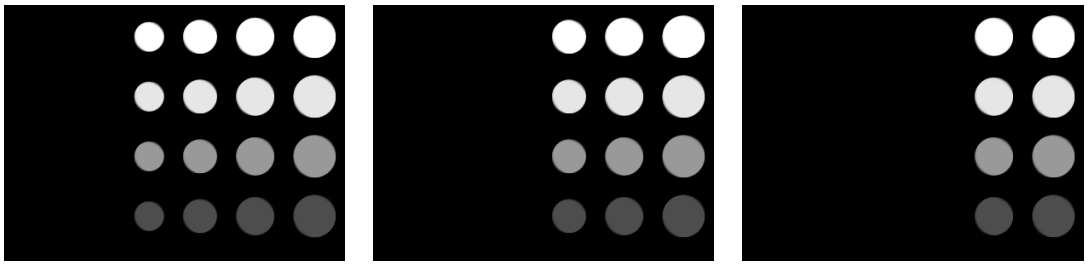
3.4 Feature Selection

This section deals with the problem to assign each pixel of the Hα image to a feature vector $\boldsymbol{\phi} \in \mathcal{R}^k$ that is as discriminative as possible. We will utilize machine learning algorithms on a set of annotated feature vectors to learn a model. In this context *annotated* means that we have a class associated to each feature vector.

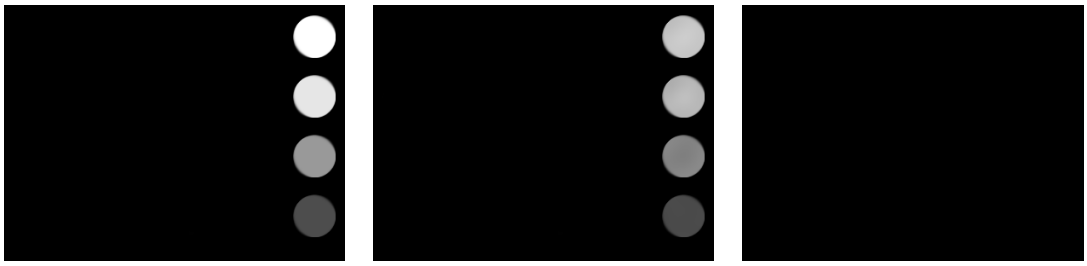
Therefore, we have a classic supervised machine learning problem with a training set



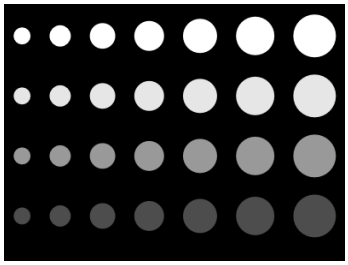
(a) $\lambda = 5$ (b) $\lambda = 6$ (c) $\lambda = 7$



(d) $\lambda = 8$ (e) $\lambda = 9$ (f) $\lambda = 10$

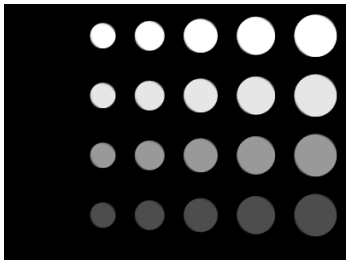


(g) $\lambda = 11$ (h) $\lambda = 12$ (i) $\lambda = 13$

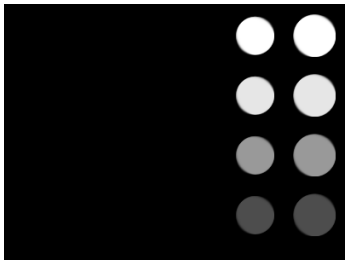


(j) Input image

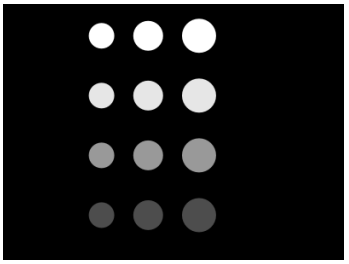
Figure 3.4: An example for the scale space generated by the TV-L^1 model. The object size is directly related to the parameter λ. If the equivalence relation $\lambda = \frac{2}{r}$ is fulfilled, the solution is not unique. We see this effect observing the most left circles for $\lambda \in \{6, 12\}$. If λ is greater than $\frac{2}{r}$ the object suddenly merges with the background.



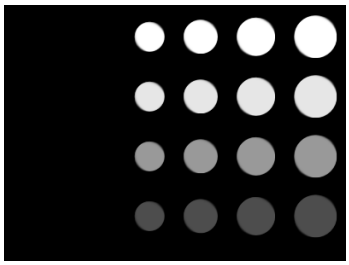
(a) v_1 with $\lambda_1 = 7$



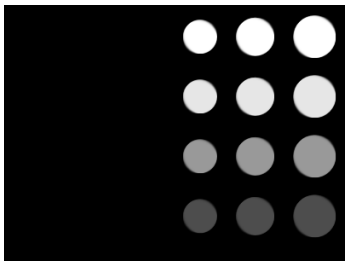
(b) v_2 with $\lambda_2 = 10$



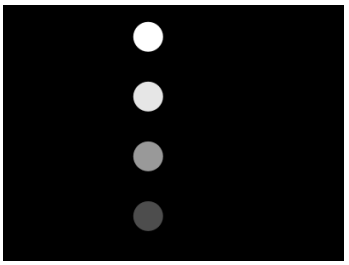
(c) $u = v_1 - v_2$



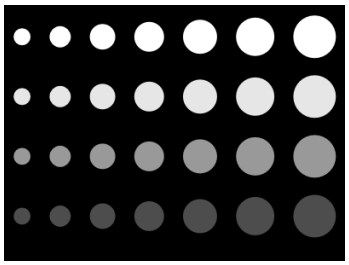
(d) v_1 with $\lambda_1 = 8$



(e) v_2 with $\lambda_2 = 9$

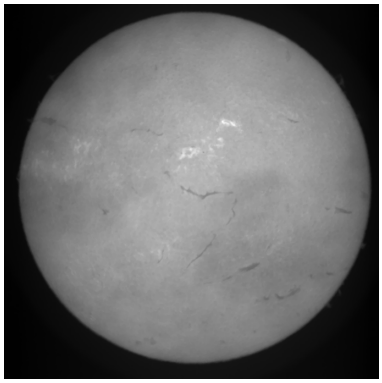


(f) $u = v_1 - v_2$

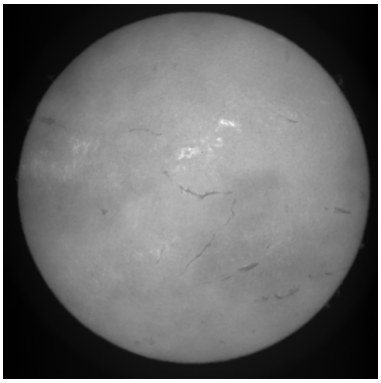


(g) Input image

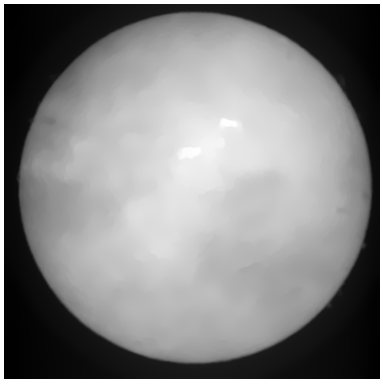
Figure 3.5: This figures demonstrates two example results of the structural bandpass filter for the input image (g). The two left images (a) and (d) show that smaller structures are removed by a smaller λ_1. If we increase the parameter λ only larger shapes remain, see image (b) and (e). Finally, the result of the structural bandpass is given by subtracting the second image from the first one and only the circles in the middle remain. See image (c) and (f).



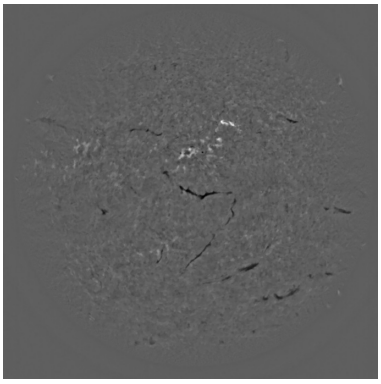
(a) f



(b) v_1



(c) v_2



(d) u

Figure 3.6: The depicted images show the result of the structural bandpass filter applied to a cloudy Hα image. Image (b) illustrates the denoised version v_1 of f. Note the influences of clouds on the image appearing as darker areas on the left and right. These large-scale intensity effects are captured in v_2, but finer structures like filaments are not captured by this filter. The result u of the structural bandpass filter is shown in figure (d).

$\mathcal{D} = \{(\boldsymbol{\phi}_i, y_i)\}_{i=1}^{N}, y_i \in [1, C] \subset \mathbb{N}$. For the classes we will use obviously *flare* and *filament*, but also *sun spot*, because sun spots have a darker appearance in Hα images than filaments and the detection results will come handy in further postprocessing steps. The rest of the image is summarized in the class *background*.

3.4.1 Features

3.4.1.1 Intensity

The most intuitive choice for the classification task is the intensity associated with each pixel. In the first chapter we have already mentioned that filaments are darker than the background. In Hα images Sun spots appear as even darker, but smaller and round objects, whereas flares are defined as drastic brighter objects.

Figure 3.7 illustrates the class probabilities $p(y|\boldsymbol{\phi})$ in a histogram. The data we use for the feature selection and the learning of the model in the next step are derived from labeled Hα images, where an expert annotated various different images by assigning the pixels to the differing classes.

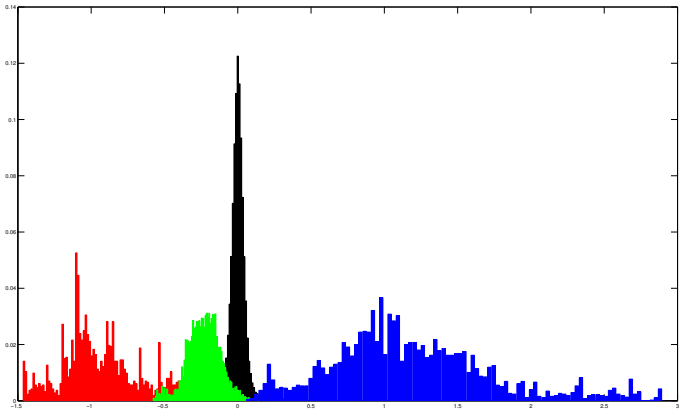


Figure 3.7: Intensity distribution of the classes *sun spot* (red), *filament* (green), *flare* (blue) and *background* (black). The training examples are derived from preprocessed Hα images that were annotated by an expert. Especially the overlap between the class *filament* and *background* is problematic, because it causes many false classifications.

As one can observe, the probability distributions of the four classes cannot be separated. The overlaps between the classes *sun spot* - *filament* and *background* - *flare* are no real problem, because most of the probability mass is good separable. In contrast, the probability distribution overlap between the class *filament* and *background* causes segmentation problems in the application. Therefore, we try to find a feature that especially emphasizes filaments and discriminates them from the background.

λ_1	λ_2	**Shape**
$\gg 0$	$\gg 0$	a dark, blob-like structure
$\ll 0$	$\ll 0$	a bright, blob-like structure
$\gg 0$	≈ 0	a dark, elongated structure
$\ll 0$	≈ 0	a bright, elongated structure
≈ 0	≈ 0	noisy

Table 3.1: Shapes described by the eigenvalues of the Hessian matrix. The eigenvalues are sorted as $|\lambda_1| \geq |\lambda_2|$.

3.4.1.2 Filament Filter

How is a filament characterized despite by its intensity in an Hα image? In the introduction, we already presented that filaments are elongated structures. This implies that a filter, which enhances elongated structures should provide additional discriminative information. Frangi et al. [42] presented a vessel enhancement filter by exploiting the curvature information of the Hessian matrix. We use the same ideas as Frangi et al. for our filament filter.

To describe the local behavior of a function $f(\boldsymbol{x})$, it is common to analyze the Taylor expansion in the neighborhood of a point $\boldsymbol{x}_0$

$$f(\boldsymbol{x}_0 + \mathrm{d}\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + \mathrm{d}\boldsymbol{x}^T \nabla f(\boldsymbol{x}_0) + \mathrm{d}\boldsymbol{x}^T \nabla^2 f(\boldsymbol{x}_0)\, \mathrm{d}\boldsymbol{x} \tag{3.16}$$

$$\nabla f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} \tag{3.17}$$

$$\nabla^2 f = \nabla^T \nabla f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix} \tag{3.18}$$

where $\nabla f(\boldsymbol{x})$ denotes to the gradient vector and $\nabla^2 f(\boldsymbol{x})$ to the Hessian matrix of the image f, respectively. As Frangi et al. [42] describe in their paper, the eigenvalues $|\lambda_1| \geq |\lambda_2|$ and eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ of the Hessian matrix describe an ellipse in the two dimensional case and can be understood as a dual to the actual shape. Depending on the eigenvalues we can distinguish between different shapes, which are summarized in table 3.1.

The response function $R(\boldsymbol{x})$ for elongated filament structures can now be designed as follows

$$R(\boldsymbol{x}) = \begin{cases} \lambda_1 - \lambda_2 & \text{if } \lambda_1 > 0 \\ 0 & \text{else} \end{cases} \tag{3.19}$$

This function ensures that we only gain response for dark structures and further emphasizes elongated shapes in a linear manner. The remaining problem with this response function is the scale. Depending on the length, or size of the filaments the response function reaches its maximum at different scales. Therefore, we define our final response function over the maximum in the scale space

$$R = \max_{\sigma_{\min} \leq \sigma \leq \sigma_{\max}} \sigma R(\boldsymbol{x}_\sigma) \tag{3.20}$$

where $\boldsymbol{x}_\sigma$ is the pixel $\boldsymbol{x}$ at scale σ and $\sigma_{\min}, \sigma_{\max}$ define the scale interval. The multiplication of the response function $R(\boldsymbol{x}_\sigma)$ with σ ensures the properties of the scale space theory [57]. The scale space itself for the image $f(\boldsymbol{x})$ with $\boldsymbol{x} = (x_1, x_2)^T$ is given as follows

$$f(\boldsymbol{x}_\sigma) = G(\boldsymbol{x}, \sigma) * f(\boldsymbol{x}) \tag{3.21}$$

$$G(\boldsymbol{x}, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x_1^2 + x_2^2}{2\sigma^2}} \tag{3.22}$$

An example result of this filter is illustrated in figure 3.8. The true positive results for the filaments are pretty good, but the filter also yields a lot of false positive response in the background, namely near plagues and around sun spots.

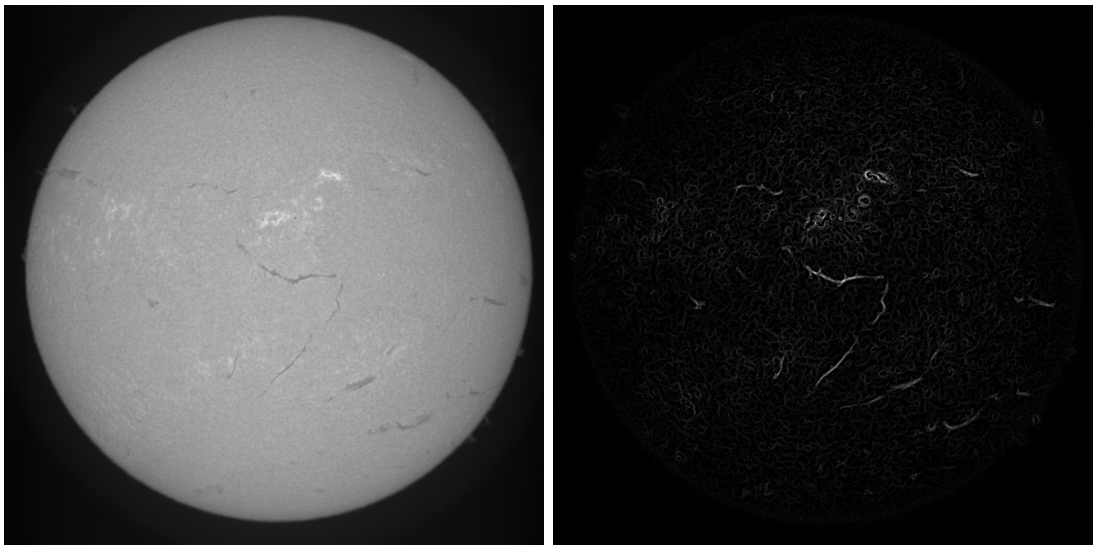


Figure 3.8: Multi-scale response of the filament filter (right) for the given input image (left). The response for filaments is pretty good, but we get also a lot of response in and near plagues and around sun spots.

3.4.1.3 Distance From Disk Center

Although the structural bandpass filter captures the limb darkening, the contrast of the solar features decreases towards the limb and the recognition, especially of filaments, gets worse closer to the limb. Therefore, we propose an additional feature, the Euclidean distance from the center of the sun to the pixel to compensate this effect. Figure 3.9 visualizes the effect by plotting a random subset of the training data in a scatter plot.

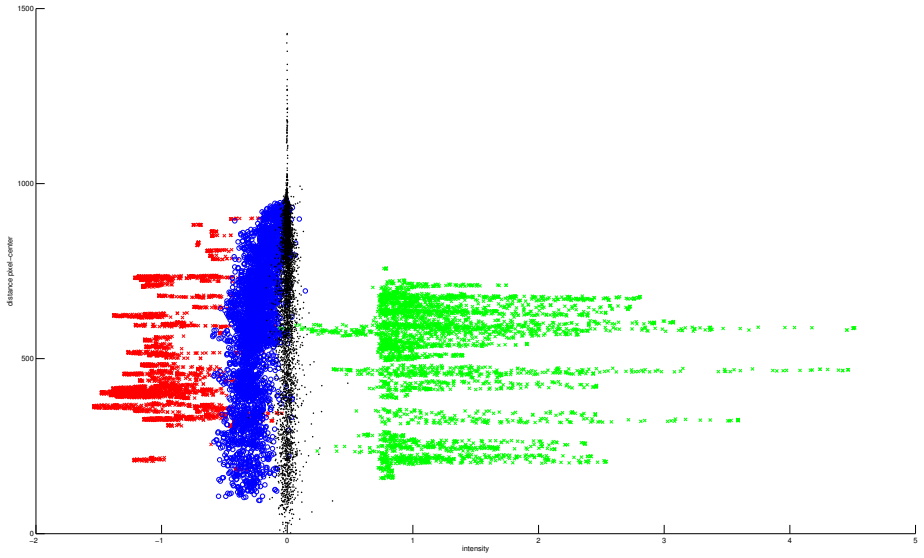


Figure 3.9: Scatter plot of a random subset of the training samples with the features intensity on the x- and distance on the y-axis. Red represents the class *sun spot*, blue *filament*, black *background* and green *flare*, respectively. One can observe that the classes *filament* and *background* discriminate better near the solar center and due to the contrast loss worse closer to the limb.

3.4.2 Model Learning

If we use the intensity and the response of the filament filter to build our feature vectors, we have $\boldsymbol{\phi} \in \mathbb{R}^2$. We can further incorporate information about the local neighborhood by using the $m \times m$ intensity and filament filter values of the neighboring pixels. This would lead to a feature vector $\boldsymbol{\phi} \in \mathbb{R}^{m^2}$, if we only use the intensity values, and $\boldsymbol{\phi} \in \mathbb{R}^{2 \cdot m^2}$, if we utilize both features. As one can observe for example in figure 3.6(d), the contrast of the filaments decreases from the center towards the limb. To incorporate this fact, it turned

out that the distance from the center of the solar disk to the pixel location is another beneficial feature. So this would lead to a feature vector $\boldsymbol{\phi} \in \mathbb{R}^{2 \cdot m^2+1}$.

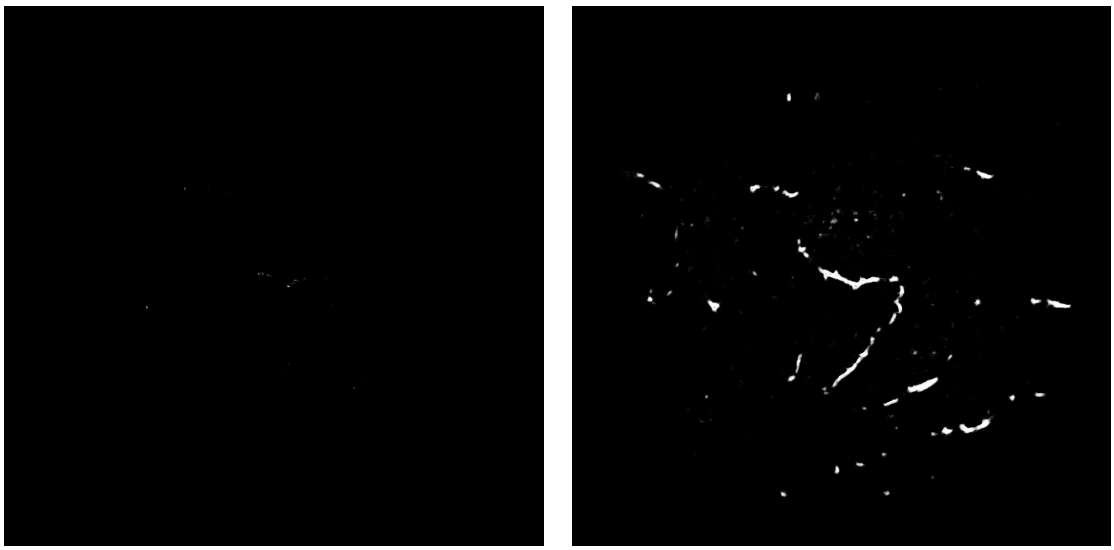
The next step is to learn a classifier model. In chapter 2 we have shown three different machine learning techniques, namely Gaussian mixture models, support vector machines and random forests. We trained all three models with different feature combinations. For the training set, we randomly selected 3000 feature vectors for the class *sunspot*, 20000 for the class *filament*, 6000 for the class *flare* and 36000 for the class *background* from the annotated image data. For the evaluation, we created an independent test set of 1800 feature vectors for the class *sunspot*, 8000 for the class *filament*, 3000 for the class *flare* and 25200 for the class *background*. The feature vectors were build by using an $m \times m$ patch with $m \in \{1, 3, 5, 7\}$ once for the intensity values and the center-pixel distance and once with the additional filament filter response.

For the Gaussian mixture models we tried different numbers of components and performed three runs from random initializations, because the expectation-maximization algorithm is not guaranteed to converge in a global optimal solution. The support vector machine was used with a linear kernel and we selected the parameter C with cross validation on the training set. For the last machine learning method, the random forest, we tested different numbers of trees. The results of our evaluation are summarized in table 3.2.

From the comparison, it is evident that the random forest outperforms the other methods on all feature combinations and overall is by 1.4886% better than the GMM and by 7.6341% better than the SVM. Further, already a few trees are sufficient for very good results and the accuracy only slightly increases, if we add further trees to the model. However, if we just consider the intensity of the actual pixel and the center-pixel distance, we observe that the results of the RF and the GMM are marginal. Further, a smaller feature vector allows a faster classification and therefore we decided to use the GMM with just those two features for our method. Figure 3.10 shows an example of applying the learned model to an Hα image.

3.5 Multi-Label Segmentation

So far we have assigned each pixel $\boldsymbol{x}$ to a feature vector $\boldsymbol{\phi}$, and based on the feature vectors computed a pixelwise probability $p(\cdot|\boldsymbol{\phi}(\boldsymbol{x}))$. However, the result is not smooth as it is evident in figure 3.10 for example. Therefore, we regularize the result and utilize the



(a) $p(\textit{sun spot}|\boldsymbol{\phi}(\boldsymbol{x}))$ (b) $p(\textit{filament}|\boldsymbol{\phi}(\boldsymbol{x}))$

(c) $p(\textit{flare}|\boldsymbol{\phi}(\boldsymbol{x}))$ (d) $p(\textit{background}|\boldsymbol{\phi}(\boldsymbol{x}))$

Figure 3.10: The images demonstrate the result of applying the learned Gaussian mixture model to a preprocessed Hα image. The color white indicates a high probability and the black one vice versa. Although the result looks reasonable good, it is not really smooth, and we regularize it with an variational method in the next step.

	1×1		3×3		5×5		7×7	
	I	I + F	I	I + F	I	I + F	I	I + F
GMM (1)	0.9607	0.9609	0.9427	0.9326	0.9292	0.9159	0.9175	0.9019
GMM (2)	0.9666	0.9658	0.9624	0.9614	0.9534	0.9445	0.9447	0.9357
GMM (3)	0.9680	0.9674	0.9636	0.9642	0.9559	0.9486	0.9492	0.9419
GMM (4)	0.9688	0.9673	0.9635	0.9658	0.9580	0.9531	0.9515	0.9487
GMM (5)	0.9691	0.9680	0.9639	0.9654	0.9599	0.9556	0.9548	0.9509
SVM	79.4789	79.5105	85.8132	88.9605	89.9500	90.4605	90.7605	90.5553
RF (15)	0.9726	0.9727	0.9764	0.9758	0.9813	0.9806	0.9834	0.9833
RF (30)	0.9731	0.9731	0.9775	0.9759	0.9815	0.9809	0.9836	0.9838
RF (45)	0.9736	0.9734	0.9783	0.9763	0.9816	0.9813	0.9837	0.9838
RF (60)	0.9737	0.9735	0.9781	0.9763	0.9814	0.9815	0.9834	0.9839

Table 3.2: Evaluation results of the machine learning approaches. *GMM* stands for Gaussian mixture model and the integer in the brackets indicates the number of components per class. The support vector machine is abbreviated by *SVM* and the random forest by *RF*. The integer in the bracket beside *RF* shows the number of forests used in the model. For the comparison, we use the accuracy defined as $\frac{\#\text{correct classified}}{\#\text{all featuer vectors}}$ on a separate test set with 38000 feature vectors. The header indicates the size of the pixel neighborhood and the set of features, I stands for intensity and F stands for filament filter, to form with the center-pixel distance the feature vectors.

variational approach we have presented in chapter 2

$$\begin{aligned} \min_{\{u\}_{l=1}^{N}} \sum_{l=1}^{N} \int_{\Omega} \mathrm{d}|\nabla u| + \sum_{l=1}^{N} \int_{\Omega} u_l q_l \,\mathrm{d}x \\ \text{s.t. } u_l(x) \geq 0, \quad \sum_{l=1}^{N} u_l(x) = 1 \end{aligned} \tag{3.23}$$

Thus, we have to define the weighting functions q_l to fit the above formulation and obtain the segmentation results u_l. One solution is taking the negative logarithm of the probabilities by setting $q_l(\boldsymbol{x}) = -\log p(class_l|\boldsymbol{\phi}(\boldsymbol{x}))$. While this is a valid solution, it does not incorporate any information of the previous images in the sequence. To obtain a temporal smoothness, we apply a moving average filter on the probabilities.

Assume $p_t(\cdot|\boldsymbol{\phi}(\boldsymbol{x}))$ is the probability of the input image at time step t, then we define the simple moving average as the mean over the n last probabilities as

$$p_{\mathrm{SMA},t}(\cdot|\boldsymbol{\phi}(\boldsymbol{x})) = \frac{\sum_{i=1}^{n} p_{t-i+1}(\cdot|\boldsymbol{\phi}(\boldsymbol{x}))}{n} \tag{3.24}$$

It is possible to avoid the summation of all summands in each iteration and increase

the performance by using the following formulation

$$p_{\mathrm{SMA},t}(\cdot|\boldsymbol{\phi}(\boldsymbol{x})) = p_{\mathrm{SMA},t-1}(\cdot|\boldsymbol{\phi}(\boldsymbol{x})) - \frac{p_{t-n}(\cdot|\boldsymbol{\phi}(\boldsymbol{x}))}{n} + \frac{p_t(\cdot|\boldsymbol{\phi}(\boldsymbol{x}))}{n} \tag{3.25}$$

The above equation weights the current probabilities equal to the probability values obtained from images in the far past, but this is actually not desirable. This could delay the recognition of sudden events like flares. Therefore, we use an exponential weighted moving average. The successive computation is given by

$$p_{\mathrm{EMA},t}(\cdot|\boldsymbol{\phi}(\boldsymbol{x})) = wp_t(\cdot|\boldsymbol{\phi}(\boldsymbol{x})) + (1-w)p_{\mathrm{EMA},t-1}(\cdot|\boldsymbol{\phi}(\boldsymbol{x})) \tag{3.26}$$

where $w \in [0,1]$ is an adjustable weighting factor and influences how fast older probabilities are discounted. This filter has the additional advantage that only the last moving average result is needed to compute the new one and is therefore more efficient in terms of memory and speed.

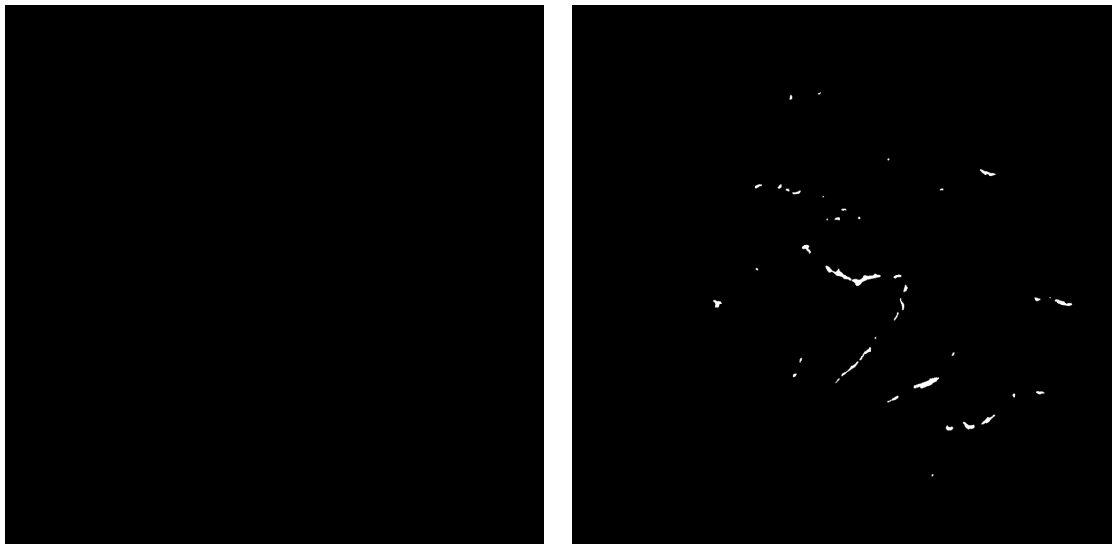
For our segmentation model, we use for the weighting function $q_i(\boldsymbol{x}) = -\log p_{\mathrm{EMA},t}(class_i|\boldsymbol{\phi}(\boldsymbol{x}))$. The segmentation results for the probabilities shown in figure 3.10 are illustrated in figure 3.11.

3.6 Postprocessing

The final step of the method is the postprocessing that contains three different goals. The first one is the identification of filaments and flares. We have given four binary images from the segmentation and want to assign a unique ID to the single filaments and flares. In addition, the ID should stay the same over the image sequence for the very same object. Therefore we apply a simple tracking scheme. The second goal is the refinement of the results, especially for the filaments. This includes the connection of interrupted filament components and flare ribbons, but also the removal of too small recognitions. The last goal is the derivation of properties from the identified objects to categorize them.

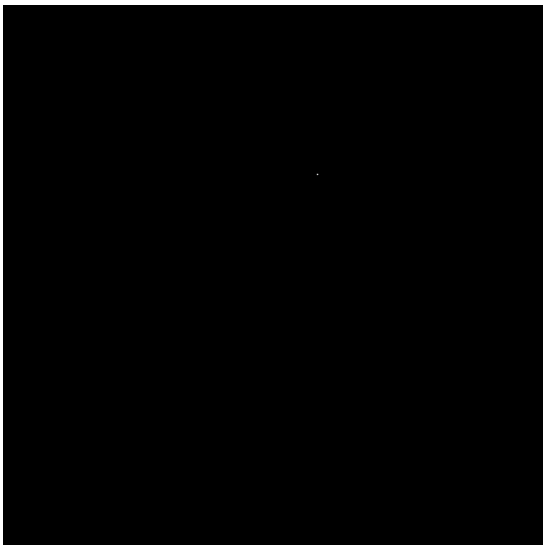
3.6.1 General

The identification and tracking task can be solved equally for filaments and flares. The first task is well-known as connected-component labeling problem and can be efficiently solved. For the second task, the tracking of the objects in the Hα image sequence, we present a simple propagation technique.



(a) *sun spot*

(b) *filament*



(c) *flare*

(d) *background*

Figure 3.11: The segmentation result for the probabilities of figure 3.10 and $\lambda = 1$.

1	1	1	0	0	1	0	1
1	1	1	0	0	1	0	1
0	1	0	0	0	1	0	1
1	1	1	0	0	0	0	1
1	1	1	0	1	0	0	1
0	0	0	0	1	1	0	1
0	0	0	1	1	1	0	1

(a) Binary image s

1	1	1	0	0	2	0	3
1	1	1	0	0	2	0	3
0	1	0	0	0	2	0	3
1	1	1	0	0	0	0	3
1	1	1	0	4	0	0	3
0	0	0	0	4	4	0	3
0	0	0	4	4	4	0	3

(b) Connected component labeling result

Table 3.3: The left table illustrates a sample binary image s with four blobs, whereas the connected components are labeled in the right table.

3.6.1.1 Connected Component Labeling

From the segmentation we obtain four binary images $s_i(\boldsymbol{x}) \rightarrow \{0, 1\}$. The next step is to identify 8-connected pixels that form a group and are separated through zeros from other groups and assign an ID to them. An example is illustrated in table 3.3. In the following, we will use the terms group, part of a filament, blob and component interchangeable.

The problem can efficiently be solved with a two-pass algorithm as presented for example in [78]. In a first pass, temporary labels get assigned and the label equivalences get stored in a union-find data structure. We have a label equivalence, if two temporary labels are neighbors. In the second pass, the temporary labels get replaced by the actual labels that are given by the root of the equivalence class. The union-find data structure is a collection of disjoint sets and has, as its name implies, two important functions. The *union* function combines two sets, for instance, $union(\{1, 3, 5\}, \{2\}) = \{1, 2, 3, 5\}$. The *find* function returns the set that contains a given number. We can implement the data structure efficiently with trees. The final algorithm is listed in algorithm 7

3.6.1.2 Tracking

The connected component labeling ensures that every filament and flare part has a unique ID per image. Nevertheless, the ID is not guaranteed to stay the same through the image sequence. For this tracking purpose we propagate the ID of previous images.

Assume that $l_t(\boldsymbol{x})$ is the current component labeled segmentation and $\{l_{t-k}(\boldsymbol{x})\}_{k=1}^{n}$ the set of n previous component labeled segmentation results. Then we change the ID of a current component i_t to the ID j, where j is given by the components of the previous

Algorithm 7 Connected components labeling for a binary input image $s \in \mathbb{N}^{m \times n}$. The result is stored in the image $l \in \mathbb{N}^{m \times n}$.

```
initialize union-find data structure e
temporaryLabel ← 2
{First pass}
for x1 ← 1...n do
  for x2 ← 1...m do
    if s(x1, x2) ≠ 0 then
      neighbors ← set of 8-connected neighbors
      if max(neighbors) = 0 then
        l(x1, x2) ← temporaryLabel
        addSet(e, {temporaryLabel})
        temporaryLabel ← temporaryLabel + 1
      else
        label ← min(neighbors)
        l(x1, x2) ← label
        for n in neighbors do
          union(e, find(n), find(label))
        end for
      end if
    end if
  end for
end for
{Second pass}
for x1 ← 1...n do
  for x2 ← 1...m do
    if s(x1, x2) ≠ 0 then
      l(x1, x2) ← min(find(l(x1, x2)))
    end if
  end for
end for
```

images that have the most overlap with the component i_t.

This can be implemented in a pixelwise fashion and a simple map data structure. For a given component of the current image and ID i_t, we iterate all overlapping pixels $\boldsymbol{x}$ of the set $\{l_{t-k}(\boldsymbol{x})\}_{k=1}^{n}$. If $l_{t-k}(\boldsymbol{x}) \neq 0$ we increment the counter for the ID $l_{t-k}(\boldsymbol{x})$ in the map. Finally we assign the ID with the highest counter.

3.6.2 Filament

The detection of filaments is more difficult due to the variability of their appearances. The elongated structure can have almost arbitrary shape and can be interrupted. In addition,

the same pattern we have trained our model for the segmentation appear around sun spots, near flares and in active regions. In this mentioned cases, the detection should be suppressed, but also if the length of a filament does not reach a certain size of interest.

For the first problem, the false detections near flares and sun spots, we apply a simple yet sufficient solution. We have the segmentation results of the class *flare* s_{flare} and *sun spot* $s_{sunspot}$ and set all pixels in the segmentation result of the *filament* $s_{filament}$ to 0, if they are near a certain distance t_d to the above mentioned objects. This successfully suppresses the detection results in these regions.

3.6.2.1 Interrupted Filaments

The reconnection of an interrupted filament is more challenging. Before presenting our approach for the filament reconnection, we introduce some definitions. Assume we have given the component labeling result $l(\boldsymbol{x})$ for the filaments. It is possible that this result contains two parts with different IDs for a single filament. This happens, if the detection was not successful due to the appearance in the Hα image. If we assign each part to the same ID, we have reconnected them.

To reconnect two parts, we need a measure that defines when two parts belong together. Our method relies on the distance of two parts. If we consider a part as a set of Cartesian coordinates $\boldsymbol{x}$, then the distance between two parts A, B is given by the set distance

$$\operatorname{dist}(A, B) = \inf_{a \in A, b \in B} \operatorname{dist}(a, b) \tag{3.27}$$

where the distance between two points a, b is given by the Euclidean distance. We can implement the set distance in a way that we can also derive the points a, b, which minimizes the distance.

A first approach is to reconnect two parts if the distance between them is under a certain fixed threshold. However, this approach does not incorporate any information of the filament structure, see figure 3.12 for example. Therefore, we present a technique based on the principal directions of the single filament parts.

As it is known from the principal component analysis (PCA) [66], the eigenvector $\boldsymbol{v}$ associated with the largest eigenvalue λ of the covariance matrix Σ points in the direction of the greatest variance and λ indicates the amount of variance. If a part consists of n pixels at the locations $\{\boldsymbol{x}_i\}_{i=1}^{n}$, then the principal direction is computed as follows

(a) Two parts of one filament (b) Three parts of three different filaments

Figure 3.12: The images show different labeled filament parts with their principal directions depicted as red line. The image on the right illustrates the segmentation result of one interrupted filament, but the angle between the principal directions is nearly zero. In contrast, the parts visualized in the right image belong to different filaments and also the corresponding principal directions diverge.

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T = \begin{pmatrix} \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_3 \end{pmatrix} \tag{3.28}$$

$$\lambda = \frac{\mathrm{tr}(\Sigma)}{2} + \sqrt{\frac{\mathrm{tr}(\Sigma)^2}{4} - \det(\Sigma)}; \tag{3.29}$$

$$\boldsymbol{v} = \begin{cases} (\lambda - \sigma_3, \sigma_2)^T & \text{if } \sigma_2 \neq 0 \\ (1,0)^T & \text{else} \end{cases} \tag{3.30}$$

where $\boldsymbol{\mu}$ is the sample mean. Using the geometric definition of the dot product $\boldsymbol{v}_1^T\boldsymbol{v}_2 = \|\boldsymbol{v}_1\|\,\|\boldsymbol{v}_2\|\cos\phi$, we can define an adaptive distance that considers the angle between the principal directions and makes the filament part reconnection adaptive to their structures. The method connects two parts with principal directions v_1 and v_2, if the distance between

the two parts is under a certain adaptive threshold t_{ad}. The threshold itself is given by

$$dist_{\text{adap}} = 7 \cdot \log(\lambda_1) \cos\phi \tag{3.31}$$

$$t_{\text{ad}} = \begin{cases} dist_{\text{min}} & \text{if } dist_{\text{adap}} \leq dist_{\text{min}} \\ dist_{\text{adap}} & \text{if } dist_{\text{min}} < dist_{\text{adap}} < dist_{\text{max}} \\ dist_{\text{max}} & \text{if } dist_{\text{max}} \leq dist_{\text{adap}} \end{cases} \tag{3.32}$$

So far we assign the same ID to two reconnected parts, but for the final step, the computation of the filament length, it is beneficial, if the two parts are connected via a line. Therefore, we utilize the Bresenham algorithm [22] for line drawing as depicted in algorithm 8 to set all pixels on the shortest line between the two parts to the ID of the filament.

Algorithm 8 Bresenham algorithm for rasterized line drawing from (x_1, y_1) to (x_2, y_2) on the image $l(\boldsymbol{x})$ [22].

$diff_x \leftarrow |x_2 - x_1|$
$diff_y \leftarrow -|y_2 - y_1|$
$step_x \leftarrow \begin{cases} 1 & \text{if } x_1 < x_2 \\ -1 & \text{else} \end{cases}$
$step_y \leftarrow \begin{cases} 1 & \text{if } y_1 < y_2 \\ -1 & \text{else} \end{cases}$
$error \leftarrow diff_x + diff_y$
$l(x_1, y_1) = ID$
while $x_1 \neq x_2 \vee y_1 \neq y_2$ **do**
  if $2 \cdot error > diff_y$ **then**
    $error \leftarrow error + diff_y$
    $x_1 \leftarrow x_1 + step_x$
  end if
  if $2 \cdot error < diff_x$ **then**
    $error \leftarrow error + diff_x$
    $y_1 \leftarrow y_1 + step_y$
  end if
  $l(x_1, y_1) = ID$
end while

3.6.2.2 Property Derivation

Finally we need to derive some properties from the filaments' detections. The most interesting characteristic of filaments for solar physicists is the length. From the segmentation

result, as a set of pixels, the derivation of the length is not trivial. Therefore, we first reduce this set to a tree structure by applying a thinning algorithm.

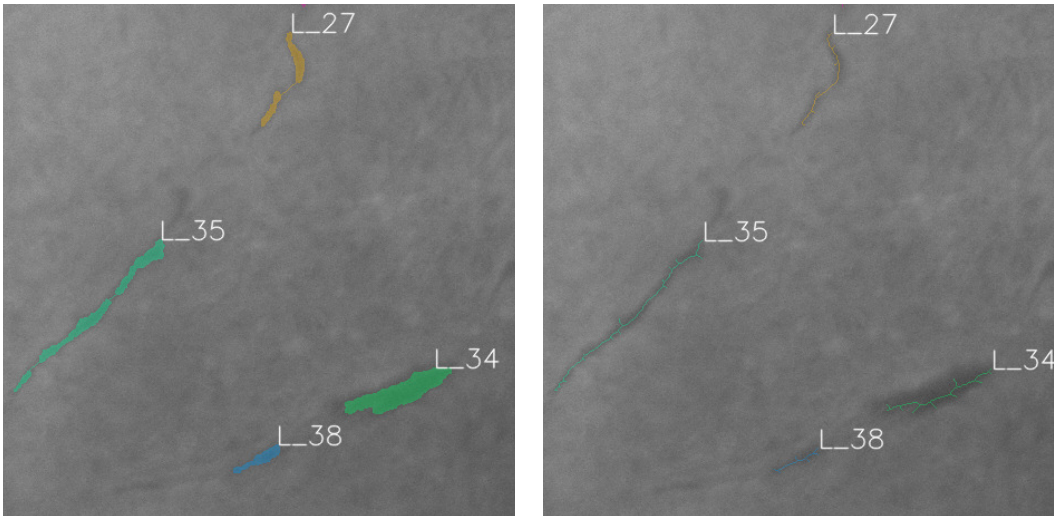
We utilize the pixelwise thinning algorithm by Guo and Hall [48], which is easy to parallelize. The algorithm first divides the image into two distinct subfields in a chessboard manner. On each subfield a pixel is deleted, if and only, if three conditions are fulfilled and after each iteration the subfields alternate. The first condition g_1 preserves the connectivity of the skeleton. Condition g_2 enforces that only boundary pixels are deleted and finally condition g_3 makes sure that no endpoints are deleted. The algorithm is summarized in algorithm 9.

Algorithm 9 Thinning algorithm for a binary image $l(\boldsymbol{p}) \in \mathbb{Z}^{m \times n}$ by [48].

Define the neighborhood of a pixel $\boldsymbol{p}$ as $\begin{bmatrix} \boldsymbol{x}_4 & \boldsymbol{x}_3 & \boldsymbol{x}_2 \\ \boldsymbol{x}_5 & \boldsymbol{p} & \boldsymbol{x}_1 \\ \boldsymbol{x}_6 & \boldsymbol{x}_7 & \boldsymbol{x}_8 \end{bmatrix}$
$state \leftarrow 0$
$switch \leftarrow 0$
repeat
  for $p_1 \leftarrow 1 \dots m$ **do**
    for $p_2 \leftarrow 1 \dots m$ **do**
      $\boldsymbol{p} = (p_1, p_2)^T$
      $g_1 \equiv \sum_{i=1}^{4} \begin{cases} 1 & \text{if } \neg l(\boldsymbol{x}_{2i-1}) \wedge (l(\boldsymbol{x}_{2i}) \vee l(\boldsymbol{x}_{2i+1})) \\ 0 & \text{else} \end{cases}$
      $g_2 \equiv \min\left(\sum_{i=1}^{4} l(\boldsymbol{x}_{2i-1}) \vee l(\boldsymbol{x}_{2i}), \sum_{i=1}^{4} l(\boldsymbol{x}_{2i}) \vee l(\boldsymbol{x}_{2i+1})\right) \in \{2, 3\}$
      $g_{3,1} \equiv (l(\boldsymbol{x}_2) \vee l(\boldsymbol{x}_3) \vee \neg l(\boldsymbol{x}_8)) \wedge l(\boldsymbol{x}_1) = 0$
      $g_{3,2} \equiv (l(\boldsymbol{x}_6) \vee l(\boldsymbol{x}_7) \vee \neg l(\boldsymbol{x}_4)) \wedge l(\boldsymbol{x}_5) = 0$
      $pixelstate \leftarrow (p_1 \% 2 + p_2 + state) \% 2$
      if $pixelstate = state = switch \wedge g_1 \wedge g_2 \wedge g_{3,pixelstate+1}$ **then**
        $l(\boldsymbol{p}) \leftarrow 0$
      end if
    end for
  end for
  $state \leftarrow (state + 1) \% 2$
  if $state = 0$ **then**
    $switch \leftarrow (switch + 1) \% 2$
  end if
until no pixel was deleted from M

An example of the thinning result is illustrated in figure 3.13. The resulting skeleton can be viewed as a graph, more specific as a tree. From this structure, we can derive the length of the filament, but the problem of finding the longest path in a general graph is known to be NP-complete. However, on a tree we can utilize a slightly modified Floyd-Warshall algorithm [41], which was originally designed for shortest path computation,

to efficiently compute the longest path. To apply the algorithm, we build an adjacency matrix A to represent the tree, where each edge gets assigned the weight -1 if two pixels are 8-connected. On this matrix, we can apply the shortest path algorithm as listed in algorithm 10. The negative result is a good approximation of the filament length.



(a) The colored filament segmentation are the input for the thinning algorithm.

(b) The resulting tree structures.

Figure 3.13: On the left side, there is a trimmed Hα image with filament detections highlighted as colored blobs. The image on the right side shows the thinning results. The colored blobs are reduced to trees that can be used for length computation.

Algorithm 10 Floyd-Warshall [41] algorithm for shortest path. The input $A \in \mathbb{Z}^{m \times m}$ is the adjacency matrix of the tree.

for $k \leftarrow 1 \dots m$ **do**
  for $i \leftarrow 1 \dots m$ **do**
    for $j \leftarrow 1 \dots m$ **do**
      if $A(i,k) + A(k,j) < dist(i,j)$ **then**
        $A(i,j) \leftarrow A(i,k) + A(k,j)$
      end if
    end for
  end for
end for
$shortestPath \leftarrow \min(A)$

3.6.3 Flare

The postprocessing for the flare detections is less comprehensive. The only thing we perform is the grouping of flare detections in terms of assigning the same ID. In contrast to the filament detections, the grouping of single flare parts is not performed because they are interrupted, but rather because a single flare can occur in two or more ribbons. Therefore, we group flare detections that are within a certain distance t_d.

3.6.3.1 Property Derivation

The derivation of the flare properties also is simpler than for the filaments. The first property is the flare area and simply is given by the number of segmented pixels with the same ID. For the categorization, some intensity values relative to the background are needed. Therefore, we compute the mean, standard deviation, maximum and minimum of the pixel intensities within the segmented regions.

Chapter 4

Implementation

Contents

In this chapter we present details on the concrete implementation of our method. The first section is devoted to the FITS format as the provided images have this image format, which is very common in the astronomic field. In section 4.2, we present the modified tool we used to gather the annotated data for the feature selection and the model learning. The last section is dedicated to the fast implementation of the algorithms on a graphic processing unit.

4.1 FITS Format

The Flexible Image Transport System (*FITS*) is the quasi standard for image files in the field of astronomy, and the input files to our method are of this format, too. It originally was defined in 1981 [86] and it's current version is 3.0 [67]. The file format allows the simultaneous storage of image metadata in ASCII tables along with the raw image data in 64 bit precision, whereas the image can be two, or three dimensional.

There also exists an official and high-level I/O library, called CFITSIO* that encapsulates the access to the image format. The library has the advantage that it is heavily

*`http://heasarc.gsfc.nasa.gov/fitsio/`, last checked 2013-04-02

tested and under continuous maintenance by William D. Pence.

We use the above mentioned library in our implementation to read the imaged data from the input files. All functions and data types are defined in the `fitsio.h` header file, which we have to include. The structure `fitsfile` stores all the relevant parameters that defines the format of a particular FITS file. With the function `fits_open_file`, the file can be opened and with `fits_get_img_param`, the most important parameters, like the image dimensions and bits per pixel, are retrieved. Finally, the function fits_read_pix is used to read the image content row by row. A complete function written in C++ is presented in listing 4.1.

4.2 Image Annotation

For feature selection and model learning as described in section 3.4, we need annotated data. In this special case we need the pixelwise annotation of Hα images, therefore each pixel of some training images should be associated with one of the four mentioned classes.

To create the annotated data, we modified the *Image annotation tool*[†] by Alexander Kläser in several places. The tool itself allows the pixelwise annotation of images for an arbitrary number of classes. For each class, it stores an associated image file containing the annotation.

The first modification is related to the image I/O. The original program is only able to read from JPEG and PNG files, but not from FITS files. Therefore, we included the FITS input as described in the previous section. Another modification concerns the possible classes, which we limited to the classes *sun spot*, *filament*, *flare* and *filament*.

To generate training data for especially difficult cases, we implemented the functionality to segment the image with our proposed method. This segmentation result can then be used as an annotation input and therefore provides a fast way to generate new training data.

A screenshot of the tool is depicted in figure 4.1.

4.3 GPU Implementation

Most of the iterative algorithms presented in chapter 3 are computationally expensive. However, they can be parallelized easily and therefore are able to utilize the computational power of modern graphic processing units (*GPU*).

[†] `http://lear.inrialpes.fr/people/klaeser/software_image_annotation`, last checked 2013-04-02

```
#include "fitsio.h"

double* readfits(std::string& file) {
    fitsfile* fptr;
    int bitpix, naxis, status = 0;
    long dimensions[2] = { 1, 1 };
    long fpixel[2] = { 1, 1 };
    double* imgData;

    if (!fits_open_file(&fptr, file.c_str(), READONLY, &status)) {
        if (!fits_get_img_param(fptr, 2, &bitpix, &dimensions, naxes, &
            status)) {
            imgData = new double[dimensions[0] * dimensions[1]];
            double* pixels = new double[dimensions[0]];

            for (fpixel[1] = dimensions[1]; fpixel[1] >= 1; fpixel[1]--) {
                if (fits_read_pix(fptr, TDOUBLE, fpixel, dimensions[0],
                    NULL, pixels, NULL, &status))
                    break;

                for (int ii = 0; ii < dimensions[0]; ii++)
                    imgData[(dimensions[1] - fpixel[1]) * fits->width + ii]
                        = pixels[ii];
            }

            delete[] pixels;
            fits_close_file(fptr, &status);
        }
        fits_close_file(fptr, &status);
    }

    if (status) {
        fits_report_error(stderr, status);
        delete[] imgData;
        return NULL;
    }

    return imgData;
}
```

Listing 4.1: Function to read the image data from a FITS file

If we take a more detailed look on the primal-dual algorithm for the TV-L^1 model for example, we observe that in principal the algorithm only involves the pointwise proximal operators (see algorithm 2), whereas the input to these operators are the image derivatives, where we use finite differences, and the dual variable. We observe similar characteristics for the image registration (see algorithm 5), the computation of the probabilities, the segmentation, the ID propagation and the thinning of the filament blobs (see algorithm 9).

The GPUs are designed for realtime, high-definition 3D graphics. This involves the

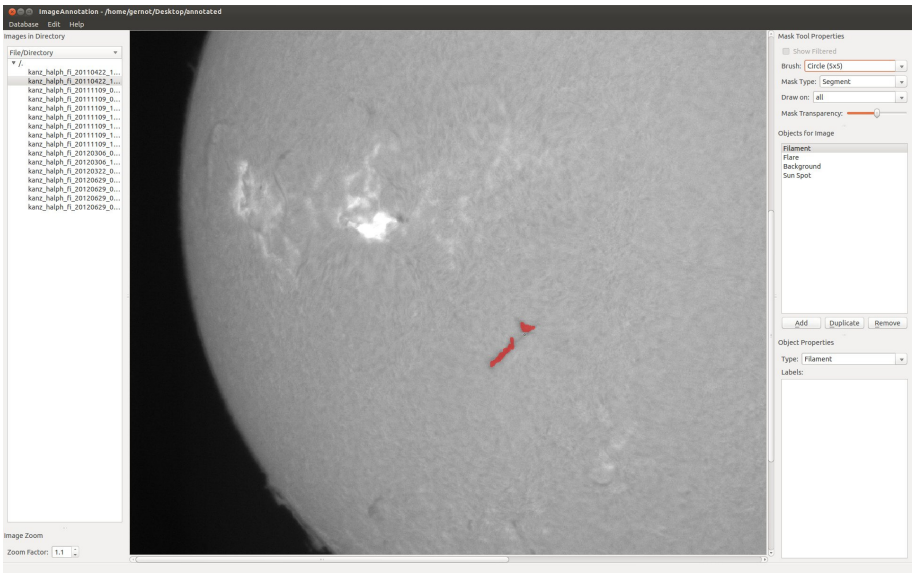


Figure 4.1: The modified image annotation tool showing an annotated filament.

highly-parallel computation of data, and therefore, more transistors on a GPU are devoted to data processing rather than data caching and flow control in contrast to CPUs. GPUs follow the principle of single instruction on multiple data (*SIMD*), whereas CPUs perform single instructions on single data (*SISD*). Especially this makes the GPU so attractive to algorithms as mentioned above, where we can apply the same set of instructions to every pixel of an image in parallel.

4.4 Compute Unified Device Architecture

As the computational power of GPUs is not only useful for 3D rendering, NVIDIA introduced the Compute Unified Device Architecture (*CUDA*)[‡] in 2006, a general-purpose parallel-computing platform and programming model. CUDA makes it possible to use a high-level programming language, an extension to C, to develop scientific and industrial application utilizing the advantages of modern GPUs.

It should be noted that there also exists an open standard for general purpose parallel computing, namely OpenCL[§]. This standard uniforms the programming environment for multi-core CPUs, GPUs and other parallel processors. However, all the parallelizable algorithms were implemented using the CUDA framework and therefore we introduce to

[‡]`http://www.nvidia.com/object/cuda_home_new.html`, last checked 2013-04-04
[§]`http://www.khronos.org/opencl/`, last checked 2013-04-04

the most important CUDA concepts needed for the implementation.

4.4.1 Programming with CUDA

The following is an introduction into the basic concepts of CUDA programming. As this cannot be seen as a complete coverage of the topic, we refer the interested reader to [65, 76].

The core of the language extension are the so called *kernels*. These are special functions, indicated by the `__global__` keyword, that are executed N times in parallel by N different threads. Each thread executing a kernel has a unique thread ID and this ID can be used for addressing in arrays. For example, if you want to add two vectors with N elements, you can simply use N threads and each thread is summing one component of the vector identified by the thread ID.

Threads are further grouped into blocks. A block allows a limited number of threads to run in parallel, whereas the actual number is limited by the hardware. Similar to the thread ID, there also exists a block ID and the dimension of the block that defines the number of threads running in a given block. Turning back to our addition example: If the vectors are really large and more threads are needed than fit into one block, we have to use more blocks. Additionally, the addressing of the component in the kernel function changes to $blockId \cdot blockDim + threadId$.

One performance-critical consideration is the used memory. CUDA threads have access to multiple memory spaces that differ in access speed and scope.

Local Memory Each thread has its own private memory and this memory is accessible with the lowest latency.

Shared Memory This type of memory is shared between the threads of a block and has the same lifetime. The shared memory buffer physically resides on the GPU and therefore is faster than the global memory.

Global Memory The global memory can be accessed from all threads, independent of the block. However, the global memory typically is implemented in the DRAM and therefore slower to access within a range of one or two magnitudes.

Constant Memory The constant memory is similar to the global one, but as the name suggests, it is a read-only memory. It is cached on the chip and therefore faster to access.

Texture Memory The texture memory is another read-only memory that has some additional properties. It is especially faster, if there exists a spatial locality in the memory access pattern. Further, it provides bilinear interpolation and border handling.

The CPU can communicate with the GPU via constant, global and texture memory. However, the memory transfer between the CPU and GPU is costly and should be avoided to gain optimal performance.

Chapter 5

Evaluation

Contents

In this chapter we will present an evaluation of the proposed method for the detection of flares and filaments in Hα image sequences. As there exist no public available datasets, or benchmarks, we decided to compare the results of our method with official observations from the KSO and the National Oceanic and Atmospheric Administration* (*NOAA*). Additionally for the evaluation of the filament segmentation, we created a dataset of 29 images that were annotated pixelwise by an expert.

5.1 Setting

All the experiments were conducted on a single PC with an Intel Core i7 CPU with a clock speed of 3.2 GHz and 11.7 GB RAM. The system was equipped with a modern CUDA enabled GPU, namely the NVIDIA GeForce GTX 680 with 4069 MB memory. As operating system a recent 64-bit Linux distribution was used and CUDA was installed in version 4.2.

As already mentioned, all the pictures have a pixel resolution of 2048×2048 pixels and a radiometric resolution of 12bit per pixel. It should be noted that a reduction of the

*`http://www.swpc.noaa.gov/ftpmenu/indices/events.html`

pixel resolution was not conducted, because it may deteriorate the recognition especially of small flares and narrow filaments. However, it could lead to a significant speed up.

In the single steps of our proposed method we use several parameters. In the first step, the preprocessing, the structural bandpass filter relies on two parameters, λ_1 and λ_2, that are related to the structure size. To remove large-scale variations, we use $\lambda_1 = 0.9$ and to suppress small scaled noise we set $\lambda_2 = 0.1$. In the postprocessing, we have first of all the number of images for the tracking as one parameter. Instead of using a fixed number of images, we use all images from the last 20 minutes. This avoids wrong tracking results caused by the rotation of the sun, if the image sequence has gaps. Especially for the filament postprocessing, we remove responses near sun spots and flares. In both cases we remove them within a 30 pixel radius of the sun spot and flare borders. The grouping of the filament parts involves an adaptive threshold, where we use $dist_{\min} = 5$ pixel and $dist_{\min} = 60$ pixel. The last parameter is the maximal distance for the flare grouping, where we use $t_d = 150$ pixel.

All the stated parameters were determined through numerous experiments.

5.2 Measures

To evaluate the obtained results, we will utilize well-known measures from information retrieval and classification tasks. Independent of the task, e.g. multi-label segmentation and event recognition, a single binary result can be correct or false. Therefore, we introduce the terms true positive (tp) and false positive (fp) that indicate that a positive result is correct, or false, respectively. Similarly we have the term true negative (tn) and false negative (fn).

The first accuracy-measurement is especially useful for segmentation tasks, as it measures the overlap between the ground truth segmentation and the resulting segmentation by using the intersection over the union (IoU) ratio. This is equivalent to the following definition of the accuracy

$$IoU = \frac{tp}{tp + fp + fn} \tag{5.1}$$

where the IoU is at maximum 1, if there were no false negatives and false positives and at minimum 0, if no true positives were classified.

To separately evaluate the number of events or pixels that were correctly classified compared to the false positives and false negatives, we introduce the precision measure.

Similarly we can gain information about the number of correctly classified samples in contrast to the missed values with the recall measure. Both are defined as follows

$$precision = \frac{tp}{tp + fp} \tag{5.2}$$

$$recall = \frac{tp}{tp + fn} \tag{5.3}$$

where both measures can take values in the closed interval $[0, 1]$.

Finally, there also exists a score that combines the measures described above. The F-score is the harmonic mean of the precision and the recall and is computed as follows

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.4}$$

where the F-score can also take values in the closed interval $[0, 1]$ only.

5.3 Flares

For the evaluation of the flare detection, we tested our method on archive images of two months. We then compared the obtained results to the official data of the NOAA and the visual inspection results of the KSO. A flare is correctly detected, if the start and end time matches up to twelve minutes and the position differs not more than ten degrees. Further, it should have roughly the same size, and therefore we utilize the categorization as listed in table 1.1.

As our method delivers the position in image coordinates and the area in number of pixels, we have to convert it to sun coordinates and 10^{-6} of sun hemispheres, respectively. For this purpose, we make use of an already existing script of the KSO.

As sample months, we took the data from July 2012 and August 2012. In these two months, the sun was rather active in terms of flare occurrences. Further, we did no selection of the images based on their quality as it will be done in the realtime application on the observatory. Finally, we did not use every image per day, but only images within a 30 second period.

First, we considered all flares that have a category of N, or higher. These are the flares that are relevant for space weather and can have an impact on the Earth's infrastructure and near earth objects. Table 5.1 summarizes the results for the two selected months. Of 25 occurred flares, we detected 22 correctly and had 1 false positive. This yields a precision of 0.95652, a recall of 0.88000 and an F-score of 0.91667.

date	true positives	false positives	false negatives
2012-07-02	1	0	0
2012-07-03	3	0	0
2012-07-04	2	0	0
2012-07-05	2	0	0
2012-07-06	3	0	0
2012-07-08	0	0	2
2012-07-09	2	0	0
2012-07-10	2	0	0
2012-07-11	0	0	1
2012-07-14	1	0	0
2012-07-29	1	0	0
2012-07-30	1	0	0
2012-08-04	0	1	0
2012-08-09	1	0	0
2012-08-11	1	0	0
2012-08-13	1	0	0
2012-08-15	1	0	0
total	**22**	**1**	**3**

Table 5.1: Evaluation of the flare detection in Hα image sequences ignoring sub-flares. The table presents only days, where a flare with importance class 1 or higher occurred.

The one false positive can be explained by the very bad viewing conditions. The flare was correctly detected, but due to the clouds the flare appeared brighter and larger as it really was. In contrast, the false negative shares a similar location close to the limb. The flares were correctly detected, but the size was underestimated and therefore led to a false categorization.

If we also consider flares in the sub-flare category, the results get worse. First, the problems mentioned get even more difficult for smaller flares that additionally have only a small life span. Further, most of the false positives can be explained by gaps of a few minutes in the image data. The results are summarized in table 5.2. Using these results in our previously stated measures, we obtain a precision of 0.82667, a recall of 0.83221 and an F-score of 0.82943.

date	true positives	false positives	false negatives
2012-07-01	8	1	6
2012-07-02	5	2	3
2012-07-03	7	0	3
2012-07-04	4	1	1
2012-07-05	2	0	3
2012-07-06	4	0	1
2012-07-07	5	0	0
2012-07-08	5	0	0
2012-07-09	6	0	1
2012-07-10	4	2	0
2012-07-11	2	0	0
2012-07-12	0	0	0
2012-07-13	no data		
2012-07-14	7	0	0
2012-07-15	0	0	0
2012-07-16	1	0	0
2012-07-17	0	0	0
2012-07-18	0	0	0
2012-07-19	1	0	0
2012-07-20	0	0	0
2012-07-21	no data		
2012-07-22	0	0	0
2012-07-23	0	0	0
2012-07-24	0	0	0
2012-07-25	0	0	0
2012-07-26	0	0	0
2012-07-27	5	0	0
2012-07-28	1	1	0
2012-07-29	3	0	1
2012-07-30	1	1	0
2012-07-31	8	0	1
2012-08-01	4	1	1
2012-08-02	1	0	0
2012-08-03	2	0	0
2012-08-04	0	2	0
2012-08-05	0	0	0
2012-08-06	0	0	1
2012-08-07	5	0	0
2012-08-08	2	2	1
2012-08-09	4	1	0

date	true positives	false positives	false negatives
2012-08-10	2	0	0
2012-08-11	1	0	0
2012-08-12	1	1	0
2012-08-13	1	1	0
2012-08-14	1	1	0
2012-08-15	3	0	0
2012-08-16	0	0	0
2012-08-17	0	0	0
2012-08-18	8	0	2
2012-08-19	4	0	0
2012-08-20	1	0	0
2012-08-21	0	1	0
2012-08-22	0	0	0
2012-08-23	0	0	0
2012-08-24	1	1	0
2012-08-25	0	0	0
2012-08-26	no data		
2012-08-27	0	0	0
2012-08-28	0	0	0
2012-08-29	0	4	0
2012-08-30	4	3	0
total	**124**	**26**	**25**

Table 5.2: Evaluation of the flare detection in Hα image sequences. The table summarizes the number of true positives, false positives and false negatives per day, where we consider the size, time of occurrence and position of the flares. For the days 2012-07-13, 2012-07-21 and 2012-08-26 no data was available.

5.4 Filaments

5.4.1 Segmentation Accuracy

We detect a filament eruption, which means the disappearance of a filament structure from the observable Hα image, by tracking all visible filaments. For this evaluation we say that a filament is erupted, if it could not be tracked in the image sequence for 15 minutes. This implies that a good segmentation of the filaments is crucial.

Therefore, we first evaluate the pixelwise segmentation results of our method. For this purpose, an expert has annotated the filaments of 28 Hα images - one image per day of the month July 2012 where the sun could be observed due to the weather conditions. An

example annotation from the ground truth is illustrated in figure 5.1.

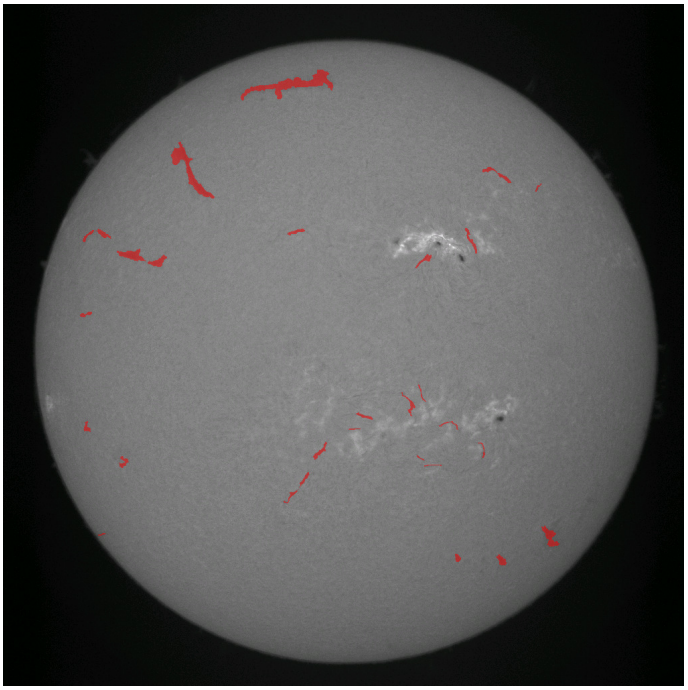


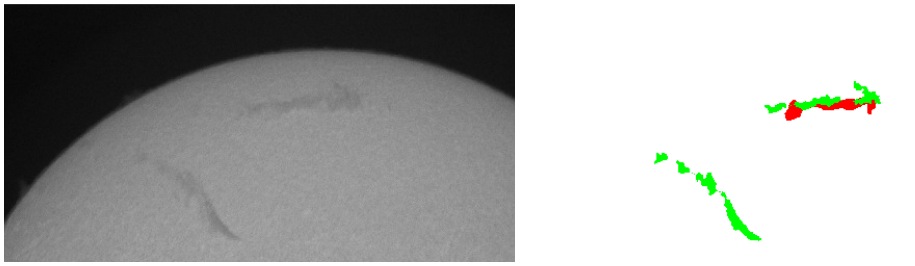
Figure 5.1: An example filament annotation from the ground truth data. The red color indicates the annotated filament pixels

For the segmentation result we first compute the intersection-overlap ratio with equation (5.1). This yields an accuracy of 0.799825. For precision and recall we obtain the values 0.92985 and 0.851186, respectively and the F-score is equal to 0.888781.

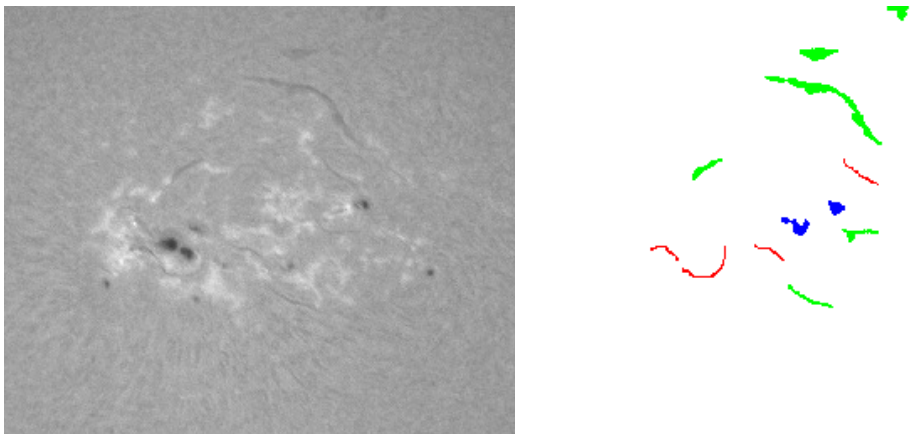
The higher precision indicates that we segment few pixels that do not belong to any filament. By studying the results it is evident, that most of the false positives are yielded near the limb, where the contrast between filaments and the solar background decreases.

The false negative results share the similar problem. As illustrated in figure 5.2(a), filaments near the limb are difficult to segment completely. It also demonstrates the requirement for a reconnection of the filament parts. Another problem are very narrow filaments, especially in active regions, as shown in figure 5.2(b).

It should further be noted that the annotation of the filaments is not always clear, as a single filament can appear as several parts due to the contrast in the Hα images. Further, there exist no precise edges between the background and the filament.



(a)



(b)

Figure 5.2: The left images show parts of different Hα images and on the right ones a comparison between the ground truth and the result of our proposed method is depicted. Green indicates true positives, red false negatives and blue false positives.

5.4.2 Filament Eruptions

For the evaluation of the flare eruptions, we tested our method on archive images of one month. We compared the obtained results to the official data of the NOAA and to the visual inspection results of the KSO, mainly because the NOAA data is not accurate to a 100%.

As a sample month, we took the archive data from July 2012. For the archive data, we took images in a 30 seconds period without further selection in terms of cloudy images.

Table 5.3 summarizes the evaluation for the month July 2012. Overall, four filament eruptions occurred in July 2012, whereas the method was able to detect all of them correctly. Additionally, we detected one false positive on 2012-07-03. In this case the

method failed, because the filament was not segmented in several images due to clouds. This implies a precision of 0.80000, a recall of 1.00000 and an F-score of 0.88889.

date	true positives	false positives	false negatives
2012-07-03	0	1	0
2012-07-08	1	0	0
2012-07-11	2	0	0
2012-07-26	1	0	0
total	**4**	**1**	**0**

Table 5.3: Evaluation of the filament eruptions in the Hα image sequences of July 2013. The table presents only days, where a filament eruption occurred.

5.5 Discussion

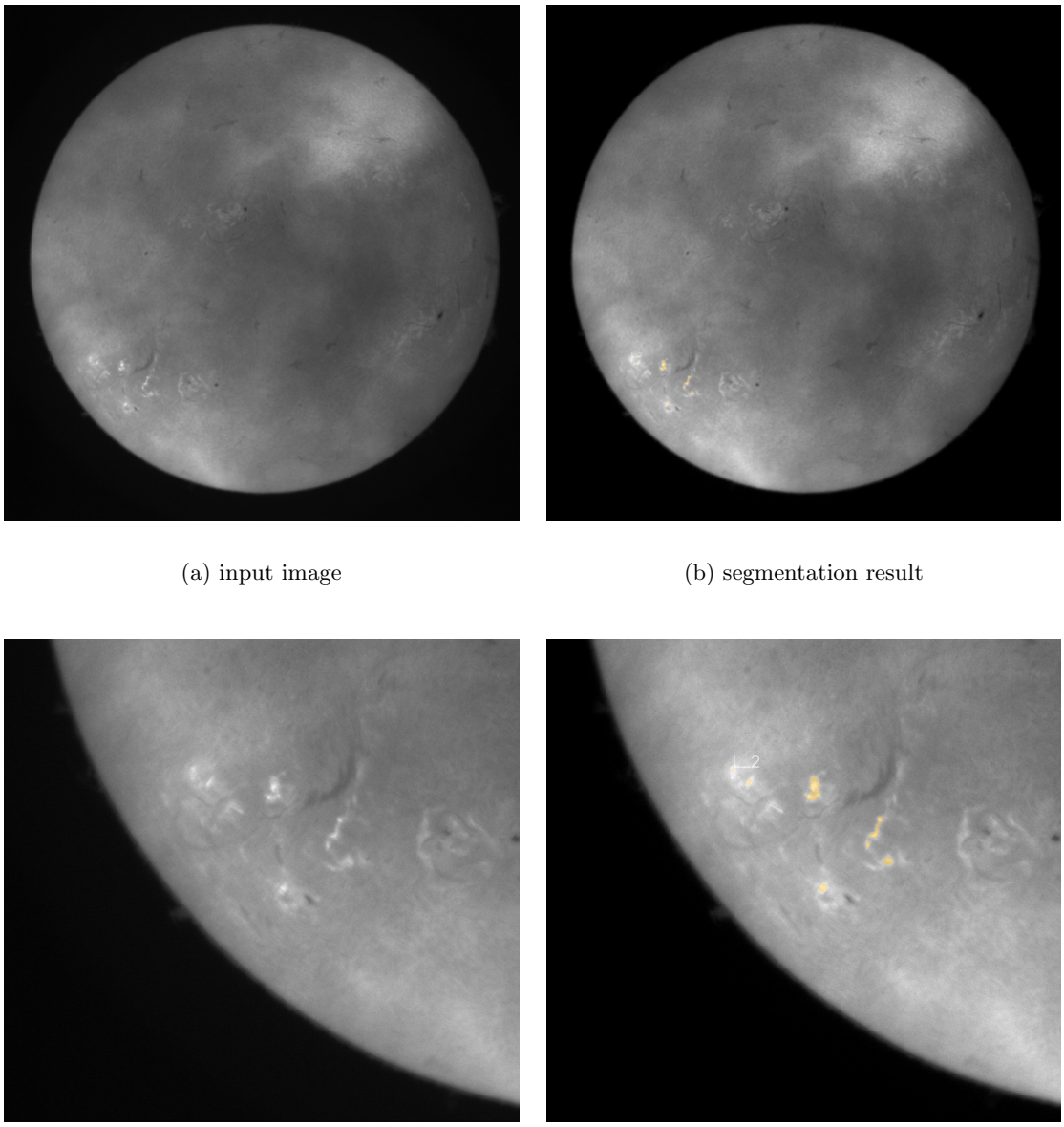
The presented results underline the power of our proposed method for the detection of flares and filaments in Hα image sequences. For the relevant flare categories the segmentation works very good, as long as not most of the solar disk is covered by clouds. This was one of the main reasons for the detection failure. The flares may appear bigger and brighter, if they are observed through clouds. See figure 5.3 for example.

Another problem is the categorization closer to the limb. First, the conversion between the area in the number of pixels to sun hemispheres gets less precise, independent of the segmentation result. But also the mean and maximum intensity of the segmented areas decreases, which leads to a more erroneous categorization.

These problems get even worse for sub-flares. However, if we consider that even the data of KSO and NAOO differs many times, our results actually are reasonably well. The results even may be improved, if the flare categorization is learned by an appropriate algorithm based on the distance of the flare center to the solar center and the segmented size and intensities.

In the evaluation of the filament segmentation and the detection of filament eruptions, we basically have the same problems. As illustrated in figures 5.2 and 5.4, the accuracy of the segmentation decreases towards the limb and is affected if the sun is covered by clouds. The structural bandpass filter equalizes the intensities covered by clouds, but if a filament is not visible due to clouds, the filter cannot make it visible.

The filament segmentation in combination with the postprocessing, including the grouping of the filament parts, as well as the tracking, has proven to be successful in



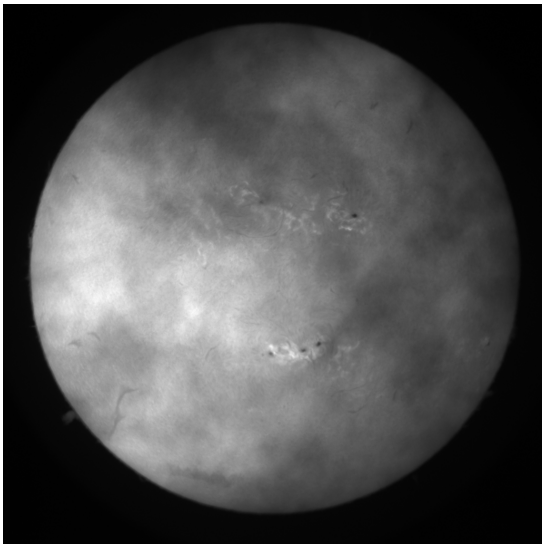
(a) input image

(b) segmentation result

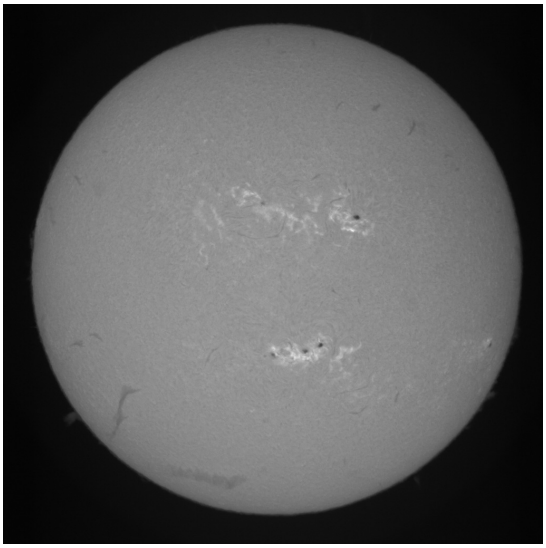
(c) input image (detailed)

(d) segmentation result (detailed)

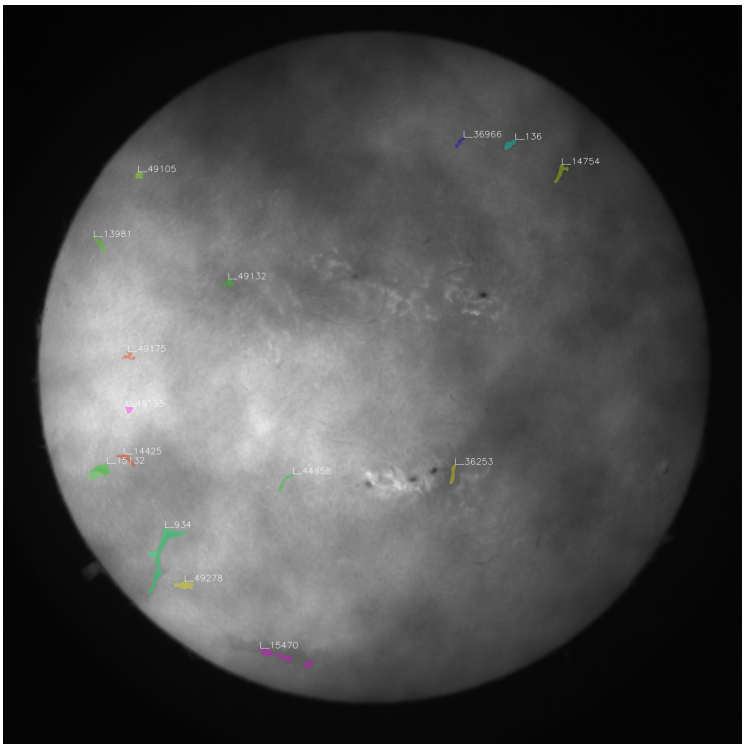
Figure 5.3: The image on the left illustrates an Hα image covered by clouds taken on 2012-08-04, where a flare occurs on the bottom left. The right image shows the same image including the flare segmentation result of our method. Due to clouds, the segmented flare area is larger then it really is and therefore the categorization failed.



(a) input image



(b) solar disk 7 minutes later



(c) segmentation result

Figure 5.4: The image (a) illustrates an Hα image covered by clouds taken on 2012-08-04. The image in (c) shows the same image including the filament segmentation result of our method. The image (b) illustrates the solar disk of the same day seven minutes later. The segmentation of the filaments close to the limb and covered by clouds clearly gets worse (see left bottom of image (c)). Additionally, the small filaments on the right, which are completely covered by clouds could not be detected.

the evaluation. A recall of 100% compared to one false positive due to the viewing conditions clearly demonstrates the advantages of our method.

As we have noted at the beginning of the evaluations, we performed no pre-selection of the images from the archive data based on quality. This is in contrast to the actual use case, where the live images automatically get selected based on viewing conditions. Therefore, the overall performance of the live system should improve significantly for flare recognition, but also for the detection of filament eruptions.

Chapter 6

Conclusion

Contents

6.1 Summary

As introduced in chapter 1, space related to the activity of the sun can disturb critical infrastructure near and on Earth. Some of the events that are associated with solar storms, namely flares and filament eruptions, are observable in Hα images from ground based observatories.

In this master thesis, we presented a novel method for the simultaneous detection and segmentation of flares and filaments in Hα image sequences. Furthermore, the method allows the recognition of filament eruptions by tracking the single filaments in the image sequence. The application combines state of the art algorithms from the areas of computer vision and machine learning as presented in chapter 2.

The actual method consists of four major blocks. The first step, the preprocessing, is presented in section 3.3. Feature selection and model learning is described in section 3.4, where we illustrated features discriminating the filaments and flares from the background and evaluating which model is the most efficient in terms of classification accuracy. Section 3.5 is devoted to the multi-label segmentation of the Hα images to regularize the classification result. The final postprocessing steps, which also include the derivation of properties from the objects to categorize them, are described in detail in section 3.6.

Most of the building blocks utilize the enormous parallel computation capabilities of

modern GPUs. Therefore, our method is able to process the Hα images in near realtime. More details on the implementation are presented in chapter 4.

In chapter 5, we finally compared the detection results of our method with categorization reports of experts visually inspecting the Hα images as well as with annotated Hα images by comparing the segmentation results pixelwise.

6.2 Outlook

For future work, the method could be improved in several places. The detection and segmentation of the filaments has some room for enhancements. The investigation of new features emphasizing the appearance of filaments would be beneficial. The possibility of incorporating information about the elongated structure of the filaments in the multi-label segmentation might improve the results, too.

Another not completely solved issue are the interrupted filament parts. Not only would it be interesting to evaluate the effects of coherence enhancing diffusion [85] for the completion of the filaments, but also to utilize higher order information of the filament shape for this purpose.

Although the application is intended to be completely autonomous, it could be advantageous to incorporate some user feedback. As filaments stay nearly unchanged over several days, a user input once or twice a day may dramatically improve the overall results.

While writing this thesis, the developed method was implemented and tested at the KSO in realtime. The obtained results will deliver new insights and additionally more data for evaluation.

Bibliography

[1] Aboudarham, J., Scholl, I., Fuller, N., Fouesneau, M., Galametz, M., Gonon, F., Maire, A., and Leroy, Y. (2008). Automatic detection and tracking of filaments for a solar feature database. In *Annales geophysicae: atmospheres, hydrospheres and space sciences*, volume 26, page 243.

[2] Adams, R. and Bischof, L. (1994). Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):641–647.

[3] Alliney, S. (1992). Digital filters as absolute norm regularizers. *Signal Processing*, 40(6):1548–1562.

[4] Alliney, S. (1997). A property of the minimum vectors of a regularizing functional defined by means of the absolute norm. *Signal Processing*, 45(4):913–917.

[5] Antia, H. M., Bhatnagar, A., and Ulmschneider, P. (2003). *Lectures on Solar Physics*, volume 619. Springer.

[6] Aschwanden, M. J. (2009). Image Processing Techniques and Feature Recognition in Solar Physics. *Solar Physics*, 262(2):235–275.

[7] Aubert, G. and Kornprobst, P. (2006). *Mathematical problems in image processing: partial differential equations and the calculus of variations*, volume 147. Springer.

[8] Aujol, J. F., Gilboa, G., Chan, T., and Osher, S. (2006). Structure-texture image decomposition—modeling, algorithms, and parameter selection. *International Journal of Computer Vision*, 67(1):111–136.

[9] Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press.

[10] Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

[11] Bernasconi, P. N., Rust, D. M., and Hakim, D. (2005). Advanced Automated Solar Filament Detection And Characterization Code: Description, Performance, And Results. *Solar Physics*, 228(1-2):97–117.

[12] Bhatnagar, A. and Livingston, W. C. (2005). *Fundamentals of Solar Astronomy*, volume 6. World Scientific Publishing Company Incorporated.

[13] Bishop, C. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

[14] Bobrinsky, N. and Del Monte, L. (2010). The space situational awareness program of the European Space Agency. *Cosmic Research*, 48(5):392–398.

[15] Bouguet, J. Y. (2001). Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*.

[16] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[17] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239.

[18] Bredies, K.; Lorenz, D. (2011). *Mathematische Bildverarbeitung: Einführung in Grundlagen und moderne Theorie*. Springer.

[19] Breiman, L. (1984). Classification and regression trees.

[20] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

[21] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[22] Bresenham, J. (1965). Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30.

[23] Cannon, F., Angling, M., Barclay, L., Curry, C., Edwards, R., Greene, G., Hapgood, M., Home, R., Jackson, D., Mitchell, C., Owen, J., Richards, A., Rogers, C., Ryden, K., Saunders, S., Sweeting, M., Tanner, R., Alan, T., and Underwood, C. (2013). Extreme Space Weather: Impacts on Engineered Systems and Infrastructure. Technical report, Royal Academy of Engineering, London.

[24] Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698.

[25] Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *International Conference on Machine Learning*, pages 161–168. ACM.

[26] Chambolle, A. (2004). An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1-2):89–97.

[27] Chambolle, A., Caselles, V., Cremers, D., Novaga, M., and Pock, T. (2010). An introduction to total variation for image analysis. *Theoretical Foundations and Numerical Methods for Sparse Recovery*, 9:263–340.

[28] Chambolle, A. and Lions, P. L. (1997). Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(2):167–188.

[29] Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145.

[30] Chan, T. and Esedoglu, S. (2005). Aspects of total variation regularized L 1 function approximation. *SIAM Journal on Applied Mathematics*, 65(5):1817–1837.

[31] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[32] Crammer, K. and Singer, Y. (2002). On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292.

[33] Criminisi, A. and Shotton, J. (2013). *Decision Forests for Computer Vision and Medical Image Analysis*. Springer.

[34] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

[35] Douglas, J. and Rachford, H. H. (1956). On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439.

[36] Duan, K. B. and Keerthi, S. (2005). Which is the best multiclass SVM method? An empirical study. *Multiple Classifier Systems*, pages 732–760.

[37] Echer, E., Gonzalez, W. D., Guarnieri, F. L., Lago, A. D., and Vieira, L. E. A. (2005). Introduction to space weather. *Advances in Space Research*, 35(5):855–865.

[38] Fernandez Borda, R. A., Mininni, P. D., Mandrini, C. H., Gómez, D. O., Bauer, O. H., and Rovira, M. G. (2002). Automatic solar flare detection using neural network techniques. *Solar Physics*, 206(2):347–357.

[39] Fleming, W. H. and Rishel, R. (1960). An integral formula for total gradient variation. *Archiv der Mathematik*, 11(1):218–222.

[40] Fletcher, T. (2009). Support vector machines explained.

[41] Floyd, R. W. (1962). Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345.

[42] Frangi, A., Niessen, W., Vincken, K., and Viergever, M. (1998). Multiscale vessel enhancement filtering. *Medical Image Computing and Computer-Assisted Interventation*, 1496:130–137.

[43] Friedman, H. (1987). *Die Sonne: Aus der Perspektive der Erde*. Spektrum der Wissenschaft.

[44] Fuller, N. and Aboudarham, J. (2004). Automatic detection of solar filaments versus manual digitization. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 467–475. Springer.

[45] Fuller, N., Aboudarham, J., and Bentley, R. D. (2005). Filament Recognition and Image Cleaning on Meudon Hα Spectroheliograms. *Solar Physics*, 227(1):61–73.

[46] Gall, J. and Lempitsky, V. (2009). Class-specific hough forests for object detection. In *Computer Vision and Pattern Recognition*, pages 1022–1029. IEEE.

[47] Gao, J., Wang, H., and Zhou, M. (2002). Development of an Automatic Filament Disappearance Detection System. *Solar Physics*, 205:93–103.

[48] Guo, Z. and Hall, R. W. (1989). Parallel thinning with two-subiteration algorithms. *Communications of the ACM*, 32(3):359–373.

[49] Hadamard, J. (1902). Sur les problèmes aux dérivés partielles et leur signification physique. *Princeton University Bulletin*, 13:49–52.

[50] Hastie, T., Tibshirani, R., and Friedman, J. (2001). The Elements of Statistical Learning: Data Mining, Inference, and Prediction.

[51] Hsu, C. W. and Lin, C. J. (2002). A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425.

[52] Idier, J. (2010). *Bayesian approach to Inverse problems*. John Wiley and Sons.

[53] Kaipio, J. and Somersalo, E. (2004). *Statistical and computational inverse problems*, volume 160. Springer.

[54] Knapik, B. T., van Der Vaart, A. W., and Van Zanten, J. H. (2011). Bayesian inverse problems with Gaussian priors. *The Annals of Statistics*, 39(5):2626–2657.

[55] Legendre, A. M. (1805). *Nouvelles méthodes pour la détermination des orbites des cometes*. F. Didot.

[56] Lin, J., Forbes, T. G., and Isenberg, P. (2001). Prominence eruptions and coronal mass ejections triggered by newly emerging flux. *Journal of Geophysical Research. A. Space Physics*, 106:25.

[57] Lindeberg, T. (1993). *Scale-space theory in computer vision*. Springer.

[58] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*.

[59] Michelot, C. (1986). A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$. *Journal of Optimization Theory and Applications*, 50(1):195–200.

[60] Mumford, D. and Shah, J. (1989). Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 42(5):577–685.

[61] Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective.

[62] Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate O (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376.

[63] Nikolova, M. (2002). Minimizers of cost-functions involving nonsmooth data-fidelity terms. Application to the processing of outliers. *SIAM Journal on Numerical Analysis*, 40(3):965–994.

[64] Nikolova, M. (2004). A variational approach to remove outliers and impulse noise. *Journal of Mathematical Imaging and Vision*, 20(1):99–120.

[65] NVIDIA (2012). *CUDA C Programming Guide*. NVIDIA Cooperation.

[66] Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

[67] Pence, W. D., Chiappetti, L., Page, C. G., Shaw, R. A., and Stobie, E. (2010). Definition of the Flexible Image Transport System (FITS), version 3.0. *Astronomy & Astrophysics*, 524.

[68] Potts, R. B. (1952). Some generalized order-disorder transformations. In *Proceedings of the Cambridge Philosophical Society*, volume 48, pages 106–109. Cambridge Univ Press.

[69] Qahwaji, R. and Colak, T. (2005). Automatic detection and verification of solar features. *International Journal of Imaging Systems and Technology*, 15(4):199–210.

[70] Qu, M., Shih, F., Jing, J., and Wang, H. (2004). Automatic solar flare tracking using image-processing techniques. *Solar Physics*, 222(1):137–149.

[71] Qu, M., Shih, F. Y., Jing, J., and Wang, H. (2003). Automatic solar flare detection using MLP, RBF, and SVM. *Solar Physics*, 217(1):157–172.

[72] Qu, M., Shih, F. Y., Jing, J., and Wang, H. (2005). Automatic solar filament detection using image processing techniques. *Solar Physics*, 228(1):119–135.

[73] Quinlan, J. R. (1993). *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann.

[74] Rockafellar, R. T. (1970). *Convex Analysis*. Princeton University Press.

[75] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268.

[76] Sanders, J. and Kandrot, E. (2011). *CUDA by Example*. Addison-Wesley.

[77] Scherzer, O., Grasmair, M., Grossauer, H., Haltmeier, M., and Lenzen, F. (2008). *Variational methods in imaging*, volume 167. Springer.

[78] Shapiro, L. and Stockman, G. C. (2002). Computer Vision. 2001.

[79] Shih, F. Y. and Kowalski, A. J. (2003). Automatic Extraction of Filaments in Hα Solar Images. *Solar Physics*, 218(1/2):99–122.

[80] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition*, pages 1297–1304. IEEE.

[81] Szeliski, R. (2010). *Computer vision: Algorithms and Applications*. Springer.

[82] Temmer, M., Veronig, A., Hanslmeier, A., Otruba, W., and Messerotti, M. (2001). Statistical analysis of solar Halpha flares. *Astronomy and Astrophysics*, 375(3):1049–1061.

[83] Tikhonov, A. N. and Arsenin, V. Y. (1977). Solutions of ill-posed problems. *WH Winston*, 21.

[84] Veronig, A., Steinegger, M., Otruba, W., Hanslmeier, A., Messerotti, M., Temmer, M., Brunner, G., and Gonzi, S. (2000). Automatic image segmentation and feature detection in solar full-disk images. In *The Solar Cycle and Terrestrial Climate, Solar and Space weather*, volume 463, page 455.

[85] Weickert, J. (1999). Coherence-enhancing diffusion filtering. *International Journal of Computer Vision*, 31(2-3):111–127.

[86] Wells, D. C., Greisen, E. W., and Harten, R. H. (1981). FITS-a flexible image transport system. *Astronomy and Astrophysics Supplement Series*, 44:363.

[87] Yuan, Y., Shih, F. Y., Jing, J., Wang, H., and Chae, J. (2011). Automatic Solar Filament Segmentation and Characterization. *Solar Physics*, 272(1):101–117.

[88] Zach, C., Gallup, D., Frahm, J. M., and Niethammer, M. (2008). Fast global labeling for real-time stereo using multiple plane sweeps. In *Vision, modeling, and visualization-odeling, and Visualization*, page 243. IOS Press.

[89] Zharkova, V. V. and Schetinin, V. (2005). Filament Recognition In Solar Images With The Neural Network Technique. *Solar Physics*, 228(1-2):137–148.

[90] Zharkova, V. V. S. V. (2003). A Neural-Network Technique for Recognition of Filaments in Solar Images. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 2773 of *Lecture Notes in Computer Science*, pages 148–154. Springer.

[91] Zirin, H. (1988). Astrophysics of the Sun.