Image Similarity in Medical Images

Viktor Gál

Proefschrift voorgedragen tot het behalen van de graad van
Doctor in de Wetenschappen, 2015

Promotor:
prof. Dr. Etienne E. Kerre

Co-Promotors:
prof. Dr. Mike Nachtegael
prof. Dr. Chris Cornelis

Vakgroep Toegepaste Wiskunde, Informatica en Statistiek
Faculteit Wetenschappen, Universiteit Gent
Krijgslaan 281, S9, B-9000 Gent

To Lea

Acknowledgment

First of all, I would like to point out that this work would not have been possible has been funded by the European Commission under the MIBISOC project (Grant Agreement: 238819, within the action Marie Curie Initial Training Network of the FP7).

I would like thank my advisors, Etienne Kerre and Mike Nachtegael, for giving me the opportunity to be a part of the MIBISOC project[1] and for the support and guidance they gave me throughout all the phases of my doctorate.

My journey in computer science research would not have be possible without the mentorship, friendship of Domonkos Tikk. He always managed to pushed me to my limits and get the most out of me.

I wish to thank Luis Gonzalez Jaime for the friendship and time we have spent together in Gent.

I cannot be greatful enuogh for Gabor for his friendship, to cheering me up and showing me the direction every time I lost my ways.

In particular, I am greatful for knowning Adél néni (Adél Gábos) and having her as my maths teacher (and adopted grandmother). Without her maths weekend sessions and optional extra maths classes I probably would have never developed such a passion for mathematics and poems. Her strength and passion for live always inspire me to never give up and continue on the path.

[1] "MIBISOC: Medical Imaging Using Bio-inspired and Soft Computing", which is a Marie Curie Initial Training Network granted by the European Commission within the Seventh Framework Program (FP7 PEOPLE-ITN-2008).

Last, but not least I would like to thank my family for the ever lasting patience, understanding and support. Without them I would not have managed to get to this point in my life.

Contents

List of Figures

List of Tables

Abstract

Visual information contained in images is a very important and rich source of knowledge, exploiting this information is indispensable for certain professions. Good examples are social media, satellite imagery for remote sensing and medical images for research and education. To use this information it is often necessary to be able to find a certain type of images, for example rivers on a satellite image or computer tomography of a certain body part. To be able to do this image classification or content based image retrieval is required.

Image classification determines the category of an image based on a variety of features. These features may be artificial, such as captions or other meta-data and they can be embedded in the visual data itself. Visual features can be as simple as the presence or dominance of a colour, or certain edges, or as complex as the presence of a deformable object at a random viewing angle or scale. Feature extraction transforms the pixel data from image space into feature-space, where—if the features are well chosen—finding the appropriate class of an image is a much more manageable task. For the best results an intelligent combination of several features—feature fusion—and machine learning algorithms are used.

One of the most active and important fields for automated image classification is medical imaging. Classifiers can be used to decide image modality i.e. the type of instrument it was taken with—MRI, CT, X-ray etc. User-studies done on clinicians have indicated that modality is one of the most important filters that they would like to be able to limit their search by on medical image databases. Another important use is the detection or

segmentation of a certain sub-part of the image to find out if it contains an organ and highlight it for further processing or examination.

Most state of the art classification methods are tailored to a certain dataset and rely on features characteristic of the images within. This means they can achieve high levels of accuracy, but at the price of losing flexibility. In this thesis this problem is addressed by introducing general configurable frameworks for classification and segmentation tasks.

The framework for image modality classification is built using a novel approach, with flexibility as a priority. It is composed of a configurable *Feature Extractor* that creates feature vectors on the input data. A *Feature Encoder* aggregates—fuses—the vectors to create better distinguishable and lower dimension input for the Machine Learning (ML) based *Classifier* component. Finally elements of the processing pipeline are optimised by a *Hybrid Parameter Optimiser* component. In general, the choice of the most appropriate components is dependent on the problem at hand. One of the key strengths of the proposed framework is being able to easily replace framework components to tailor the system to specific data sets.

As mentioned above, education and reliable manual or automated detection of an illness often require segmentation of a certain organ from an image. The framework proposed for this task consists of a *Localiser* that is responsible for detecting a *Region of Interest* within an image, a *Deformable Model* (DM) whose purpose is to precisely mark the final result, a *Term Set* that is the collection of relevant features that mark the desired part—in this case the organ—and a *Driver* that guides the DM in the correct configuration. A key aspect of the framework is the role of the driver. While in the vast majority of the works in literature DM evolution is tackled as an energy minimisation problem, in this thesis the driver directly guides the evolution of the model using a model derived from a ML method.

Although the examples and specific implementations in this thesis are focused on medical use, the proposed frameworks are readily applicable to general image classification and segmentation problems, due to the flexibility of the components.

The structure of the thesis is as follows, first a thorough introduction

is given on the building blocks of the proposed frameworks, complete with references to their use from the literature and related concepts. Then the framework for image modality is described, first through its components then a concrete application and classification results are shown on the ImageCLEF 2011 medical modality classification data set [91]. This is followed by the framework for segmentation—organ detection. Again the components of the system are described in detail, then the proposed framework is introduced together with concrete lung and hippocampus detection examples on the Segmentation in Chest Radiographs (SCR) and the Allen Brain Atlas (ABA) data sets respectively. The implemented system detected the right lung with 69.4% accuracy and marked the rough location of the hippocampus on all test images. Finally conclusions, possible improvements and future work are presented.

Introduction

As digital photography became ubiquitous and digital images are being used increasingly to store information it is becoming more and more important to be able to categorize the vast amount of data available from them. The most obvious examples are images from outer space in remote sensing applications, images of streets from all around the globe, medical images such as mammography or CT images, or a simple web search looking for pictures of an animal a person or other specific thing. The goal of classification is to decide whether a given image fits a predefined category for example whether it contains a bicycle, was taken outdoors, is an X-ray image or a portrait etc. Classification is a multi-step process, at its core it relies on features of an image to decide which category it fits best. The features are dependent on the desired classes and the images themselves, they can be meta-data such as captions, date taken etc., or features extracted from raw pixel data. The purpose of feature extraction is to create a representation of the data in feature space where it becomes easier to separate. To distinguish between the images in feature space various methods can be used, supervised or unsupervised machine learning methods, soft computing and many others. Closely related to classification is the problem of content based segmentation that is to find and mark a given object or feature on an image, independently—to the extent possible—of scale, rotation, illumination and noise. Detecting the presence of the object is classification in itself, marking the contours is an additional effort. It can serve simply as a tool for directing attention, or as a preprocessing step for a higher level process for example detecting a face, then deciding the gender, detecting text, then reading it, or detecting an organ then search for abnormalities.

Most classification methods are heavily specialized, i.e. they are custom made for a given data set with a feature set tailored to the body of images to be processed. This lack of flexibility prevents most state of the art algorithms to be used across different domains, on different data sets.

Objectives

The purpose of this PhD dissertation is to create new, flexible image classification frameworks to be used in general image modality detection and object segmentation applications. To this end the state of the art is researched in:

- Feature extraction, feature descriptors
- Classification related machine learning and soft computing methods
- Deformable models for marker placement
- Image classification methodology

The overall intention is to lay out a set of well-defined steps for a process that is able to perform image classification on a wide range of image data sets. The steps themselves are customizable by the user to tailor the process to the given classification needs and the data set, but the user has to be given enough tools to be able to handle a variety of different sets such as hospital patient records, (medical) journal articles, etc. As a secondary goal, to facilitate further processing, a content based segmentation framework is to be created based on the same principles of flexibility.

To demonstrate the versatility and usefulness of the proposed frameworks example applications are implemented on well-known freely available data sets, so that fair comparison can be made to other works in the literature. As a practical approach the domain of medical image processing is chosen to implement and demonstrate the effectiveness of the proposed frameworks. Medical application is considered as it is an active and important research area in this domain, therefore good quality datasets, with

real-world relevance are available and thus the produced results can be compared to existing ones from the literature. Since analyzing medical images is a very important task in modern medicine, results of these investigations are of clear practical value which gives additional motivation to this work. It was decided to use a data set of mixed modality images for the modality classifier experiment and to perform organ segmentation as the demonstrative experiment for image segmentation. To sum it up the goals of this thesis work are the following:

- Give an overview of the state of the art and background information on relevant techniques to introduce the concepts developed in the rest of the thesis.
- Describe a general, configurable framework for medical image modality classification.
- Introduction of a novel automatic image segmentation framework based on machine learning and deformable models.
- Demonstrate the viability of the proposed frameworks through examples taken from medical image data sets through example implementations, modality classification and organ segmentation.

Image modality classification

According to ImageCLEF [76]—a project dedicated to automatic annotation and retrieval of images—image modality is a very important aspect for image retrieval, and the problem is far from being solved: "Previous studies have shown that imaging modality is an important aspect of the image for medical retrieval. In user-studies, clinicians have indicated that modality is one of the most important filters that they would like to be able to limit their search by. Many image retrieval websites (Goldminer, Yottalook) allow users to limit the search results to a particular modality. However, this modality is typically extracted from the caption and is often

not correct or present. Studies have shown that the modality can be extracted from the image itself using visual features. Additionally, using the modality classification, the search results can be improved significantly."

For the image modality classification experiment the data set from ImageCLEF2011 [91] was used. The set was created with the purpose of testing image modality classifier algorithms and contains 988 training and 1024 testing images of 18 different modalities. The images themselves were taken from PubMed journal articles.

Organ segmentation

Organ segmentation was chosen as a demonstrative experiment because segmentation of organs is one of the major tasks in medical imaging. Most of the segmentation algorithms operate locally, i.e. they depend on a manual placement of seed points in order to successfully segment the desired structure. Thus, an automatic organ segmentation system, which is capable to segment various organs without manual labour, requires an organ detection model that can automatically provide those seeding points for the Image Segmentation (IS) model. Another possible and interesting application of organ detection is the efficient retrieval of selected parts of patient scans from radiological databases. In case a physician would like to examine a particular organ, the ability to determine its position and extent automatically means it is not necessary to retrieve the entire scan, but only a small part of it. Also the segmented Region Of Interest (ROI) can serve as an input for higher level algorithms,i.e. an algorithm that looks for tumors or other malicious anomalies.

For this experiment the SCR data set was used. [63] "The SCR database has been established to facilitate comparative studies on segmentation of the lung fields, the heart and the clavicles in standard posterior-anterior chest radiographs." Images of the set were taken from another openly usable data set, the Japanese Society of Radiological Technology (JSRT) database [156] and enhanced with ground truth information. The database contains 247 frontal chest radiographs. The objective of the demonstrative experiment

is to locate and mark the right lung with a deformable model contour in an automatic manner.

Structure

In order to achieve the previously described objectives, the current Ph.D thesis is organized in four chapters. The structure of each of them is briefly introduced as follows.

In Chapter 1 we introduce the preliminary background information of the wide range of techniques required for a proper understanding of the work developed in this thesis. First we introduce the reader to the various techniques to extract characteristic features from images by reviewing some image descriptors commonly adopted in literature. Then, we introduce the basis of ML with a focus on the methods considered along in this dissertation. Finally we review Deformable Models (DMs) focusing on those employed in this work. Then, we introduce Soft Computing (SC) and evolutionary computation, with a particular interest on the Scatter Search (SS) optimisation framework.

Chapter 2 is devoted to the research work carried out to study, enhance and evaluate medical image modality classification. First we introduce a framework for classification of medical images. As the chapter's title suggests we are mainly focusing on modality classification of medical images, as it is an important aspect of the image for medical retrieval [76]. We provide a reference implementation of the framework and show its performance compared to other state-of-the-art classification methods.

In Chapter 3 first we introduce the problem of *structured output prediction* and a conditional model called Structured Output Support Vector Machine (SOSVM) that provides a linear prediction rule for the structured output. Then, we define organ detection as a structured prediction problem. Namely, we present a part-based organ detector, where the structured output is defined by parts and their configuration. We test the model on two different data sets, where both the sought organ and the modality of the images have very different characteristics. In the second part of this

chapter we present a ML based procedure as an alternative to the classical optimisation formulation for DM guidance. To do so, we first study the relevant literature and categorise common existing approaches. Then, we present a general purpose DM framework able to learn the segmentation model from examples extracted from training images. Differently from optimisation-based approaches, the proposed system directly guides the DM evolution through common ML algorithms. To show the feasibility of our alternative approach, we provide a reference implementation tailored to the medical imaging field using Level Sets (LSs) as DM, an ensemble classifier as ML method, the—previously introduced—part-based organ detector as localiser, and a proper set of image visual descriptors. We test it against a large set of state-of-the-art segmentation algorithms over two image data sets with different characteristics.

Finally, in Chapter 4 we draw some conclusions by summarising the most relevant achievements.

CHAPTER 1

Preliminaries

"Everything not saved will be lost."

— *Nintendo* "Quit Screen" *Message*

In this chapter we introduce some concepts that we will use in the rest of this dissertation. First, we review a set of image descriptors widely employed to describe image properties in computer vision. Then, we present ML, a branch of artificial intelligence that concerns the construction and study of systems able to learn from data. Subsequently, we deal with DMs, a group of methods widely employed in computer vision to perform IS. Finally, we introduce SC, an umbrella of techniques to provide inexact solutions to computationally hard tasks. These techniques are tolerant of imprecision, uncertainty, partial truth, and approximation. In particular, we focus on Evolutionary Computing (EC), a set of methods to tackle optimisation problems in an efficient way. The size and diversity of this set of concepts, from the Artificial Intelligence (AI) and computer vision research fields, is due to the high degree of interdisciplinarity of this work.

1.1 Feature descriptors

As a result of the massive generation of content in our modern society, the amount of audio-visual information available in digital format is increasing

considerably. Therefore, it has been necessary to design some systems that allow us to describe the content of several types of multimedia information in order to search and classify them.

In computer vision, Feature Descriptors (FDs) or visual descriptors are descriptions of the visual features of the contents in images or videos. They describe elementary characteristics such as the shape, the colour, the texture or the motion, among others [178, 129]. Since FDs aim at encoding the contents description, they need to express the information variability by being able to discriminate among objects or scenes.

Among image characteristics taken into account by general purpose FD low-level descriptors [80, 52] for IS the most common employed ones are:

- *Colour* is one of the most basic qualities of visual content. Employed FDs describe the colour distribution or the colour relation between sequences or group of images. Among the former, colour histogram is one of the most widely used colour features. It is invariant to image rotation, translation, and viewing axis. The effectiveness of the colour histogram feature depends on the colour space used and the quantisation method [178].

- *Texture* descriptors characterize image regions. Texture is an important feature of a visible surface where repetition or quasi-repetition of a fundamental pattern occurs. Three popular texture descriptions are the co-occurrence matrix [71], Gabor filters [53, 54], and local binary pattern [138].

- *Edge* detection [66] is employed to identify points in a digital image at which the image brightness has discontinuities. Edges are detected to capture important events and changes in properties of the world. Popular techniques are the Sobel [66] and the Canny [32] edge detectors. More recently, the histogram of oriented gradients [50], a FD used in object detection, has been designed as based on gradient directions, that is edge orientations.

- *Shape* contains important semantic information due to the possibility to discriminate objects through their shape. Shape features can be classified into global and local features [178]. Global features are the properties derived from the entire shape. Examples of global features are roundness or circularity, central moments, eccentricity, and major axis orientation. Local features are those derived by partial processing of a shape and do not depend on the entire shape. Examples of local features are size and orientation of consecutive boundary segments, points of curvature, corners, and turning angle.

1.1.1 Haralick features

To characterize texture, Haralick [70] introduced an FD considering texture tonal primitive properties as well as the spatial interrelationships between them. Haralick considered texture-tone as a two-layered structure, the first layer having to do with specifying the local properties which manifest themselves in tonal primitives and the second layer having to do with specifying the organisation among the tonal primitives.

If we define $I(x, y)$, where $0 < x < W - 1$, $0 < y < H - 1$, with W and H width and height of the image in pixels, the gray level of pixel (x, y) and we consider it as a random variable, it is possible to calculate first and second order probability density functions.

1.1.1.1 First order statistics

First order statistics measure the probability of observing a gray level in a random part of the image. They can be calculated from the gray level intensities histogram of the image. First order statistics only depend on the individual pixel values, not on interactions or co-occurrences of pixel values in a neighborhood [168]. For instance, the mean gray values is a first order statistic of the image.

Assuming an image has N_G gray levels, that is, every pixel has a value $i \in [0, N_G - 1]$, it is possible to define the non-normalized image histogram

as

$$P(i) = \frac{N(i)}{N}$$
$$\sum_{i=0}^{N_G-1} P(i) = 1$$

where $N(i)$ is the number of pixels in the image whose value is i and N defines the total number of pixels in the image. Fig. 1.1 shows an example image. Fig. 1.2 depicts the gray level histogram of Fig. 1.1 while the relative numeric representation is shown in Table 1.1.

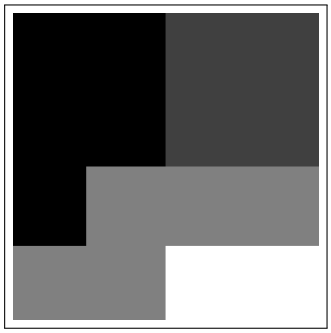


Figure 1.1: Image with 4×4 pixels and 4 gray levels.

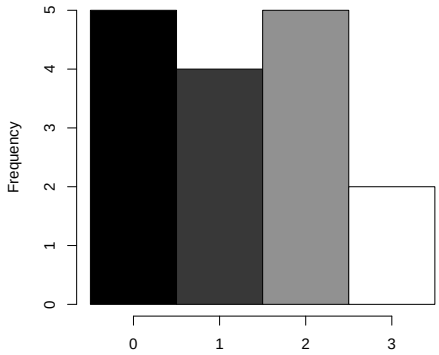


Figure 1.2: Gray level histogram of image in Fig. 1.1.

The first order statistical parameters are detailed as follows.

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

Table 1.1: Numeric form of the image in Fig. 1.1.

Mean gray level. Low values of this parameter imply a dark image, while high values a bright one. It is defined as:

$$\mu = \sum_{i=0}^{N_G-1} i \cdot P(i)$$

Gray level variance. This parameter shows how much the brightness of the image deviates from the mean value.

$$\sigma^2 = \sum_{i=0}^{N_G-1} (i-\mu)^2 \cdot P(i)$$

Coefficient of variation. This parameter is a normalized measure of dispersion of a probability distribution or frequency distribution.

$$cv = \frac{\sigma}{\mu}$$

Skewness. This parameter is a measure of the extent to which a probability distribution of a real-valued random variable "leans" to one side of the mean. The skewness value can be positive or negative, or even undefined.

$$Sk = \frac{1}{\sigma^3} \sum_{i=0}^{N_G-1} (i-\mu)^3 \cdot P(i)$$

Kurtosis. This parameter is any measure of the "peakedness" of the probability distribution of a real-valued random variable. If $Ku > 0$ the obtained histogram is strongly multimodal while for $Ku < 0$ it is similar to a Gaussian. The kurtosis is defined as:

$$Ku = \frac{1}{\sigma^4} \sum_{i=0}^{N_G-1} [(i-\mu)^3 \cdot P(i) - 3]$$

Energy. This parameter gives a measure of the non-uniformity of the histogram.

$$e = \sum_{i=0}^{N_G-1} P(i)^2$$

Entropy. Differently from energy, this parameter gives a measure of the uniformity of the histogram:

$$s = \left| \sum_{i=0}^{N_G-1} P(i) ln[P(i)] \right|, \; with \; 0 \leq s \leq lnN_G$$

1.1.1.2 Second order statistics

Second order statistics are defined as the probability to observe a pair of gray level values at the ends of a dipole of given length, placed in a given position and orientation in the image plane. These are properties of a pair of pixel values [168, 70].

Gray Level Co-occurrence Matrix Gray Level Co-occurrence Matrix (GLCM) is a matrix defined over an image to show the distribution of co-occurrent values at a given offset [69]. More rigorously, a GLCM C is defined over an image I of size $n \times m$ and parametrized by an offset $(\Delta x, \Delta y)$ as:

$$C_{\Delta x,\Delta y}(i,j) = \sum_{p=1}^{n} \sum_{q=1}^{m} \begin{cases} 1, & if \;\; I(p,q) = i \;\; and \;\; I(p+\Delta x, q+\Delta y) = j \\ 0, & else \end{cases}$$

This matrix can be calculated for every image colour depth. However, it is worth to note that a 32-bit GLCM would be of size $2^{32} \times 2^{32}$, making it unfeasible to tackle with.

The parametrisation $(\Delta x, \Delta y)$ makes the GLCM matrix non-invariant to rotation. By choosing an offset vector, an image rotation different from 180 degrees will give rise to a different co-occurrences distribution for the same rotated image. This is an undesirable property in texture analysis. Therefore, the GLCM matrix is calculated using a set of offset vectors covering a 180 degree range (i.e. at 0, 45, 90, and 135 degrees) at the same distance, to obtain a certain degree of rotational invariance.

To show how to compute a GLCM we will employ as reference Fig. 1.1. This image has only four different gray intensities (0, 1, 2, 3). The associated GLCM will be, therefore, a 4×4 matrix.

By fixing the offset vector size to 1 and the rotating direction to right, we will then consider the pairs of neighboring pixels (offset = 1) in which the reference pixel is on the left. This configuration is called $(1, 0)$. Subsequently, every image pixel having a pixel at its right (last column on the right is excluded) is considered as reference pixel, computing, this way, a squared 4×4 matrix. The values on the rows indicate the reference pixels, while values on columns indicate the neighboring pixel. The resulting matrix will count the gray intensities pairs occurrences and is shown in Table 1.2.

	0	1	2	3
0	(0,0)	(0,1)	(0,2)	(0,3)
1	(1,0)	(1,1)	(1,2)	(1,3)
2	(2,0)	(2,1)	(2,2)	(2,3)
3	(3,0)	(3,1)	(3,2)	(3,3)

Table 1.2: Pairs of neighboring intensities considered in the GLCM of Table 1.1.

By applying this algorithm to the example image, we will obtain the matrix in Table 1.3. The values in this matrix are the counts of adjacencies of the gray intensity of the reference pixel (rows) with respect to the neighbors (columns), as shown in Table 1.2.

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

Table 1.3: GLCM given by the offset vector $(1, 0)$. This matrix is not symmetric.

GLCMs usually employed in textural analysis are symmetric and rotation-invariant. The GLCM obtained in the example (Tables 1.2 and 1.3) does

not satisfy this property as only the offset vector $(1, 0)$ has been used to generate it. In order to obtain symmetric and rotation-invariant matrices, it is mandatory to also consider different rotations of the offset vector. Applying 45 degree rotations to the same example implies the following offsets: (1,0), (-1,0), (0,1), (0,-1), (1,1), (-1,-1), (1,-1), (-1,1). The resulting symmetric GLCM is shown in Table 1.4.

16	4	6	0
4	12	5	0
6	5	12	6
0	0	6	2

Table 1.4: GLCM calculated by using all the offset vectors. The resulting matrix is symmetric.

After calculating the GLCM matrix in the way shown earlier, the matrix has to be rescaled, so that it can be considered as a probability density function, that is, it has to satisfy the property $\sum_{i,j=0}^{N_G-1} P_{ij} = 1$. The rescaled version of matrix in Table 1.4 is shown in Table 1.5.

0.1905	0.0476	0.0714	0.0000
0.0476	0.1429	0.0595	0.0000
0.0714	0.0595	0.1429	0.0714
0.0000	0.0000	0.0714	0.0238

Table 1.5: GLCM with rescaled values.

After rescaling the GLCM, it is possible to obtain some second order statistical parameters. They are detailed as follows.

Energy. This parameter is a function of the homogeneity of the texture. Given a high textural homogeneity, the GLCM will have values distributed in a quite uniform way among the cells, with a resulting low energy:

$$E = \sum_{i=0}^{N_G-1} \sum_{j=0}^{N_G-1} P_{ij}^2 \ , \quad N_G^{-1} \leq E \leq 1$$

Entropy. Differently from Energy, this parameter provides high figures when the values of P_{ij} are uniformly distributed over the whole matrix:

$$S = -\sum_{i=0}^{N_G-1}\sum_{j=0}^{N_G-1} P_{ij} ln[P_{ij}] \ , \ \ 0 \leq S \leq lnN_G^2$$

Contrast. The contrast is a measure of the dispersion level of the GLCM. The higher the dispersion, the higher the parameter value, a measure of the second order diagonal moment of the matrix. Therefore, this parameter can be considered as a contrast measure:

$$I = \sum_{i=0}^{N_G-1}\sum_{j=0}^{N_G-1} (i-j)^2 P_{ij}$$

Homogeneity. This parameter gives a measure of the texture homogeneity, a different formulation of the energy formulation.

$$H = \sum_{i=0}^{N_G-1}\sum_{j=0}^{N_G-1} \frac{P_{ij}}{1+|i-j|}$$

1.1.2 Gabor filters

A Gabor filter [53, 54] is a filter used for edge detection. Image analysis by the Gabor functions is similar to perception in the human visual system as it was discovered that the visual cortex response can be modeled by Gabor filters [53, 54, 117]. They have been found particularly appropriate for image representation [105]. A Gabor filter is the product of an elliptical Gaussian envelope and a complex plane wave [107], defined as

$$\psi_{s,d}(x,y) = \psi_{\vec{k}}(\vec{z}) = \frac{||\vec{k}||}{\delta^2} \cdot \exp\left(-\frac{||\vec{k}||^2 \cdot ||\vec{z}||^2}{2\delta^2}\right) \times \left[\exp i\vec{k}\cdot\vec{z} - \exp\left(-\frac{\sigma^2}{2}\right)\right],$$

where $\vec{z} = [x, y]$ is the variable in a spatial domain and $\vec{k}$ is the frequency vector which determines the scale and orientation of Gabor filters, $\vec{k} = k_s e^{i\phi_d}$, where $k_s = k_{\max}/f^s$, $k_{\max} = \pi/2$, $f = 2$, $s = 0, 1, 2, 3, 4$ and $\phi_d = \pi d/8$, for $d = 0, 1, 2, 3, 4, 5, 6, 7$. In the equations the parameter s represents the scale while d represents the orientation. Examples of the real part of Gabor filters

are presented in Fig. 1.3, where Gabor functions (full complex functions) are with five different scales and eight different orientations, making a total of 40 Gabor functions. The number of oscillations under the Gaussian envelope is determined by $\delta = 2\pi$. The term $\exp(-\sigma^2/2)$ is subtracted in order to make the kernel free of a continuous component, and thus insensitive to the average illumination level [107].

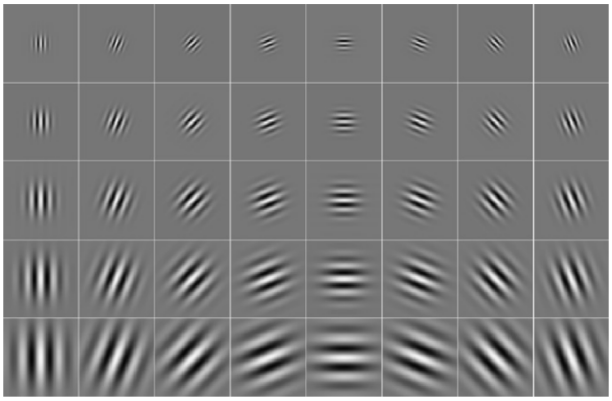


Figure 1.3: The real part of Gabor filters in five different scales and eight different directions (image taken from [107]).

1.1.3 Scale-invariant feature transform

Scale-invariant feature transform (SIFT) [112] has been proposed for extracting distinctive invariant features from images to perform matching of different views of an object or scene. It consists of two main parts: (i) interest point detector and (ii) feature descriptor.

1.1.3.1 Interest point detector

SIFT method uses scale-space Difference of Gaussian (DoG) to detect interest points in images. As for an input image, $I(x, y)$ the scale space is defined as a function, $L(x, y, \sigma)$ produced from the convolution of a variable-scale Gaussian $G(x, y, \sigma)$ with the input image and the difference-of-Gaussian function $D(x, y, \sigma)$ can be computed from the difference of two nearby scales

separated by a multiplicative factor k:

$$\begin{aligned} D(x,y,\sigma) &= [G(x,y,k\sigma) - G(x,y,\sigma)] * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma) \end{aligned} \tag{1.1}$$

Then local maxima and minima of $D(x,y,\sigma)$ are computed based on comparing each sample point to its eight neighbors in current image and nine neighbors in the scale above and below. At this scale, the gradient magnitude, $m(x,y)$, and orientation, $\theta(x,y)$, is computed using pixel differences in Ex.(1.2). Thereafter, an orientation is determined by building a histogram of gradient orientations weighted by the gradient magnitudes from the key-point's neighborhood and it is assigned to each interest point combined with the scale above and provides a scale and rotation invariant coordinate system for the descriptor.

$$\begin{aligned} m(x,y) &= \sqrt{[L(x+1,y) - L(x-1,y)]^2 + [L(x,y+1) - L(x,y-1)]^2} \\ \theta(x,y) &= \tan^{-1} \frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)} \end{aligned} \tag{1.2}$$

After detecting the interest points in each image, the SIFT descriptor computes the gradient magnitude and orientation at each image sample point in a region around the key-point location weighted by a Gaussian window. The coordinates of the descriptor and the gradient orientations are rotated relative to the key-point orientation to achieve orientation invariance.

1.1.3.2 Descriptor

Previous steps found key-point locations at particular scales and assigned orientations to them. This ensured invariance to image location, scale and rotation. Now we want to compute a descriptor vector for each key-point such that the descriptor is highly distinctive and partially invariant to the remaining variations such as illumination, 3D viewpoint, etc. This step is performed on the image closest in scale to the key-point's scale.

First a set of orientation histograms is created on 4×4 pixel neighborhoods with 8 bins each. These histograms are computed from magnitude and orientation values of samples in a 16×16 region around the key- point

such that each histogram contains samples from a 4×4 subregion of the original neighborhood region. The magnitudes are further weighted by a Gaussian function with σ equal to one half the width of the descriptor window. The descriptor then becomes a vector of all the values of these histograms. Since there are $4 \times 4 = 16$ histograms each with 8 bins the vector has 128 elements. This vector is then normalized to unit length in order to enhance invariance to affine changes in illumination. To reduce the effects of non-linear illumination a threshold of 0.2 is applied and the vector is again normalized.

Although the dimension of the descriptor, i.e. 128, seems high, descriptors with lower dimension than this don't perform as well across the range of matching tasks [111]. Longer descriptors continue to do better but not by much and there is an additional danger of increased sensitivity to distortion and occlusion. It is also shown that feature matching accuracy is above 50% for viewpoint changes of up to 50 degrees. Therefore SIFT descriptors are invariant to minor affine changes. To test the distinctiveness of the SIFT descriptors, matching accuracy is also measured against varying number of key- points in the testing database, and it is shown that matching accuracy decreases only very slightly for very large database sizes, thus indicating that SIFT features are highly distinctive.

1.1.4 Local binary patterns

The Local Binary Patterns (LBPs) [138] is a powerful illumination invariant texture primitive. LBPs are a particular case of the Texture Spectrum model proposed in [176].

A histogram of the binary patterns computed over a region is used for texture description. The operator describes each pixel by the relative gray-levels of its neighboring pixels, see Fig. 1.4 for an illustration with 8 neighbors. If the gray-level of the neighboring pixel is higher or equal, the value is set to one, otherwise to zero. The descriptor describes the result over the

neighborhood as a binary number (binary pattern):

$$LBP_{R,N}(x,y) = \sum_{i=0}^{N-1} s(n-n_c)2^i, \quad s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

where n_c corresponds to the gray-level of the center pixel of a local neighborhood and n_i to the gray-levels of N equally spaced pixels on a circle of radius R. Since correlation between pixels decreases with distance, a lot of the texture information can be obtained from local neighborhoods. Thus, the radius R is usually kept small. In practice, Ex. 1.3 means that the signs of the differences in a neighborhood are interpreted as a N-bit binary number, resulting in $2N$ distinct values for the binary pattern. From above, it's easy to figure out that the LBP has several properties that favor its usage in interest region description. The features have proven to be robust against illumination changes, they are very fast to compute, and do not require many parameters to be set.

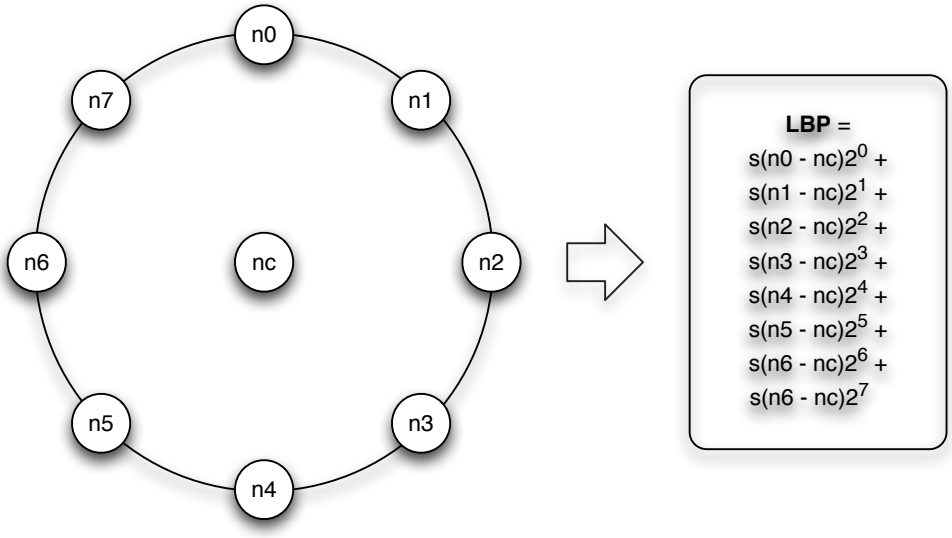


Figure 1.4: LBP feature for a neighborhood of 8 pixels.

1.1.5 Histogram of oriented gradients

Histogram of Oriented Gradients (HOG) [50] are feature descriptors used in computer vision and image processing for the purpose of object detection. The method counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms [61, 60], SIFT descriptors and shape contexts [15], but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalisation for improved accuracy.

The essential thought behind the HOG descriptors is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The HOG descriptor maintains a few key advantages over other descriptor methods. Since the HOG descriptor operates on localized cells, the method upholds invariance to geometric and photometric transformations, except for object orientation. Such changes would only appear in larger spatial regions.

The implementation of these descriptors can be achieved by dividing the image into small connected regions, called cells, and compiling a histogram of gradient directions or edge orientations for the pixels within each cell. The combination of these histograms then represents the descriptor. For improved accuracy, the local histograms can be contrast normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalisation results in better invariance to changes in illumination or shadowing.

Finally, it has further been determined that when HOG is combined with the LBP, it improves the detection performance considerably on some data sets [177]. The calculation of the algorithm is detailed as follows.

1. **Gradient computation**: as in many feature detectors in image pre-processing the first step is to ensure normalized colour and gamma values. However, the authors in [50] point out this step can be omitted in HOG descriptor computation, as the ensuing descriptor normalisation essentially achieves the same result. Image pre-processing thus

provides little impact on performance. Instead, the first step of calculation is the computation of the gradient values. The most common method is to simply apply the 1-D centered, point discrete derivative mask in one or both of the horizontal and vertical directions. Specifically, this method requires filtering the colour or intensity data of the image with the following filter kernels:

$$[-1, 0, 1] \text{ and } [-1, 0, 1]^\intercal. \tag{1.4}$$

2. **Orientation binning**: The second step of calculation involves creating the cell histograms. Each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. The cells themselves can either be rectangular or radial in shape, and the histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is *unsigned* or *signed*. Dalal and Triggs [50] found that unsigned gradients used in conjunction with 9 histogram channels performed best in their human detection experiments. As for the vote weight, pixel contribution can either be the gradient magnitude itself or some function of the magnitude. In actual tests the gradient magnitude itself generally produces the best results.

3. **Descriptor blocks**: In order to account for changes in illumination and contrast, the gradient strengths must be locally normalized, which requires grouping the cells together into larger, spatially connected blocks. The HOG descriptor is then the vector of the components of the normalized cell histograms from all of the block regions. These blocks typically overlap, meaning that each cell contributes more than once to the final descriptor. Two main block geometries exist: *a*) rectangular R-HOG blocks and *b*) circular C-HOG blocks. R-HOG blocks are generally square grids, represented by three parameters: (i) the number of cells per block, (ii) the number of pixels per cell, and (iii) the number of channels per cell histogram. In the human detection experiment performed in [50], the optimal parameters were

found to be 3×3 cell blocks of 6×6 pixel cells with 9 histogram channels. Moreover, it was found that some minor improvement in performance could be gained by applying a Gaussian spatial window within each block before tabulating histogram votes in order to weight pixels around the edge of the blocks less. The R-HOG blocks appear quite similar to the SIFT descriptors. However, despite their similar formation, R-HOG blocks are computed in dense grids at some single scale without orientation alignment, whereas SIFT descriptors are computed at sparse, scale-invariant key image points and are rotated to align orientation. In addition, the R-HOG blocks are used in conjunction to encode spatial form information, while SIFT descriptors are used singly.

C-HOG blocks can be found in two variants: (i) those with a single, central cell, and (ii) those with an angularly divided central cell. In addition, these C-HOG blocks can be described with four parameters: (i) the number of angular bins, (ii) the number of radial bins, (iii) the radius of the center bin, and (iv) the expansion factor for the radius of additional radial bins.

4. **Block normalisation**: The authors explored four different methods for block normalisation in [50]. Let v be the non-normalized vector containing all histograms in a given block, $\|v\|_k$ be its k-norm for $k = 1, 2$, and e be some small constant (the exact value, hopefully, is unimportant). Then the normalisation factor can be one of the following:

   - L2-norm: $f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$,
   - L2-hys: L2-norm followed by clipping (limiting the maximum values of v to 0.2) and renormalizing,
   - L1-norm: $f = \frac{v}{\|v\|_1 + e}$, and
   - L1-sqrt: $f = \sqrt{\frac{v}{\|v\|_1 + e}}$.

   In addition, the scheme L2-hys can be computed by first taking the L2-norm, clipping the result, and then renormalizing. In the exper-

iments developed, it was found the L2-hys, L2-norm, and L1-sqrt schemes provided similar performance, while the L1-norm provided slightly less reliable performance. Anyway, all four methods showed very significant improvement over the non-normalized data.

1.2 Machine learning

ML is a branch of artificial intelligence dealing with the construction and study of systems that can learn from data. That is, ML aims at programming computers to optimize a performance criterion using examples or past experience [4]. We need learning in cases where we cannot directly write a computer program to solve a given problem, but need numerical data or experience. Mitchell provided a more formal definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [126].

The core of ML deals with knowledge representation and generalisation. Representation of data instances and functions evaluated on these instances are part of all ML systems.

Generalisation, in this context, is the ability of an algorithm to perform accurately on new, unseen examples after having been trained on a learning data set. The core objective of a learner is to generalize from its experience. The training examples come from some generally unknown probability distribution and the learner has to extract something more general from them, something about that distribution, that allows it to produce useful predictions in new cases.

Based on the type of the considered inputs and on the desired outcome, there are mainly three different types of ML methods:

- *Supervised learning* generates a function that maps inputs to desired outputs (also called labels, because they are often provided by human experts labeling the training examples). The supervised learning algorithm attempts to generalize a function or mapping from inputs to

outputs which can then be used to speculatively generate an output for previously unseen inputs.

- *Unsupervised learning* algorithms operate on unlabeled examples, i.e., input where the desired output is unknown. They aim at finding hidden structures in unlabeled data by analyzing the regularities in the input. In statistics, this is called density estimation. A very extended method for density estimation is clustering where the aim is to find clusters or groupings of the inputs.

- *Reinforcement learning* is used in those applications where the output of the system is a sequence of actions. In such a case, a single action is not important; what is important is the policy, that is, the sequence of correct actions to reach the goal. Therefore, reinforcement learning is concerned with how intelligent agents ought to take actions in an environment so as to maximize some notion of cumulative reward. The agent should be able to assess the goodness of policies and learn from past good action sequences to be able to generate a policy.

1.2.1 Applications

ML has been applied to a large set of applications from different fields and research areas, including computer vision, engineering, mathematics, physics, neuroscience, and cognitive science. Among notable applications are natural language processing, IS, search engines, credit card fraud detection, stock market analysis, recommender systems, information retrieval, robot locomotion, and DNA sequence mining. In most of the cases, however, these problems are reduced to two categories: classification and regression. They are defined as follows.

Classification is probably the most frequently studied problem in ML and it has led to a large number of important algorithmic and theoretic developments over the past century. In its binary form it reduces to the question: given a pattern x drawn from a domain $\mathcal{X}$, estimate which value

an associated binary random variable $y \in -1, 1$ will assume. More formally, classification is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. For instance, given pictures of apples and oranges, we might want to state whether the object in question is an apple or an orange. An observation, (also called instance, pattern, or example) is described by its features. These are the characteristics of the examples for a given problem. Thus, the input to a classification task can be viewed as a two-dimensional matrix, whose axes are the examples and the features. The classification problem can be split in two categories: binary classification and multiclass classification. In binary classification only two classes are involved, whereas multiclass classification involves assigning an object to one of several classes. Since many classification methods have been developed specifically for binary classification, multiclass classification often requires the combined use of multiple binary classifiers.

Regression is a problem where the output is a number, rather than a class. In this case the goal is to estimate a real-valued variable $y \in \mathbb{R}$ given a pattern x. For instance, we might want to estimate the value of a stock the next day, the yield of a semiconductor fabrication given the current process, the iron content of ore given mass spectroscopy measurements, or the heart rate of an athlete, given accelerometer data. One of the key issues in which regression problems differ from each other is the choice of a loss. For instance, when estimating stock values our loss for a put option will be decidedly one-sided. On the other hand, a hobby athlete might only care that our estimate of the heart rate matches the actual on average.

1.2.2 Supervised learning

1.2.2.1 Support vector machines

Support Vector Machines (SVMs) are based on Structural Risk Minimisation principle [172] from computational learning theory. It has been suc-

cessfully applied in various classification tasks like image recognition.

1.2.2.2 Linear support vector machines

The separable case The simplest model of SVMs is the so-called *maximum margin classifier*. It works only for data which are linearly separable, hence it cannot be used in many real-world situations.

Let $\mathcal{S}_{train} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be a set of training examples. The examples are represented as feature vectors $\mathbf{x}_i \in \mathcal{X}$ obtained from a feature space $\mathcal{X}$ and the labels $y_i \in \{-1, 1\}$ are assumed to indicate whether an image is assigned to a category or not.

Suppose that there exists a hyperplane which separates the positive examples ($\mathcal{S}^+ = \{(\mathbf{x}_i, y_i) \mid y_i = 1\}$). from the negative one ($\mathcal{S}^- = \{(\mathbf{x}_i, y_i) \mid y_i = -1\}$). This is called the *separating hyperplane*. The points $\mathbf{x}_i$ which lie on the hyperplane satisfy $\mathbf{w} \cdot \mathbf{x}_i + b = 0$, where $\mathbf{w}$ is normal to the hyperplane and $\frac{|b|}{||\mathbf{w}||_2}$ is the perpendicular distance from the hyperplane to the origin.

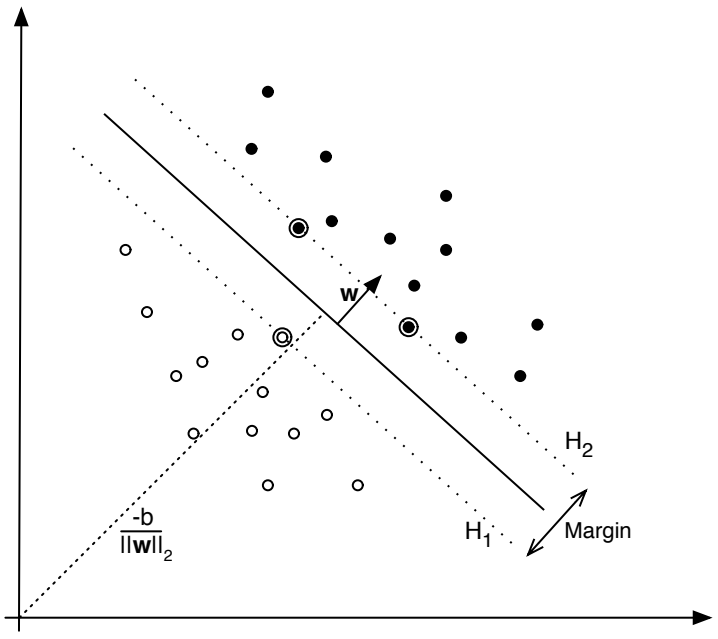


Figure 1.5: Linear separating hyperplanes for the separable case. The support vectors are circled.

Let d_+ (d_-) be the shortest distance from the separating hyperplane to the closest positive (negative) example. The *margin* (γ) of the separating hyperplane is defined to be $d_+ + d_-$. In linearly separable case, the SVM simply looks for the separating hyperplane that has the largest margin. More formally,

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq 1 \quad \text{for } \{(\mathbf{x}_i, y_i) \mid y_i = 1\} \tag{1.5}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \text{for } \{(\mathbf{x}_i, y_i) \mid y_i = -1\} \tag{1.6}$$

These can be combined into one set of inequalities:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall\ (\mathbf{x}_i, y_i) \in \mathcal{S}_{train} \tag{1.7}$$

The examples for which the equality Eq. (1.5) holds lie on the $H_1 : \mathbf{x}_i \cdot \mathbf{w} + b = 1$ hyperplane, with normal $\mathbf{w}$ and perpendicular distance of $\frac{|1-b|}{||\mathbf{w}||_2}$ from the origin. Analogously, the examples for which Ex. (1.6) holds lie on the hyperplane $H_2 : \mathbf{x}_i \cdot \mathbf{w} + b = -1$, again with normal $\mathbf{w}$ and perpendicular distance of $\frac{|-1-b|}{||\mathbf{w}||_2}$ from the origin. Thus $d_+ = d_- = \frac{1}{||\mathbf{w}||_2}$ and the margin is $\gamma = \frac{2}{||\mathbf{w}||_2}$. So in order to maximize the margin, one needs to minimize $||\mathbf{w}||_2^2$. More formally:

$$\begin{aligned} \text{Minimize :} \quad & ||\mathbf{w}||_2^2 \\ \text{subject to :} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i - b) - 1 \geq 0, \quad \forall\ (\mathbf{x}_i, y_i) \in \mathcal{S}_{train} \end{aligned} \tag{1.8}$$

Since this optimisation problem is rather hard to handle numerically, Lagrange multipliers are used to translate the problem into an equivalent Quadratic Programming (QP) problem [172]. Positive Lagrange multipliers $\alpha_i,\ i = 1, \ldots, |\mathcal{S}_{train}|$ are introduced for each of the inequality constraints of Ex. (1.7). The rule to form the Lagrangian:

- for constraints of form $c_i \geq 0$, the constraint inequalities are multiplied by positive Lagrange multipliers and subtracted from the objective function,

- for equality constraints, the Lagrange multipliers are unconstrained.

Applying these rules to the Ex. (1.7) gives the Lagrangian:

$$L_P = \frac{1}{2}||\mathbf{w}||_2^2 - \sum_{i=1}^{|\mathcal{S}_{train}|} \alpha_i y_i(\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{|\mathcal{S}_{train}|} \alpha_i \tag{1.9}$$

Minimizing L_p with respect to $\mathbf{w}, b$ and maximizing it with respect to $\boldsymbol{\alpha}$ yields the solution for the problem above and can be found at an extremum point where

$$\frac{\partial L_p}{\partial b} = 0 \quad and \quad \frac{\partial L_p}{\partial \mathbf{w}} = 0$$

which transform into

$$\sum_{i=1}^{|\mathcal{S}_{train}|} \alpha_i y_i = 0 \tag{1.10}$$

$$\mathbf{w} = \sum_{i=1}^{|\mathcal{S}_{train}|} \alpha_i y_i \mathbf{x}_i \tag{1.11}$$

for L_p. The dual QP problem can be obtained by plugging these constraints into Ex. (1.9) which gives:

$$L_D = \sum_{i=1}^{|\mathcal{S}_{train}|} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{1.12}$$

Therefore the support vector training for the separable, linear case amounts to maximise L_D with respect to the α_i, subject to constraints (1.11) and positivity of the α_i, with solution given by (1.10).

Note that the formulation of this optimisation problem replaces $\mathbf{w}$ with the product of the Lagrangian multipliers and the given training examples $\sum_{i=1}^{|\mathcal{S}_{train}|} \alpha_i y_i \mathbf{x}_i$. Hence the separating hyperplane can be defined only through the given training examples $\mathbf{x}_i$.

In the solution, the points for which $\alpha_i > 0$ are called *support vectors*, and lie on one of the hyperplanes H_1, H_2. All other training points have $\alpha_i = 0$ and lie either on H_1 or H_2 (such that the equality in Ex. (1.7) holds), or on that side of H_1 or H_2 such that the strict inequality in Ex. (1.7)

holds. For these machines, the support vectors are the critical elements of the training set. If all other training points were removed or moved around, but so as not to cross H_1 or H_2, and training was repeated, the same separating hyperplane would be found.

After the learning process, to determine if a given test example is whether a positive or a negative one, simply requires to evaluate the inner product of the test example with the obtained support vectors:

$$h(\mathbf{x}_j) = sign(\mathbf{w} \cdot \mathbf{x}_j + b)$$

The non-separable case The former formulation holds only if the training set of examples are linearly separable, i.e. there were no errors in the set of training examples. Note that this is rather a harsh assumption.

In case of a linearly non-separable training set the above defined dual optimisation problem does not lead to a feasible solution (the objective function i.e. L_D will grow arbitrary large). Even if a solution is found this might not be the solution minimising the risk on the given training data with respect to some loss function [171].

One solution for extending the previous ideas to handle linearly non-separable data, is to relax the constraints (1.5) and (1.6), but only when necessary. This can be done by introducing positive *slack variables* $\xi_i, i = 1, \ldots, |\mathcal{S}_{train}|$, in the constraints [44]. Including these variables in the Eq. (1.7) gives:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1 - \xi_i \quad \xi_i \geq 0, \ \forall \ (\mathbf{x}_i, y_i) \in S_{train} \tag{1.13}$$

As it can be seen, for an error to occur, the corresponding ξ_i must exceed unity, so $\sum_i \xi_i$ is an upper bound on the number of training errors. The logical way for addressing the extra cost for the errors is to change the objective function to be minimized from $\frac{1}{2}||\mathbf{w}||_2^2$ to

$$\frac{1}{2}||\mathbf{w}||_2^2 + C\left(\sum_{i=1}^{\mathcal{S}_{train}} \xi_i\right)^k$$

where $C > 0$ is a parameter to be chosen by the user, a larger C corresponding to assigning a higher penalty to errors. This is a convex programming

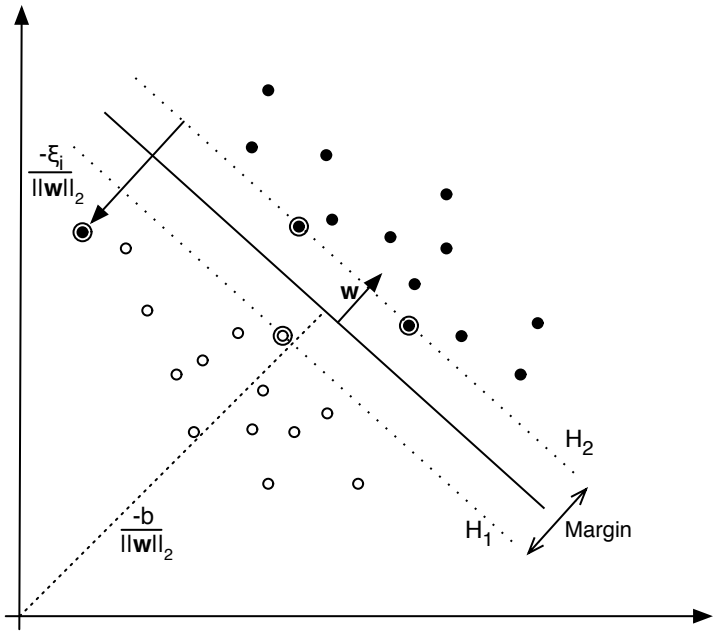


Figure 1.6: Linear separating hyperplanes for the non-separable case.

problem for any k. For $k \in \{1, 2\}$ it is also a QP problem. The $k = 1$ has the further advantage that neither the ξ_i, nor their Lagrange multipliers appear in the dual problem [31]:

$$\text{Maximize :} \qquad L_D = \sum_{i=1}^{|\mathcal{S}_{train}|} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{1.14}$$

$$\text{subject to :} \qquad 0 \leq \alpha_i \leq C, \text{ and } \sum_i \alpha_i y_i = 0 \tag{1.15}$$

The solution is again given by

$$\mathbf{w} = \sum_{i=1}^{N_S} \alpha_i y_i \mathbf{x}_i$$

where N_S is the number of support vectors. The only difference from the linearly separable case is, that the α_i now have an upper bound of C.

1.2.2.3 Non-linear support vector machines

Although it was not stated explicitly, but in the methods introduced above the assumption was that the decision function is a linear function of the data (see Ex. (1.11); the weight vector is the linear combination of the training examples). [27] showed that this constraint may be omitted by applying a rather old trick [1]. That is, by mapping the linearly inseparable training data into a higher dimensional space it becomes separable.

It is important to see that the data in the training problem (Ex.(1.14) and (1.15)), only appears in the form of a dot product $(\mathbf{x}_i \cdot \mathbf{x}_j)$. Suppose that there is a function that maps a vector into a—possibly infinite dimensional—Hilbert space $\mathcal{H}$.

$$\Phi : \mathbb{R}^n \to \mathcal{H}$$

By employing this function on the original $\mathcal{S}_{\text{train}}$ training set, one will get the $\mathcal{S}'_{train} = \{(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_n), y_n)\}$ set, where the data is mapped into the given Hilbert space.

It is obvious that training algorithm would only depend on the data through dot products in $\mathcal{H}$ $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, i.e. on functions of the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Thus, if one can define a so-called *kernel function*, that calculates the dot product in the mapped space ($K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$), one would never need to explicitly even know what Φ is. As stated in [172] the kernel function must satisfy Mercer's condition, otherwise the corresponding QP may have no solution.

After finding an appropriate kernel function K, if one replaces $\mathbf{x}_i \cdot \mathbf{x}_j$ by $K(\mathbf{x}_i, \mathbf{x}_j)$ everywhere in the training algorithm, the algorithm will happily produce a SVM which lives in an infinite dimensional space, and furthermore do so in roughly the same amount of time it would take to train on the un-mapped data. All the considerations of the previous sections hold, since still doing a linear separation, but in a different space.

Figure 1.7 illustrates a two-dimensional classification example mapped to the three-dimensional space by $\Phi(x_1, x_2) = \langle x_1^2, \sqrt{2}x_1 * x_2, x_2^2 \rangle$. As it shows, the mapping makes the examples linearly separable.

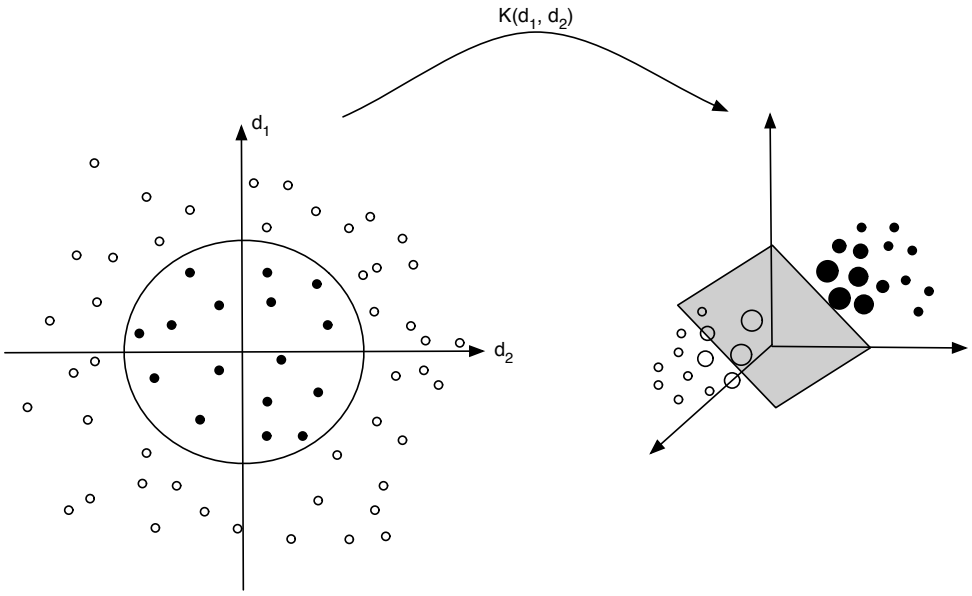


Figure 1.7: Transformation from two-dimensional to three-dimensional space by using the kernel $K(x_1, x_2) = \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$. The shaded plane right shows the separating plane in the three dimensional space.

The solution of the above defined SVM will give $\mathbf{w}$ that lives in $\mathcal{H}$. Since testing an arbitrary example $\mathbf{x}_j$ only depends on the dot product of the example vector and the support vectors, one can—again—avoid calculating $\Phi(\mathbf{x}_i)$ and use the kernel function instead:

$$h(\mathbf{x}_j) = sign\left(\sum_{i=1}^{N_S} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b\right) \tag{1.16}$$

Some commonly used kernel functions:

$$\begin{aligned} K_{poly}(\mathbf{x}_i, \mathbf{x}_j) &= (\mathbf{x}_i \cdot \mathbf{x}_j + b)^d \\ K_{rbf}(\mathbf{x}_i, \mathbf{x}_j) &= \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{c}\right) \\ K_{sigmoid}(\mathbf{x}_i, \mathbf{x}_j) &= \tanh(\gamma(\mathbf{x}_i \cdot \mathbf{x}_j) + b) \end{aligned}$$

where the sigmoid function only satisfies the Mercer condition for certain b values.

The training and testing of a SVM depends on the data only through the kernel function.

The classification of a test example consists of the evaluation of Ex. (1.16). This requires $O(MN_s)$ operations, where M is the number of operations required to evaluate the kernel. For example, for dot product and RBF kernels M is $O(d_L)$, the dimension of the data vectors.

In order the increase the speed and decrease memory requirements of a SVM's training three different approaches have been proposed:

- *Chunking*: it starts with a small, arbitrary subset of the data for training. The rest of the training data is tested on the resulting classifier. The test results are sorted by the margin of the training examples on the wrong side. The first N of these and the already found support vectors are taken for the next training step. The training stops at the upper bound of the training error. This method requires the number of support vectors N_S to be small enough in order that a Hessian —$N_S \times N_S$—matrix fits into the memory.

- *Decomposition*: it was introduced in [140]. Only a small portion of the training data is used for training in a given time. Moreover, only a subset of the support vectors have to be in the actual "working set".

- *Sequential Minimal Optimisation (SMO)* was introduced in [147]. It is based on the convergence theorem provided in [140]. The optimisation problem is split into simple, analytically solvable problems, that involves only two Lagrangian multipliers. Thus, the SMO algorithm consists of two steps: (i) using a heuristic to choose the two Lagrangian multipliers (ii) analytically solving the optimisation problem of the chosen multipliers and updating SVM. The advantage of SMO lies in the fact, that numerical QP is avoided entirely. Plus, SMO requires no matrix storage since only two Lagrangian classifiers are solved at a time.

1.2.2.4 Random forest

Among ensemble learning decision trees, Random Forest (RF) [30] is endowed with some attractive capabilities. It is introduced in this section

since it is employed as an important part of the proposal explained in Section 3.4 of this dissertation. It combines bagging and the random selection of features to construct a collection of decision trees with controlled variation. RF is employed to construct a prediction rule in a supervised learning problem (classification or regression) and to assess and rank variables with respect to their ability to predict the response. The latter is done by considering the so-called variable importance measures that are automatically computed for each predictor within the RF algorithm [29].

Each tree of the ensemble is grown by RF as follows:

- Let N be the number of examples in the training set. n instances are sampled at random with replacement (that is, a bootstrap sample), from the original data. This sample will be the training set for growing the tree.

- If there are M input variables (features), a number $m \ll M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.

- Each tree is grown to the largest extent possible. There is no pruning.

To classify a new object from an input vector, the input vector is put down each of the trees in the forest. Each tree gives a classification and we say the tree votes for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Lowering the value of m reduces both the correlation among trees and the strength of each tree, while raising it increases both. The optimal range of m is application dependent, but it is usually quite wide. This is the only adjustable parameter to which random forest is somewhat sensitive.

RF is particularly attractive for a large set of notable features. First of all, its accuracy is comparable with the state-of-the-art learning algorithms while running efficiently on large data bases. It can handle thousands of input variables without variable deletion, even in the case of $M < N$ [29]. Moreover, RF has an effective method for estimating missing data and

maintains accuracy when a large proportion of the data are missing. It gives estimates of what variables are important in the classification and generates an internal unbiased estimate of the generalisation error as the forest building progresses. Finally, it has methods for balancing error in class population unbalanced data sets.

1.2.3 Unsupervised learning

1.2.3.1 Clustering

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimisation problem. The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimisation that involves trial and failure. It will often be necessary to modify data preprocessing and model parameters until the result achieves the desired properties.

K-means Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where each observation is a d-dimensional real vector, *k-means* clustering aims to partition

the n observations into k sets ($k \leq n$) $\mathbf{S} = S_1, S_2, \ldots, S_k$ so as to minimize the within-cluster sum of squares:

$$\underset{\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in \mathcal{S}_i} ||\mathbf{x}_j - \mu_i||^2 \tag{1.17}$$

where μ_i is the mean of points in S_i.

1.3 Deformable models

The Shi level set In [154] and later in [155], the authors presented a complete and practical algorithm for the approximation of LS-based curve evolution suitable for real-time implementation. This is a two-cycle algorithm to approximate LS-based curve evolution without the need of solving Partial Differential Equations (PDEs). The evolution is divided into two cycles: one cycle for the data-dependent term and a second cycle for the smoothness regularisation, derived from a Gaussian filtering process. In both cycles, the evolution is developed through a simple element switching mechanism between two linked lists, that implicitly represents the curve using an integer valued level-set function. This approach leads to very significant computation speedups compared to exact PDE-based approaches. Since the Shi model is different from the standard LS (as it does not make use of PDEs) and it is employed in our proposal described in Section 3.4, the remaining of this section deals with relevant implementation details of this LS model. In particular, Algorithm 1 shows the complete Shi LS Fast Two-Cycle (FTC) algorithm.

In the LS method, a curve C is represented implicitly as the zero LS of a function ϕ defined over a regular grid D of size $M_1 \times M_2 \times \cdots M_k$, with k being the number of dimensions of the images at hand. The set of grid points enclosed by C is called Ω. The list of inside neighboring grid points L_{in} and outside neighboring grid points L_{out} for the object region Ω is:

$$L_{in} = \{x | x \in \Omega \text{ and } \exists y \in N(x) \text{ s.t. } y \in D \setminus \Omega\} \tag{1.18}$$

$$L_{out} = \{x | x \in D \setminus \Omega \text{ and } \exists y \in N(x) \text{ s.t. } y \in \Omega\} \tag{1.19}$$

Step 1: Initialize arrays $\hat{\phi}$, $\hat{F}_d$, $\hat{F_{\text{int}}}$ and lists L_{out} and L_{in};
Step 2 (**cycle one: data dependent evolution**):
for $i = 1 : N_a$ **do**
    Compute the data dependent speed F_d for each point in L_{out} and L_{in} and store its sign in $\hat{F}_d$;
    Outward evolution. For each point $x \in L_{\text{out}}$, switch_in(x) if $\hat{F}_d(x) > 0$;
    Eliminate redundant points in L_{in}. For each point $x \in L_{\text{in}}$, if $\forall y \in N(x)$, $\hat{\phi}(y) < 0$, delete x from L_{in}, and set $\hat{\phi}(x) = -3$;
    Inward evolution. For each point $x \in L_{\text{in}}$, switch_out(x) if $\hat{F}_d(x) < 0$;
    Eliminate redundant points in L_{out}. For each point $x \in L_{\text{out}}$, if $\forall y \in N(x)$, $\hat{\phi}(y) > 0$, delete x from L_{out}, and set $\hat{\phi}(x) = 3$;
    Check the stopping conditions. If it is satisfied, go to Step 3; otherwise continue this cycle;
end
Step 3 (**cycle two: smoothing via Gaussian filtering**):
for $i = 1 : N_s$ **do**
    Compute the smoothing speed $\hat{F_{\text{int}}}$ for each point in L_{out} and L_{in};
    Outward evolution. For each point $x \in L_{\text{out}}$, switch_in(x) if $\hat{F_{\text{int}}}(x) > 0$;
    Eliminate redundant points in L_{in}. For each point $x \in L_{\text{in}}$, if $\forall y \in N(x)$, $\hat{\phi}(y) < 0$, delete x from L_{in}, and set $\hat{\phi}(x) = -3$;
    Inward evolution. For each point $x \in L_{\text{in}}$, switch_out(x) if $\hat{F_{\text{int}}}(x) < 0$;
    Eliminate redundant points in L_{out}. For each point $x \in L_{\text{out}}$, if $\forall y \in N(x)$, $\hat{\phi}(y) > 0$, delete x from L_{out}, and set $\hat{\phi}(x) = 3$;
end
Step 4: If stopping condition not satisfied in cycle one, go to Step 2;

Algorithm 1: Shi LS FTC algorithm.

where $N(x)$ is a discrete neighborhood of x.

The data structures used by the algorithm consist of:

- an integer array $\hat{\phi}$ for approximating the LS function;
- an integer array $\hat{F}$ for the speed function;
- two lists of grid points adjacent to the evolving curve C: L_{in} and L_{out}.

The grid points inside C but not in L_{in} are defined interior points while those outside C but not in L_{out} are defined exterior points. The function $\hat{\phi}$ locally approximates the LS signed distance function and is defined as:

$$\hat{\phi}(x) = \begin{cases} +3, & \text{if } x \text{ is an exterior point} \\ +1, & \text{if } x \in L_{out} \\ -1, & \text{if } x \in L_{in} \\ -3, & \text{if } x \text{ is an interior point.} \end{cases}$$

Only the sign of the evolution speed is used in the algorithm. Two basic procedures are necessary to perform the evolution. On the one hand, `switch_in()` moves the boundary outward by one pixel. For a point $x \in L_{out}$, it is defined as follows. The first thing to be done is to switch x from L_{out} to L_{in}. With $x \in L_{in}$ now, all its neighbors that were exterior points become neighboring grid points and are added to L_{out} in the second step. By applying a `switch_in()` procedure to any point in L_{out}, the boundary is moved outward by one grid point in that location.

Similarly, the procedure `switch_out()` effectively moves the boundary inward by one pixel. By applying a `switch_out()` procedure to an inside neighboring grid point $x \in L_{in}$, we move the boundary inward by one grid point at that location.

At every iteration, first the speed at each point in L_{out} and L_{in} is calculated and the sign of the speed is stored in the array $\hat{F}$. After that, the two lists are scanned through sequentially to evolve the curve. The list L_{out} is scanned first and a `switch_in()` is applied at a point if $\hat{F} > 0$. This scan takes care of those parts of the curve with positive speed and moves them

outward by one grid point. After this scan, some of the points in L_{in} become interior points due to the newly added inside neighboring grid points, so they are deleted from L_{in}. Then, the list L_{in} is scanned through and `switch_out()` is applied for a point with $\hat{F} < 0$. This scan moves those parts of the curve with negative speed inward by one grid point. Similarly, points that have become exterior points are deleted from L_{out} after this scan. After a scan through both lists, the following stopping conditions are checked:
(a) The speed at all the neighboring grid points satisfies

$$\hat{F}(x) \leq 0 \quad \forall x \in L_{out}$$
$$\hat{F}(x) \geq 0 \quad \forall x \in L_{in}$$

(b) A pre-specified maximum number of iteration is reached.

In addition to the steps described so far, the algorithm also performs a curvature dependent smoothness regularisation cycle, by means of a Gaussian filtering process. This second cycle approximates curvature-based evolution, but avoids the need for expensive computations. The Gaussian filter is of the form

$$G(x) = \frac{1}{2\pi\sigma^2}\exp\bigg(-\frac{1}{2\sigma^2}|x|^2\bigg).$$

Instead of calculating the force F using image cues, in this case the force depends on the filtering and is defined as follows:

$$\hat{F}(x) = \begin{cases} -1, & \text{if } G \otimes H(-\hat{\phi})(x) < \frac{1}{2} \\ 0, & \text{otherwise,} \end{cases}$$

where H is the Heaviside function and $\otimes$ is the convolution operator [66].

The two cycles described so far are then repeated to evolve the curve towards the object boundary.

1.3.0.1 Speed functions

In this section, we provide a brief overview of five of the most representative speed functions used by geometric deformable contours.

Caselles et al. and Malladi et al. The geometric deformable contour formulation, proposed by Caselles et al. [33] and Malladi et al. [116], takes the following form:

$$\frac{\partial \phi}{\partial t} = c(k + V_0)|\nabla \phi|,$$

where

$$c = \frac{1}{1 + |\nabla (G_\sigma * I|}.$$

Positive V_0 shrinks the curve, and negative V_0 expands the curve. The curve evolution is coupled with the image data through a multiplicative stopping term c. This scheme can work well for objects that have good contrast. However, when the object boundary is indistinct or has gaps, the geometric deformable contour may leak out because the multiplicative term only slows down the curve near the boundary rather than completely stopping the curve. Once the curve passes the boundary, it will not be pulled back to recover the correct boundary.

Caselles et al. and Kichenassamy et al. To remedy the problem introduced in the latter section, Caselles et al. [34] and Kichenassamy et al. [181, 92] used an energy minimisation formulation to design the speed function. This leads to the following geometric deformable contour formulation:

$$\frac{\partial \phi}{\partial t} = c(k + V_0)|\nabla \phi| + \nabla c \cdot \nabla \phi.$$

Note that the resulting speed function has an extra stopping term $\nabla c \cdot \nabla \phi$ that can pull back the contour if it passes the boundary. This term behaves in similar fashion to the Gaussian potential force in the parametric formulation. However, the latter formulation can still generate curves that pass through boundary gaps.

Siddiqi et al. Siddiqi et al. [157] partially address the problem described in the last section by altering the constant speed term through energy minimisation, leading to the following geometric deformable contour:

$$\frac{\partial \phi}{\partial t} = \lambda(ck|\nabla \phi| + \nabla c \cdot \nabla \phi) + (c + \frac{1}{2}\mathbf{X} \cdot \nabla c)|\nabla \phi|.$$

In this case, the constant speed term V_0 is replaced by the second term, and the term $1/2 \cdot \mathbf{X} \cdot \nabla c$ provides additional stopping power that can prevent the geometrical contour from leaking through small boundary gaps. The second term can be used alone as the speed function for shape recovery as well. Although this model is robust to small gaps, large boundary gaps can still cause problems.

CV model "Active Contours Without Edges" [36] is a classic geometric and implicit method by Chan and Vese that tries to solve the Mumford-Shah functional [130]. This algorithm was designed to detect objects whose boundaries are not necessarily defined by gray level gradients. Indeed, it ignores edges completely, converting CV in a region-based method. The idea is to separate the image into two regions having homogeneous intensity values. More formally, the process minimizes the energy functional shown in Ex. 1.20. The functional is used to evolve a LS representing the contour C, by using the conventional variational calculus approach.

$$\begin{aligned} E(C) = \mu \cdot \text{Length}(C) + \nu \cdot \text{Area}(C) \\ + \lambda_1 \int_C |I(x,y) - \overline{I}_C|^2 dxdy \\ + \lambda_2 \int_{\Omega \setminus C} |I(x,y) - \overline{I}_{\Omega \setminus C}|^2 dxdy \end{aligned} \tag{1.20}$$

In Ex.1.20, I is the intensity value of the image to be segmented and $\overline{I}$ is its average value. Along with the length of C and its area, there are a third and fourth term representing the variance of the intensity level (i.e., the homogeneity) inside and outside C, respectively. Each term has a weight that determines its influence on the total energy, so that, for instance, the smaller μ, the more the length of the curve can increase without penalizing the minimisation.

Geodesic Active Contours Some hybridisations between geometric and parametric DMs have already been presented, like Geodesic Active Contours (GACs) [34]. Such approach is based on the relation between active contours

and the computation of geodesics or minimal distance curves, connecting classical "snakes" based on energy minimisation and geometric active contours based on the theory of curve evolution. The technique is based on active contours evolving in time according to intrinsic geometric measures of the image. The evolving contours naturally split and merge, allowing the simultaneous detection of several objects and both interior and exterior boundaries.

1.4 Soft computing

The term SC [183, 23] has been coined for this family of robust, intelligent systems, that in contrast to precise, traditional modes of computation, are able to deal with vague and partial knowledge. It mainly comprises four different partners: Fuzzy Logic (FL) [182, 143], EC [56, 7], Neural Networks (NNs) [73, 18] and Probabilistic Reasoning (PR) [142, 134]. The term *soft computing* distinguishes these techniques from *hard* computing that is considered less flexible and computationally demanding. The key point of the transition from hard to soft computing is the observation that the computational effort required by conventional computing techniques sometimes not only makes a problem intractable, but is also unnecessary as in many applications precision can be sacrificed in order to accomplish more economical, less complex and more feasible solutions. Imprecision results from our limited capability to resolve detail and encompasses the notions of partial, vague, noisy and incomplete information about the real world. In other words, it becomes not only difficult or even impossible, but also inappropriate to apply hard computing techniques when dealing with situations in which the required information is not available, the behavior of the considered system is not completely known or the measures of the underlying variables are noisy.

SC techniques are meant to operate in an environment that is subject to uncertainty and imprecision. According to [184], the guiding principle of SC is:

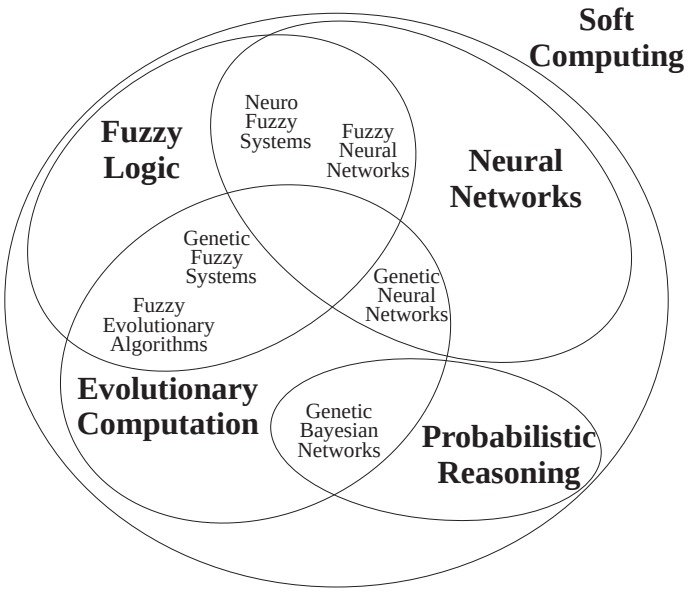


Figure 1.8: Hybridisation in SC

> "exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness, low solution cost and better rapport with reality."

All four methodologies that constitute the realm of SC have been conceptualized and developed over the past forty years. Each method offers its own advantages and brings certain weaknesses. Although they share some common characteristics, they are considered complementary as desirable features lacking in one approach are present in another. Consequently, after a first stage in which they were applied in isolation, the last two decades witnessed an increasing interest on hybrid systems obtained by symbiotically combining the four components of SC [24, 43, 83]. Figure 1.8 shows some hybrid systems positioned in the corresponding intersection of SC techniques.

Along its development, SC has gone beyond the use of its four initial components and has incorporated some others as rough set theory, chaos theory, belief theory, and Bayesian networks, as well as different branches form machine learning, such as clustering or decision trees. The next section is devoted to introduce EC, the SC methodology mainly used in this thesis.

1.5 Evolutionary computation

1.5.1 Conceptual foundations of evolutionary computation

Evolutionary Algorithms (EAs) constitute a class of search and optimisation methods, which imitate the principles of natural evolution [56]. The common term *evolutionary computation* comprises techniques such as *genetic algorithms*, *evolution strategies*, *evolutionary programming* and *genetic programming*. Their principal mode of operation is based on the same generic concepts, a population of competing candidate solutions, random combination and alteration of potentially useful structures to generate new solutions and a selection mechanism to increase the proportion of better solutions. The different approaches are distinguished by the genetic structures that undergo adaptation and the genetic operators that generate new candidate solutions.

Each cell of a living organism embodies a strand of DNA distributed over a set of chromosomes. The cellular machinery transcribes this blueprint into proteins that are used to build the organism. A gene is a functional entity that encodes a specific feature of the individual such as hair colour. The term allele describes the value of a gene, which determines the manifestation for an attribute, e.g., blond, black, brown hair colour. Genotype refers to the specific combination of genes carried by a particular individual, whereas the term phenotype is related to the physical makeup of an organism. Each gene is located at a certain position within the chromosome, which is called locus.

The literature in EAs alternatively avails the terms chromosome and genotype to describe a set of genetic parameters that encode a candidate solution to the optimisation problem. In the remainder of this dissertation, we will employ the term chromosome to describe the genetic structures undergoing adaptation. Consequently, the term gene refers to a particular functional entity of the solution, e.g., a specific parameter in a multi-dimensional optimisation problem. In classical genetic algorithms [65, 77],

solutions were often encoded as binary strings. In this context, a gene corresponds to a single bit, an allele corresponds to either a 0 or 1 and loci are string positions.

Progress in natural evolution is based on three fundamental processes: mutation, recombination and selection of genetic variants. The role of mutation is the random variation of the existing genetic material in order to generate new phenotypical traits. Recombination hybridizes two different chromosomes in order to integrate the advantageous features of both parents into their offspring. Selection increases the proportion of better adapted individuals in the population. Darwin coined the term *survival of the fittest* to illustrate the selection principle that explains the adaptation of species to their environment. The term fitness describes the quality of an organism, which is synonymous with the ability of the phenotype to reproduce offspring in order to promote its genes to future generations.

EAs provide a universal optimisation technique applicable to a wide range of problem domains, such as parameter optimisation, search, combinatorial problems, machine learning, and automatic generation of computer programs [56]. Unlike specialized methods designed for particular types of optimisation tasks, they require no particular knowledge about the problem structure other than the objective function itself. EAs are distinguished by their robustness, the ability to exploit accumulated information about an initial unknown search space in order to bias subsequent search into useful subspaces. They provide an efficient and effective approach to manage large, complex and poorly understood search spaces, where enumerative or heuristic search methods are inappropriate.

An EA processes a population of genetic variants from one generation to the next. A particular chromosome encodes a candidate solution of the optimisation problem. The fitness of an individual with respect to the optimisation task is described by a scalar objective function. According to Darwin's principle, highly fit individuals are more likely to be selected to reproduce offspring to the next generation. Genetic operators such as recombination and mutation are applied to the parents in order to generate new candidate solutions. As a result of this evolutionary cycle of selec-

tion, recombination and mutation, more and more suitable solutions to the optimisation problem emerge within the population.

The major components and the principal structure of a generic EA are shown in Fig. 1.9. During generation t, the EA maintains a population $P(t)$

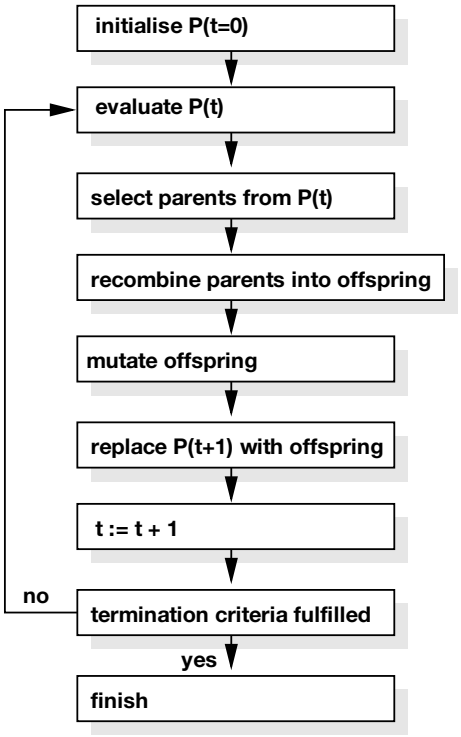


Figure 1.9: Generic structure of an Evolutionary Algorithm

of chromosomes. The population at time $t = 0$ is initialized at random. Each individual is evaluated by means of the said scalar objective function giving some measure of fitness. A set of parents is selected from the current population in a way that more fit individuals obtain a higher chance for reproduction. Recombination merges two chromosomes to form an offspring that incorporates features of its parents. In this way, recombination fosters the exchange of genetic information among individual members of the population. The offspring is subject to mutations, which randomly modify a gene in order to create new variants. The current population is replaced by considering the newly generated offspring, thus forming the next generation by only using them or a combination of the current and the

offspring population. The evolutionary cycle of evaluation, selection, recombination, mutation and replacement continues until a termination criterion is fulfilled. The stopping condition can be either defined by the maximum number of generations or in terms of a desired fitness to be achieved by the best individual in the population.

1.5.2 The scatter search template

This section will be focused on a specific kind of EA, SS.

SS fundamentals were originally proposed by Fred Glover [64] and have been later developed in some texts like [98]. The main idea of this technique is based on a systematic combination between solutions (instead of a randomized one like that usually done in genetic algorithms) taken from a considerably reduced evolved pool of solutions named Reference set (between five and ten times lower than usual genetic algorithms population sizes) as well as on the typical use of a local optimiser.

In this section we give the basic flow of the SS based on the well-known "five methods" template [98]. The advanced features of SS are related to the way these five methods are implemented. That is, the sophistication comes from the implementation of the SS methods instead of the decision to include or exclude some elements. The fact that the mechanisms within SS are not restricted to a single uniform design allows the exploration of strategic possibilities that may prove effective in a particular implementation. These observations and principles led the authors in [98] to propose the following template for implementing SS that consists of five methods.

1. A *diversification generation method* to generate a collection of diverse trial solutions, using an arbitrary trial solution (or seed solution) as an input.

2. An *improvement method* to transform a trial solution into one or more enhanced trial solutions. (Neither the input nor the output solutions are required to be feasible, though the output solutions will more usually be expected to be so. If no improvement of the input trial

solution results, the "enhanced" solution is considered to be the same as the input solution.)

3. A *reference set update method* to build and maintain a reference set consisting of the b "best" solutions found (where the value of b is typically small, e.g., no more than 20), organized to provide efficient accessing by other parts of the method. Solutions gain membership to the reference set according to their quality or their diversity.

4. A *subset generation method* to operate on the reference set, to produce a subset of its solutions as a basis for creating combined solutions.

5. A *solution combination method* to transform a given subset of solutions produced by the subset generation method into one or more combined solution vectors.

This basic design starts with the creation of an initial set of solutions P, and then extracts from it the Reference Set (*RefSet*) of solutions. The diversification generation method is used to build a large set P of diverse solutions. The size of P (*PSize*) is typically at least 10 times the size of *RefSet*. The reference set, *RefSet*, is a collection of both high quality solutions and diverse solutions that are used to generate new solutions by way of applying the combination method. In this basic design we can use a simple mechanism to construct an initial reference set and then update it during the search. The size of the reference set is denoted by $b = b_1 + b_2 =\mid RefSet \mid$. The construction of the initial reference set starts with the selection of the best b_1 solutions from P. These solutions are added to *RefSet* and deleted from P. For each solution in P - *RefSet*, the minimum of the distances to the solutions in *RefSet* is computed. Then, the solution with the maximum of these minimum distances is selected. This solution is added to *RefSet* and deleted from P and the minimum distances are updated[1]. This process

[1]In applying this maximum-minimum criterion, or any criterion based on distances, it can be important to scale the problem variables, to avoid a situation where a particular variable or subset of variables dominates the distance measure and distorts the appropriate contribution of the vector components.

is repeated b_2 times, where $b_2 = b - b_1$. The resulting reference set has b_1 high quality solutions and b_2 diverse solutions. Regardless of the rules used to select the reference solutions, the solutions in *RefSet* are ordered according to quality, where the best solution is the first one in the list. The simplest form of the subset generation method consists of generating all pairs of reference solutions. That is, the method would focus on subsets of size 2 resulting in $(b^2 - b)/2$ *NewSubsets*. The pairs in *NewSubsets* are selected one at a time in lexicographical order and the solution combination method is applied to generate one or more trial solutions. These trial solutions are subjected to the improvement method, if one is available. Then, the reference set update method is applied once again. The basic procedure terminates after all subsets in *NewSubsets* are subjected to the combination method and none of the improved trial solutions are admitted to *RefSet* under the rules of the reference set update method.

Figure 1.10 graphically shows the interaction among these five methods and puts in evidence the central role of the reference set.

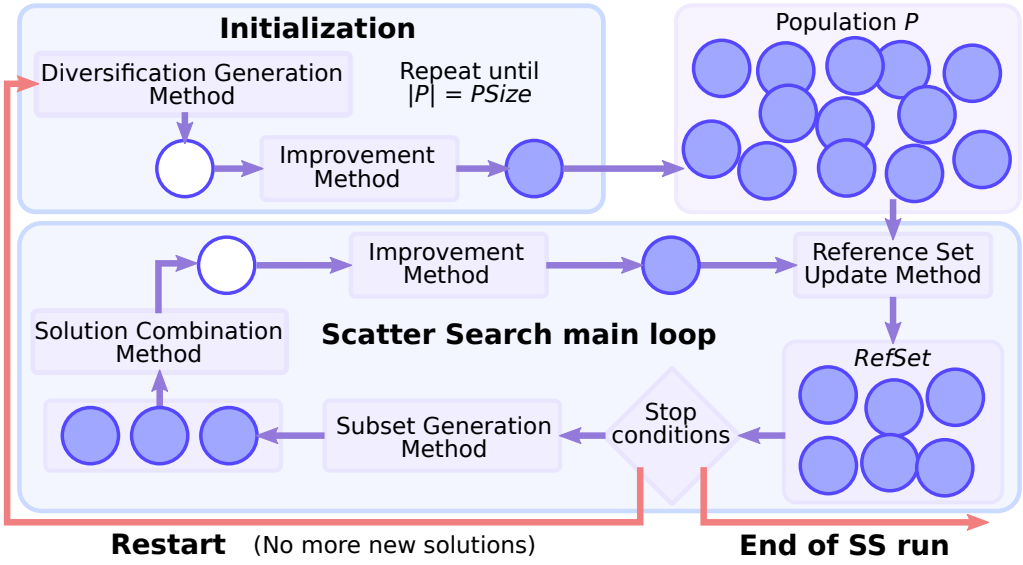


Figure 1.10: The control diagram of SS.

Of the five methods in the SS methodology, only four are strictly re-

quired. The improvement method is usually needed if high quality outcomes are desired, thus making the SS design become a memetic algorithm [139], but a SS procedure can be implemented without it.

CHAPTER 2

Image Modality

"There is something to be learned from a rainstorm. When meeting with a sudden shower, you try not to get wet and run quickly along the road. But doing such things as passing under the eaves of houses, you still get wet. When you are resolved from the beginning, you will not be perplexed, though you will still get the same soaking. This understanding extends to everything."

— 山本常朝, 葉隠

Imaging modality (e.g. MRI, CT, X-ray etc.) is an important aspect of the image for medical retrieval [76]. User-studies done on clinicians have indicated that modality is one of the most important filters that they would like to be able to limit their search by on medical image databases.

However, this modality is typically extracted from the caption and is often not correct or present. Studies have shown that the modality can be extracted from the image itself using visual features [144, 99, 82].

In this chapter we first introduce the reader with our general framework for image modality classification.

2.1 A framework for image modality classification

Methods for modality classification of medical images typically first extract a set of features from the available data. Choices here include a feature extraction method based on unimodal analysis, i.e. using only one source, or multimodal analysis, i.e. using two or more sources. Based on these extracted features, a classification algorithm assigns the images with the modality class labeling.

In general, as we have mentioned above, for modality classification of medical images two different sources are used for feature extraction, *text* and *image*. In the former case various text-based feature extraction methods are applied on the text which refers to the image in question. For example, one can define regular expressions [166] for certain keywords which describes an image modality and apply them on the caption of the images. In the latter case different Feature Descriptors (FDs), like Histogram of Oriented Gradients (HOG), Scale-invariant feature transform (SIFT) etc.—see Section 1.1—are employed to extract visual descriptors that later are used for modality classification.

After the feature extraction, the next step in modality classification is the actual classification step. Typically a supervised learning method is employed in this task, which is trained on the extracted features and the labeling of the images.

In case one extracts more than one feature vector they need to be fused somehow in order to be used by a supervised learning method. Snoek et al. [160] identified two general fusion strategies, namely: *early fusion* and *late fusion*. Figure 2.1 shows a general scheme for early fusion. Early fusion approaches first extract unimodal features. After analysis of the various unimodal sources, the extracted features are combined into a single representation, e.g. concatenation of unimodal feature vectors into one big feature vector. The main disadvantage of the approach is the difficulty to combine features into a common representation.

Approaches that rely on late fusion also start with extraction of uni-

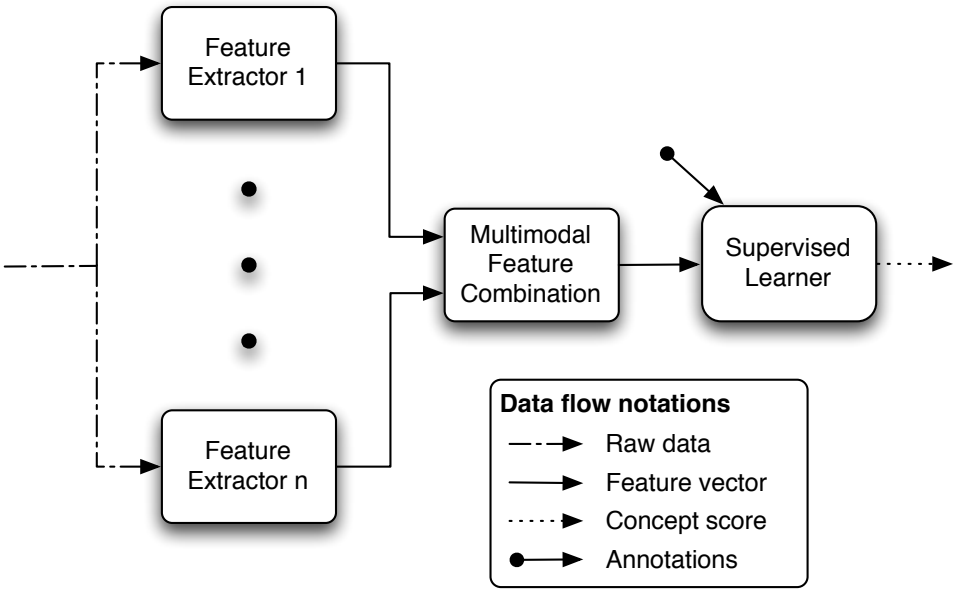


Figure 2.1: General scheme for *early fusion*. [160].

modal features, but in contrast to early fusion, where features are then combined into a multimodal representation, approaches for late fusion learn semantic concepts directly from unimodal features. These scores are combined afterwards to yield a final detection score. In general, late fusion schemes combine learned unimodal scores into a multimodal representation. Then late fusion methods rely on supervised learning to classify semantic concepts. Late fusion focuses on the individual strength of modalities. Unimodal concept detection scores are fused into a multimodal semantic representation rather than a feature representation. There are two big disadvantages of late fusion schemes (i) higher computational complexity than in case of early fusion, as every modality requires a separate supervised learning stage, moreover, the combined representation requires an additional learning stage and (ii) the potential loss of correlation in mixed feature space. Figure 2.2 illustrates a general scheme for late fusion.

As we have mentioned earlier depending on the data set, i.e. on the available feature vectors, there are different strategies to do modality classification of medical images, since there are data sets that only contain

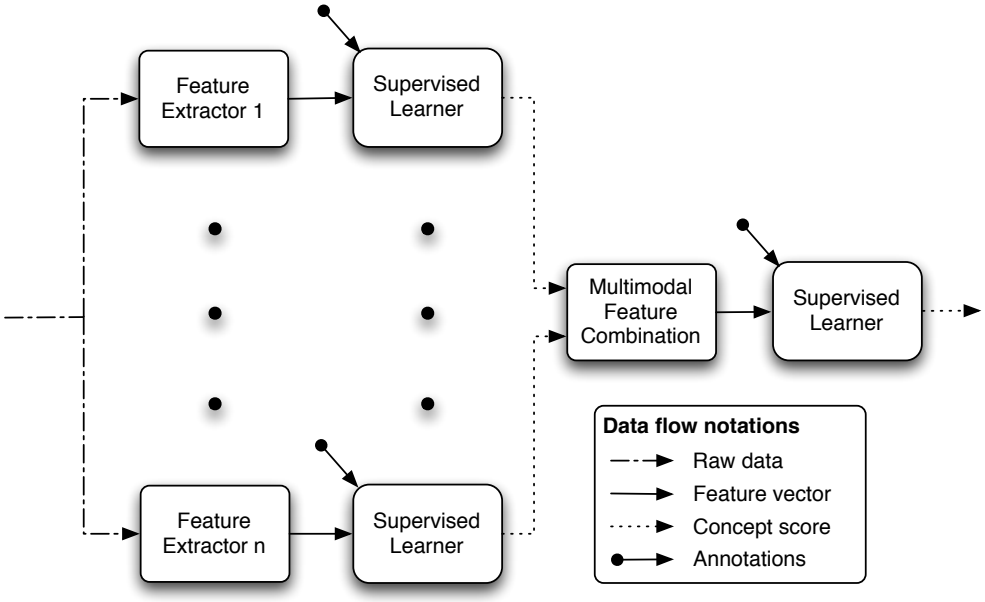


Figure 2.2: General scheme for *late fusion*. [160].

images, hence only visual features are available, but other data sets (like ImageCLEF, see Section 2.5.1 for more details) contain textual information regarding the images, e.g. simple image captions, sentences that refers to the image in question etc.

In order to be able to handle these differences of the data sets, we present a novel framework for modality classification of medical images. Figure 2.3 shows the different components of the framework and the connections among them.

The framework consists of the following components:

- *Feature extractor*: a set of user defined algorithms to extract feature vectors on the input data.
- *Feature encoder*: a user defined function to aggregate the acquired feature vectors.
- *Classifier*: an ML-based classification or regression method, to learn and infer the modality classes of the input images.

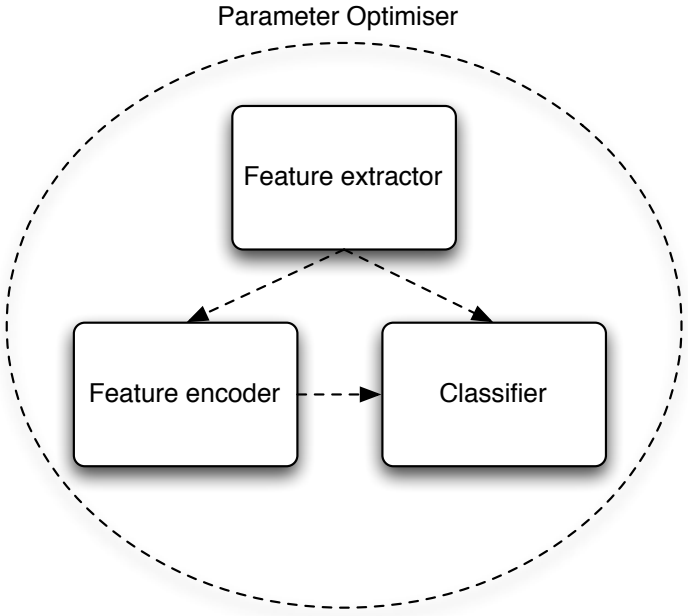


Figure 2.3: Framework for modality classification of medical images.

- *Parameter optimiser*: Hyper-Parameter Optimisation (HPO) or model selection method, like grid search.

The framework components are connected in a pipeline-like manner. The first component of the pipeline is the feature extractor, which is responsible to extracting different characteristics of the input (data set). Depending on the nature of the extracted feature it is either used directly by the classifier component or first processed by the feature encoder component. E.g. in case of a global image descriptor it is usually directly used by the classifier, as it is often a low dimensional feature vector. When the number of extracted FDs or their dimension is just too high, or they are not enough discriminative for a—supervised or unsupervised—Machine Learning (ML) algorithm, the features are first processed by the feature encoder component—essentially a dimension reduction method—, which generates a much more discriminative feature vector for the classifier. Finally the parameter optimiser component is responsible for fine tuning the various hyper-parameters of the algorithms that are used in the given pipeline.

In general, the choice of the most appropriate components is dependent on the problem at hand. One of the key strengths of the proposed framework is to being able to easily replace the framework components to tailor the system to specific data sets.[1]

2.2 Features

2.2.1 Feature detection based on a priori information

In many cases features can be defined and detected based on prior knowledge of the images we are working with. For instance we might be aware that some of the images contain graphs, which contain large white areas and straight axes, or—especially in the case of medical images—we might have descriptions of the contents in textual form for example as a caption. In the following we briefly summarize the most popular methods.

2.2.1.1 Textual features

The methods outlined below rely on non-visual information embedded in the images. The main advantage of this information is that it usually comes from human input and can be considered reliable since the purpose of labeling is often classification, however it should not be blindly trusted. Naturally the usefulness and availability of these feature descriptors depend on the data to be classified, so that the classifier has to be set up accordingly.

Caption text Figures in scientific publications often have descriptive captions that provide information on the modality of the image. "Contrast-enhanced axial computed tomographic scan", "HRCT showing extensive areas of consolidation with air bronchogram" are examples of captions of images assigned to the 'CT' modality class. However, the caption may be missing or may not hint at the modality, e.g. "E. coli that satisfy the similarity threshold values." As the examples suggest, the linguistic constructs

[1] `https://github.com/illes/multimodal`

expressing modality can have a high variation. To deal with the problem of ambiguity Boolean operators, AND, OR and wildcards are used, like in any common text based search tool. To broaden the scope of the search and find more of the relevant result candidates, a medical thesaurus – such as the UMLS metathesaurus or MeSH terms - can be used to automatically obtain synonyms for the search terms.

Considering these remarks, we can extract binary features from caption texts as follows. Let's define a set of regular expressions to be matched against the caption text, a match results in a value of 1. Regular expressions should be created for each word having a high information gain for any of the modality classes and can be manually refined later on to capture linguistic variations (e.g. `f?MRI?`) and multi-word phrases (e.g. `error bars?`).

Meta-data Besides human input—the caption text—automatically added information can be of help to the classification process. By analysing meta-data stored in JPEG files' EXIF section it can be determined whether image post-processing software was used. The 'Comment' field contains mentions of commonly used image manipulation software (e.g. Adobe Photoshop, MS Paint). Another piece of information that can be extracted from the EXIF is whether the image is stored as gray-scale only. This hints on the method for image capture e.g. radiology and determines later post processing possibilities.

MeSH terms Scientific articles indexed by Medline/PubMed[2] are tagged with MeSH terms (medical subject headings) by field experts. This makes them especially useful for classification. MeSH terms can be seen as a thesaurus for the life sciences containing entries like 'Human', 'Liver Neoplasms' and 'Magnetic Resonance Imaging', entries can be further qualified by e.g. 'methods', 'pathology'. Searching and indexing is further facilitated by a descriptor hierarchy in which the descriptors first describe a broader category—such as diseases—then lower level descriptors narrow down the category, e.g. Digestive System Diseases. The thesaurus-like quality of the

[2] `http://www.ncbi.nlm.nih.gov/pubmed`

index makes it suitable to be used in conjunction with caption text-based classification, to provide synonyms for a given term. Since the purpose of MeSH is to identify the topic of journal articles it can be safely assumed that the article's MeSH terms and its figures' modality are correlated, and hence define features corresponding to individual MeSH terms and qualifiers. A unique identifier for the article (e.g. PMID or DOI) is required to retrieve its MeSH annotations, however, such identifiers can be absent. As the number of MeSH terms, qualifiers and their combinations far exceeds the number of modality labels, a feature selection has to be performed by keeping only those that are present for at least a predefined number of articles in the training set.

Radiopaedia Radiopaedia[3] is a community wiki for radiology images and patient cases. Images are tagged by users with the body system (e.g. Heart, Musculoskeletal) depicted, but unfortunately for us, not with the type of radiology method used to create the image. Leveraging the mutual information between body systems and radiology methods, features for modality classification can be derived by taking the output probabilities of a classifier trained to predict body systems shown in the image.

2.2.1.2 Visual features

Apart from textual clues visual features can be defined based on observations on the input images. For instance we might be aware that some of the images contain graphs, which contain large white areas and straight axes. A certain type of medical images might also contain substantial areas of a specific colour band, a good example is human skin.

Mean of pixels If it is known in advance that the images to be classified contain graphs and diagrams, such as the **Graphic** 1st-level group of the ImageCLEF 2011 (see Section 2.5.1) data set we can create a descriptive feature from the knowledge that the background pixels are mainly white.

[3] `http://radiopaedia.org`

Hence, we can defined a simple feature $f_{mean} = \overline{\mathbf{I}_j}$, that represents the mean value of the pixels in an image. By simply thresholding these values one could identify the images that belong to the **Graphic** group with a very high accuracy.

Axis recognition The previously mentioned mean of pixels method gave a strong support for recognising images in the **Graphic** top-level group, but as it consists of two sub-groups, **Graphs** and **Drawing**, thus a new feature was required to differentiate the images belonging to one or the other category. By observing the images in these two categories one can easily point out the main difference by using a simple edge detector: the images belonging to the **Graphs** category are mainly consisting of horizontal and vertical lines (i.e. the x-y axis of a graph), whereas the images in **Drawing** category are mostly diagrams, where the orientation of the lines is random.

Based on this idea we can define the following feature. Let $L_{\vec{I}_j}$ be the set of all the detected lines and $GL_{\vec{I}_j}$ be the set of *good lines* in an arbitrary image $\vec{I}_j$, where a given line is a *good line* if its orientation is horizontal or vertical and it is within a given margin of the picture's border. The latter condition is to eliminate the borders of an image as *good lines*.

Using these two sets the following feature can be defined

$$f_{lines}(\vec{I}_j) = \frac{|GL_{\vec{I}_j}|}{|L_{\vec{I}_j}|} \tag{2.1}$$

In order to detect the lines and their orientation in an image a simple Hough transform can be used [55].

Skin detection A special case of colour based feature definition is when the type of colour falls in a well-defined colour band and is known in advance; a good example is skin detection. Medical images that show various skin abnormalities are usually simple photographs thus they have very similar characteristics to the general photo labeled images. The challenge lies in defining a good descriptive feature that uniquely identifies skin. As an example feature detector to our framework we decided to apply the algorithm

from [35]. This algorithm uses a multi stage process to locate areas in the image where a facial area is present. Its main purpose is to enhance face area quality in video conferences however the detection itself only relies on topological and geometrical features of the face itself at its last steps, so it is well suited for general skin detection.

As a first step colour segmentation is performed using a pre-defined colour map in YCrCb colour space. Regions containing skin fall into a very narrow chromaticity band regardless of skin colour, however objects in the background can also fall into this band, producing a noisy output from this stage. The second stage eliminates some of the noise by binarising the result and performing morphological operations thus getting rid of smaller isolated areas, then density regularisation is performed that removes pixels that have a low number of skin coloured neighbors according to a pre-defined threshold. The third stage further reduces background regions by filtering areas that have a homogeneous luminance as the background tends to have more homogeneous lighting. The rest of the steps are only relevant for face detection, therefore we omit them.

Using this simple skin detector algorithm the feature can be defined the following way: $f_{skin}(\vec{I}_j)$ for an image $\vec{I}_j$

$$f_{skin}(\vec{I}_j) = \overline{SD(\vec{I}_j)} \tag{2.2}$$

where the function $SD(\cdot)$ calculates the skin-segmented binary image of an input image, and $\overline{\vec{I}_k}$—as previously defined—is the mean value of image $\vec{I}_k$.

2.2.2 Generic Feature Descriptors

More general state of the art methods do not rely on a priori information, they extract visual features from the raw data of the images, and use these for classification. Some of these methods have already been introduced to the reader in Section 1.1.

2.2.2.1 Dense SIFT

Dense SIFT (DSIFT) [173] is a fast algorithm for the calculation of a large number of SIFT (see Section 1.1.3) descriptors of densely sampled features of the same scale and orientation.

Various studies have shown [26, 25] that when applying the SIFT descriptor to tasks such as scene classification or object category classification, often better classification results are obtained by computing the SIFT descriptor over dense grids in the image domain, opposed to at sparse interest points obtained by an interest operator. Basically this is because a larger set of local image descriptors computed over a dense grid usually provide more information than corresponding descriptors evaluated at a much sparser set of image points.

2.2.2.2 Affine-invariant SIFT

The SIFT detector normalizes rotations and translations and simulates all zooms out of the query and of the search images. Because of this feature, it is the only fully scale-invariant method.

Affine-invariant SIFT (ASIFT) [127] simulates with enough accuracy all distortions caused by a variation of the camera optical axis direction and then it applies the SIFT method. Specifically, ASIFT simulates three parameters: (i) the scale, (ii) the camera longitude angle, (iii) and the latitude angle, which is equivalent to the tilt and normalizes the translation and rotation. The key observation is that, although a tilt distortion is irreversible due to its non-commutation with the blur, it can be compensated up to a scale change by digitally simulating a tilt of same amount in the orthogonal direction. As opposed to the normalisation methods that suffer from this non-commutation, ASIFT simulates and hence achieves the full affine invariance.

ASIFT proceeds by the following steps:

1. Each image is transformed by simulating all possible affine distortions caused by the change of camera optical axis orientation from a frontal

position. These distortions depend upon two parameters: the longitude φ and the latitude θ. The images undergo φ-rotations followed by tilts with parameter $t = |\frac{1}{\cos\theta}|$ (a tilt by t in the direction of x is the operation $u(x, y) \to u(tx, y)$). For digital images, the tilt is performed by a directional t-subsampling. It requires the previous application of an anti-aliasing filter in the direction of x, namely, the convolution by a Gaussian with standard deviation $c\sqrt{t^2 - 1}$. The value $c = 0.8$ is the value chosen by Lowe [112] for the SIFT method, which ensures a very small aliasing error [128].

2. These rotations and tilts are performed for a finite and small number of latitude and longitude angles, the sampling steps of these parameters ensuring that the simulated images keep close to any other possible view generated by other values of φ and θ.

3. All simulated images are compared by a similarity invariant matching algorithm.

2.2.3 Feature encoding

2.2.3.1 Bag of visual-words

The state-of-the-art content based image retrieval systems has been significantly improved by the introduction of SIFT features and the Bag-of-Words (BoW) image representation [135, 85, 40, 145].

The Bag of visual words (BoVW) image representation is based on the BoW model in Natural language processing (NLP). BoW in NLP is a popular method for representing documents. In this model a document is simply represented by the number of different words that are in the document. The idea behind this is, that documents on the same topic have similar words with similar number of occurrences in them (see Latent Dirichlet allocation (LDA) [22]).

In case of an image, the basic idea of BoW is that a set of local image patches is sampled using some method—e.g. densely or using a key-point detector—and a vector of visual descriptors is evaluated on each patch

independently. Since the visual-word image representation is analogous to the BoW representation of text documents in terms of both form and semantics, it makes techniques for text categorisation readily applicable to the problem of scene classification.

Images can be represented by sets of key-point descriptors, but the sets vary in cardinality and lack meaningful ordering. This creates difficulties for learning methods (e.g. classifiers) that require feature vectors of fixed dimension as input.

Codebook generation After acquiring the FDs for all the images in the data set, the next step in the BoVW model is to convert these feature vectors to *codewords*—analogy to words in text documents—, which produces a *codebook* ($\mathbf{D}$)—analogy to a word dictionary.

One of the simplest ways to acquire these codewords is by performing K-means clustering over all the extracted FDs [106]. The codewords in this case are the centres of the learned clusters. The number of the clusters is the codebook size (M)—the size of the word dictionary.

In other words, let $\mathbf{X}$ be a set of N-dimensional FDs extracted from an image, i.e. $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_K] \in \mathbb{R}^{N \times K}$. Given a codebook with M entries, $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_M] \in \mathbb{R}^{N \times M}$, different coding schemes convert each descriptor into a M-dimensional code to generate the final image representation.

Unlike the vocabulary of a text corpus whose size is relatively fixed, the size of a visual-word vocabulary is controlled by the selected number of cluster points in the clustering process. The choice of the vocabulary size involves the trade-off between discriminativity and generalisability. In case of a small visual-word vocabulary, a feature is not very discriminative, dissimilar FDs can be mapped to the same visual word. As one increases the vocabulary size, the codewords become more discriminative, but in the meanwhile less generalisable and more forgiving to noises, as similar FDs can be mapped to different visual words. Using a large vocabulary not only increases the cost of clustering, i.e. the codebook generation, but as well increases the dimension of the generated feature vectors, consequently increases the training time of the supervised classifier using these features.

There is no consensus as to the appropriate size of a visual-word vocabulary. The vocabulary size used in existing works varies from several hundreds [103], to thousands and tens of thousands [158, 186]. These results are not directly comparable due to the difference on corpus and classification methods.

Codeword assignment In the BoVW representation two FDs are considered identical if they are assigned to the same visual word—cluster centre. Whereas, two features assigned to different—even very close—clusters are considered totally different. In effect the quantisation provides a very coarse approximation to the actual distance between the two features—zero if assigned to the same visual word, and infinite otherwise. In practice this hard assignment leads to errors because of variability in the FD [145].

The variability arises from many sources: image noise, varying scene illumination, instability in the feature detection process and non-affine changes in the measurement regions. Distortions that cannot be handled by the invariance built into the FD result in a change in the descriptor value, and in turn this may result in the same surface patch being assigned to different visual words in different images. Descriptors corresponding to the same physical patch in different images will usually be close to each other. Although, this is not always the case. Severe image distortions, such as strong lighting variations or self occlusions can abruptly change the descriptor value of the patch.

As mentioned earlier, BoVW model generally uses vector quantisation (VQ) coding for codeword assignment of the extracted FDs, which solves the following constrained least square fitting problem:

$$\begin{aligned} \underset{C}{\text{argmin}} \quad & \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{D}\mathbf{c}_i||^2 \\ \text{s.t.} \quad & \forall i \; ||\mathbf{c}_i||_0 = 1, ||\mathbf{c}_i||_1 = 1, \mathbf{c} \geq 0 \end{aligned} \tag{2.3}$$

where $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N]$ is the set of codes for $\mathbf{X}$. The cardinality constraint $|\mathbf{c}_i|_0 = 1$ means that there will be only one non-zero element in each code $\mathbf{c}_i$, corresponding to the quantisation id of $\mathbf{x}_i$. The non-negative, l_1

constraint $|\mathbf{c}_i|_1 = 1, \mathbf{c}_i \geq 0$ means that the coding weight for $\mathbf{x}$ is 1. In other words, the single non-zero element is found by searching the nearest neighbor.

In order to mitigate the quantisation loss of VQ, the restrictive cardinality constraint $||\mathbf{c}_i||_0 = 1$ in Ex 2.3 can be relaxed by using a sparsity regularisation term. In a study by Yang et al. [180] such a sparsity regularisation term is selected to be the l_1-norm of $\mathbf{c}_i$, and coding each FD $\mathbf{x}_i$ thus becomes a standard sparse coding [104] problem:

$$\underset{C}{\operatorname{argmin}} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{D}\mathbf{c}_i||^2 + \lambda||\mathbf{c}_i||_0 \tag{2.4}$$

The sparsity regularisation term plays several important roles:

1. The codebook D is usually over-complete, i.e. $M > N$, and hence l_1 regularisation is necessary to ensure that the under-determined system has a unique solution.

2. The sparsity prior allows the learned representation to capture salient patterns of FDs.

3. The sparse coding can achieve much less quantisation error than VQ.

As a result a much better classification accuracy can be achieved [180].

Weighting schemes Since term weighting is a key technique in Information Retrieval (IR) [8, 152], we explore its use in adjusting visual-word vectors. Two major factors in term weighting are *term frequency* (tf) and *inverse document frequency* (idf). A third factor is the normalisation factor, which converts the feature into unit-length vector to eliminate the difference between short and long documents.

We apply popular term weighting schemes in IR to the visual-word feature vectors. These schemes are summarized in Table 2.1, where they are named after the convention in [152]. These schemes are chosen to allow us to study the impact of tf, idf and the normalisation factor. Note that tf_i is the number of times a visual word t_i appears in an image, N is the total

Name	Factors	Value for t_i
txx	tf	tf_i
txc	tf, normalization	$\frac{tf_i}{\sum_i tf_i}$
tfx	tf-idf	$tf_i \cdot \log(\frac{N}{n_i})$
tfc	tf-idf, normalization	$\frac{tf_i \cdot \log(\frac{N}{n_i})}{\sum_i tf_i \cdot \log(\frac{N}{n_i})}$

Table 2.1: Weighting schemes for visual-word feature

number of images in the corpus, and n_i is the number of images having visual-word t_i.

2.3 Kernels for Image Classification

As previously introduced in Section 1.2.2.3 by using a kernel function, i.e. mapping a linearly inseparable training data into a higher dimensional space it becomes separable.

In this section we are going to review various kernels that have been recently introduced and successfully applied in the field of image categorisation [113, 9].

2.3.1 Histogram intersection kernel

The Histogram Intersection Kernel (HIK) [164] is often used as a measurement of similarity between histograms $\mathbf{x}$ and $\mathbf{z}$.

$$K_{HIK}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} \min(x_i, z_i) \tag{2.5}$$

A generalisation of the HIK is defined by [28]:

$$K_{GHIK}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} \min(|x_i|^\beta, |z_i|^\beta) \tag{2.6}$$

that is positive definite for all $\beta > 0$.

Since the HIK is positive definite [137] it can be used as a kernel for discriminative classification using Support Vector Machine (SVM).

In recent publications, intersection kernel SVMs have been shown to be successful for detection and recognition, e.g. pyramid match kernel [68] and spatial pyramid matching [103]. Unfortunately, as it was shown in Section 1.2.2.3, this success typically comes at great computational expense, as non-linear kernels require memory and computation linearly proportional to the number of support vectors for classification.

Maji et al. [113] introduced a way to overcome the memory and computational limitations of the intersection kernel SVM classification. The key idea is that for a class of kernels (including the intersection kernel), the classifier can be decomposed as a sum of functions, one for each histogram bin, each of which can be efficiently computed. Using this approach a classification with time complexity $\mathcal{O}(n \log m)$ and space complexity $\mathcal{O}(nm)$. Moreover, using an approximation scheme a time and space complexity of $\mathcal{O}(n)$, independent of the number of support vectors can be achieved.

2.3.2 Information theoretic kernels

Kernels on probability measures have been shown very effective in classification problems involving text, images, and other types of data [49, 84]. Given two probability measures P_i and P_j, representing two objects, several information theoretic kernels can be defined [120].

2.3.2.1 Jensen-Shannon kernel

The Jensen-Shannon Kernel (JSK) is a non-extensive mutual information kernel defined by non-extensive entropy. Extensive entropy is such that the joint entropy of two probability distributions is the sum of their individual entropies. Non-extensive entropy relaxes this condition. The idea of non-extensive entropy was first introduced by Havrda and Charvit[72], and its applications have been extensively documented by Gell-Mann and Renyi [62].

The JSK is computed from the Jensen-Shannon Divergence (JSD), which is an example of a non-extensive entropy measure. In quantum physics the JSD has been extended from classical probability distributions to density matrix representations. It has been shown to be a convenient means of measuring both the entanglement [114] and mixing [115] of quantum states, and also to have useful metric properties [101].

The JSK is defined on probability distributions over structured data. Assume $M_+^1(\mathcal{X})$ is a set of probability distributions where $\mathcal{X}$ is a set provided with some σ-algebra of measurable subsets, the kernel $K_{JS} : M_+^1(\mathcal{X}) \times M_+^1(\mathcal{X}) \to \mathbb{R}$, is positive definite with the following kernel function [120]:

$$K_{JS}(P_i, P_j) = \ln(2) - JSD(P_i, P_j) \tag{2.7}$$

where $JSD(P_i, P_j)$ is the JSD between the probability distributions P_i and P_j defined as

$$JSD(P_i, P_j) = H\left(\frac{P_i + P_j}{2}\right) - \frac{H(P_i) + H(P_j)}{2} \tag{2.8}$$

For a mixture of n probability distributions $P_1, \ldots, P_n$ with mixing proportions $\pi_1, \ldots, \pi_n$, the divergence is given by:

$$JSD(P_1, \ldots, P_n) = H(\sum_{i=1}^{n} \pi_i P_i) - \sum_{i=1}^{n} \pi_i H(P_i) \tag{2.9}$$

where $H(P_i)$ is the Shannon entropy for the probability distribution P_i given by:

$$H(P_i) = -\sum_{k=1}^{n} P_i(k) \log P_i(k) \tag{2.10}$$

where $P_i(k)$ is the k-th element of the probability distribution P_i.

2.3.2.2 Jensen-Tsallis kernel

Jensen-Tsallis Kernel (JSK) is given by

$$K_{JT}(p_1, p_2) = \ln_q(2) - T_q(p_1, p_2) \tag{2.11}$$

where $\ln_q(x) = \frac{x^{1-q}-1}{1-q}$ is the q-logarithm,

$$T_q(p_1, p_2) = S_q\left(\frac{p_1 + p_2}{2}\right) - \frac{S_q(p_1) + S_q(p_2)}{2^q} \tag{2.12}$$

is the Jensen-Tsallis q-difference, and $S_q(r)$ is the Jensen-Tsallis entropy, defined for a multinomial $r = (r_1, \ldots, r_L)$, with $r_i \leq 0$ and $\sum_i r_i = 1$, as

$$S_q(r_1, \ldots, r_L) = \frac{1}{1-q}\left(1 - \sum_{i=1}^{L} r_i^q\right) \tag{2.13}$$

2.3.3 Combined kernel

As we have discussed in Section 2.1 there are various strategies to combine feature vectors. The most straightforward solution is to apply early fusion, i.e. simply join the extracted feature vectors into one big vector. The problem with this approach that it has been proven that based on the nature and the source of the feature vectors different kernels will be more suitable, i.e. will yield to better classification accuracy. For example in case of histogram-based features a HIK or JSK yields to a much better accuracy than linear kernel [113, 49, 84].

To tackle the above mentioned problems, where features are from multiple, heterogeneous data sources, one can define a combined kernel [161] to construct weighted linear combinations of multiple kernels in the following way:

$$K_{combined}(\mathbf{x}, \mathbf{z}) = \sum_{m=1}^{M} \beta_m K_m(\mathbf{x}, \mathbf{z}) \tag{2.14}$$

where $\beta_m > 0$ and each kernel K_m uses only a distinct set of features.

2.3.4 Kernel normalisation

Previous studies [75] have shown that normalisation of the vectors in the input space plays an important role in Support Vector (SV) classification. Normalisation of the input data not only can influence dramatically the results of the classification, but the convergence of the SV algorithm as well.

Graf et al. [67] showed that normalisation in the input space was not appropriate when considering non-linear SVMs since the feature space where classification is performed is then not normalized.

Let us consider the most elementary type of preprocessing for SVMs, normalisation of vectors $\mathbf{x} \in \mathbb{R}^N$ in input space. The corresponding normalised vectors $\mathbf{x}'$ are given by:

$$\mathbf{x}' = \frac{\mathbf{x}}{||\mathbf{x}||_2} = \frac{\mathbf{x}}{\sqrt{\sum_{i=1}^{N} |x_i|^2}} \tag{2.15}$$

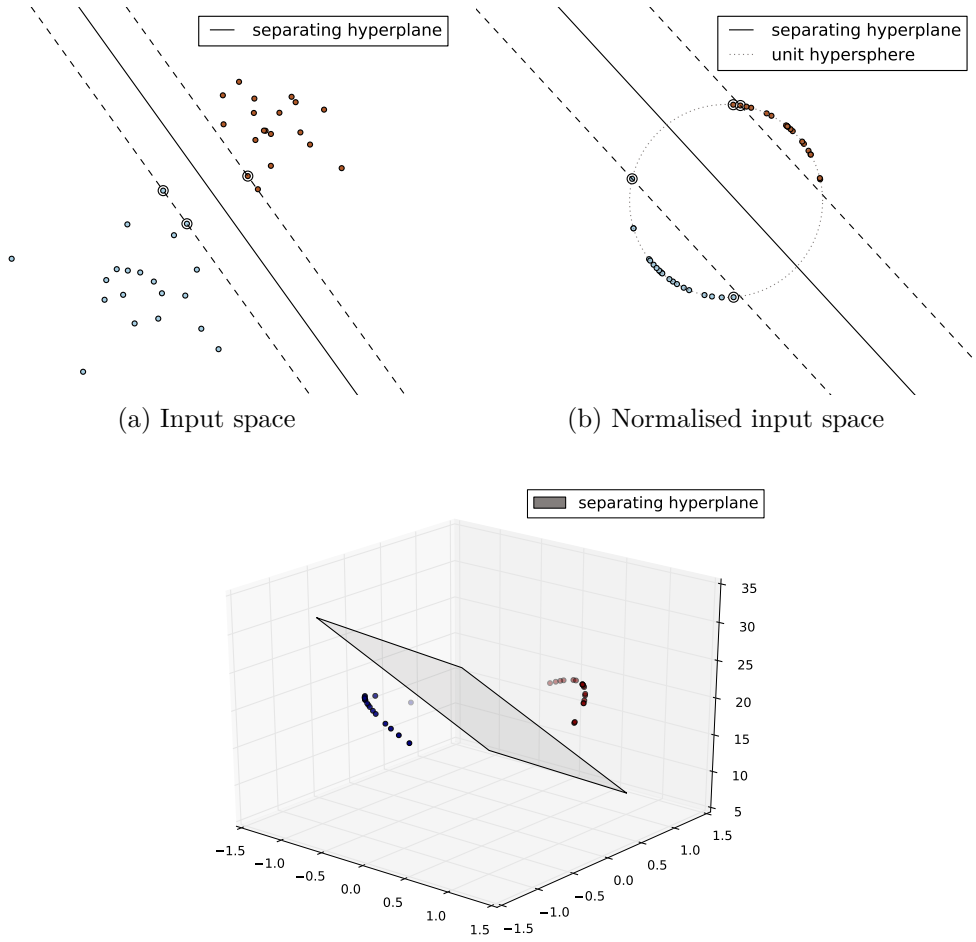
The vector $\mathbf{x}'$ lies on a unit hypersphere in $\mathbb{R}^N$. The SV is designed to find the Optimal Separating Hyperplane (OSH) in the feature space which is obtained by a non-linear mapping from the normalised input space. When considering the effect of such a mapping, in most cases we lose the normalisation or scaling in the feature space as shown in Figure 2.4. This may create a problem for the SV algorithm since it yields best classification performance for "input" vectors in the feature space which are in some way "scaled" [44, 172].

A natural extension is thus to normalize the feature space, yielding normalised kernel functions:

$$K'(\mathbf{x}, \mathbf{z}) = \frac{K(\mathbf{x}, \mathbf{z})}{\sqrt{K(\mathbf{x}, \mathbf{x})K(\mathbf{z}, \mathbf{z})}} \tag{2.16}$$

Clearly $K'(\mathbf{x}, \mathbf{x}) = 1$. Thus, all vectors in the feature space lie on a unit hypersphere. It is easy to see that for linear dot products, normalisation in feature space and in input space is equivalent, that is, Ex. 2.15 and Ex. 2.16 coincide. The normalisation of kernels is a conformal transformation [5] of the original kernels. Hence, the angles between vectors of the feature space are invariant with respect to normalisation of the kernel functions [159].

In the case of binary SVMs with positive kernel functions, these normalised kernels play a predominant role. The angles between all vectors are then less than $\frac{\pi}{2}$ since $K(\mathbf{x}, \mathbf{z}) > 0$, thus, the data points are placed on a portion of the same orthant on the unit hypersphere in the feature space allowing them to be separated from the origin by a hyperplane.



(a) Input space

(b) Normalised input space

(c) Feature space

Figure 2.4: Normalisation in the input space and its representation in the feature space.

2.3.5 Homogeneous kernel map

Despite the effectiveness in classification of non-linear data, the non-linear (kernel-based) SVM has much higher computational complexity for prediction than its linear counterpart. It requires $|SV|d$ summations and $|SV|$ exponential operations to compute the kernel function values for evaluating the classifier output at a single instance, where d is the data dimension and $|SV|$ is the cardinality of the set of support vectors. Thus, the larger the training data set the greater the number of supports vectors and consequently—see above—higher complexity of prediction. Hence non-

linear SVM is inefficient for large-scale prediction tasks, where there are thousands or millions of data points to classify in the testing set.

Linear SVMs, on the other hand, are simple to train and use, as they only involve inner product operations with the input data. Moreover, recent advances have made it possible to learn linear SVMs in time linear with the number of training examples [86, 78]. However, they can be quite restricted in discriminative power and can not be applied to non-linear data.

Hence, it is desirable to have a classifier model with both the efficiency of linear SVMs and the power of non-linear SVMs.

The Homogeneous Kernel Map (HKM) [174] is a finite dimensional linear approximation of homogeneous kernel, including the intersection, χ^2, and Jensen-Shannon kernels. These kernels are frequently used in computer vision applications because they are particular suited to data in the format of histograms, which includes many common visual descriptors.

Let $x, y \in \mathbb{R}_+$ be non-negative scalars and let $k(x, y) \in \mathbb{R}^+$ be a homogeneous kernel such as the intersection one:

$$k_{HIK} = \min(x, y) \tag{2.17}$$

For vectorial data $\mathbf{x}, \mathbf{y} \in \mathbb{R}_+^d$ the homogeneous kernels is defined as an additive combination of scalar kernels $K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} k(x_i, y_i)$.

The HKM of order n is a vector function $\Psi(x) \in \mathbb{R}^{2n+1}$ such that, for any choice of $x, y \in \mathbb{R}_+$, the following approximation holds:

$$k(x, y) \approx \langle \Psi(x), \Psi(y) \rangle \tag{2.18}$$

Given the feature map for the scalar case, the corresponding feature map $\Psi(\mathbf{x})$ for the vectorial case is obtained by stacking $[\Psi(x_1), \ldots, \Psi(x_n)]$. Note that the stacked feature $\Psi(\mathbf{x})$ has dimension $d(2n + 1)$.

Using linear analysis tools, like linear SVM on top of data set that has been encoded by the HKM is therefore approximately equivalent to using a method based on the corresponding non-linear kernel.

Any positive (semi-)definite kernel $k(x, y)$ defined on the non-negative reals $x, y \in \mathbb{R}_+$ can be extended to the entire real line by using the definition:

$$k_{\pm}(x, y) = sign(x)sign(y)k(|x|, |y|). \tag{2.19}$$

The HKM implements this extension by defining $\Psi_{\pm}(x) = sign(x)\Psi(|x|)$.

Any (1-)homogeneous kernel $k_1(x, y)$ can be extended to a so called γ-homogeneous kernel $k_\gamma(x, y)$ by

$$k_\gamma(x, y) = (xy)^{\frac{\gamma}{2}} \frac{k_1(x, y)}{\sqrt{xy}} \tag{2.20}$$

Smaller values of γ enhance the kernel non-linearity and are sometimes beneficial in applications.

2.4 Hyper-parameter optimisation

HPO or model selection is the problem of choosing a set of hyper-parameters for an algorithm, usually with the goal of obtaining good generalisation.

From the previous sections one can see that each method, regardless whether it is a ML or a feature extraction method, has a set of hyper-parameters—flags, values, and other configuration information—that guides the algorithm. Sometimes this configuration applies to the space of functions that the learning algorithm searches (e.g. the number of dictionary element of the BoVW model, see Section 2.2.3.1). Sometimes this configuration applies to the way the features are interpreted by the learning algorithm (e.g. the weights of each kernel in case of combined kernel, see Section 2.3.3).

The common practice is to judge a learning algorithm by its best-case-scenario performance. Hence, researchers are expected to maximize the performance of their algorithm by optimising over hyper-parameter values by e.g. cross-validating using data withheld from the training set. Despite decades of research into global optimisation [133, 93, 148] and the publishing of several HPO algorithms [131, 79, 17, 118] it would seem that most ML researchers still prefer to carry out this optimisation by hand, and by grid search.

2.4.1 Grid Search

The *de facto* standard way of performing HPO is grid search, which is simply an exhaustive searching through a manually specified subset of the hyper-parameter space of an algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set.

For example in case of a typical soft-margin SVM classifier equipped with a combined kernel, after the M different kernels have been chosen, one needs to set the right weight for each kernel, i.e. $\beta_i,\ i \in [1, M]$. In order to perform grid search, one selects a finite set of "reasonable" values for each β_i, say

$$\begin{aligned} \beta_1 &\in \{0.1, 0.2, 0.5, 1.0\} \\ \beta_2 &\in \{0.05, 0.3, 0.55, 1.0\} \\ &\dots \\ \beta_M &\in \{0.15, 0.25, 0.65, 1.0\} \end{aligned}$$

Grid search then trains an SVM with each tuple $(\beta_1, \dots, \beta_M)$ in the Cartesian product of these M number of sets and evaluates their performance on a held-out validation set (or by internal cross-validation on the training set, in which case multiple SVMs are trained per pair). Finally, the grid search algorithm outputs the settings that achieved the highest score in the validation procedure.

Grid search suffers from the curse of dimensionality, but is often embarrassingly parallel because typically the hyper-parameter settings it evaluates are independent of each other.

Bergstra and Bengio [16] showed empirically and theoretically that randomly chosen trials are more efficient for hyper-parameter optimisation than trials on a grid. In case of a Neural Network (NN) configured by a pure grid search, they found that random search over the same domain is able to find models that are as good or better within a small fraction of the computation time. Using the same computational time random search finds better models by effectively searching a larger, less promising configuration space.

By doing a Gaussian process analysis of the function from hyper-parameters to validation set performance they found that for most data sets only a few of the hyper-parameters really matter, but that different hyper-parameters are important on different data sets. This phenomenon makes grid search a poor choice for configuring algorithms for new data sets.

2.4.2 irace

López-Ibánez et al. [110] introduced a package called Iterated Race for Automatic Algorithm Configuration, an iterated racing procedure, which is an extension of the Iterated F-race procedure [10].

Iterated racing is a method for automatic configuration that consists of three steps: (i) sampling new configurations according to a particular distribution, (ii) selecting the best configurations from the newly sampled ones by means of racing, and (iii) updating the sampling distribution in order to bias the sampling towards the best configurations. These three steps are repeated until a termination criterion is met. In iterated racing, each configurable parameter has an independent sampling distribution, which is either a normal distribution for numerical parameters, or a discrete distribution for categorical parameters. The update of the distributions consists of modifying the sampling distributions, the mean and standard deviation in the case of the normal distribution, or the discrete probability values of the discrete distributions. The update biases the distributions to increase the probability of sampling, in future iterations, the parameter values in the best configurations found.

After new configurations are sampled, the best configurations are selected by means of racing. Racing was first proposed in machine learning to deal with the problem of model selection [118]. Birattari et al. [17] adapted the procedure for the configuration of optimisation algorithms. A race starts with a finite set of candidate configurations. At each step of the race, the candidate configurations are evaluated on a single instance. After each step, those candidate configurations that perform statistically worse than at least another one are discarded, and the race continues with the

remaining surviving configurations. This procedure continues until reaching a minimum number of surviving configurations, a maximum number of instances that have been used or a predefined computational budget. This computational budget may be an overall computation time or a number of experiments, where an experiment is the application of a configuration to an instance.

2.5 Experiments

2.5.1 Evaluation setting

Our experiments are based on the ImageCLEF 2011 medical modality classification data set [91], where there are 988 images in training- and 1024 images in the testing data set, which were taken from PubMed articles. The data set defines 18 different modality classes.

Table 2.2 shows the imbalanced distribution of the images within the various modality classes.

First we compare the performance of using only visual features with different kernels. In the first case we used the SIFT features in a BoW image representation using hard quantisation with the other predefined visual features in the previous section. In the second case instead of the hard quantised SIFT features, we used the ASIFT features in a BoW image representation using the described sparse coding. Table 2.3 shows that in both of the cases the Jensen-Shannon divergence based kernel was the best performing one between the three different kernels and that using ASIFT with sparse coding yields to a better accuracy.

Now we show the influence of each of the proposed methods in the previous section on combining multiple, heterogeneous data sources. Hence, we use the best performing feature and kernel combination of the visual features and combine them with the proposed textual features (caption text and the MeSH terms).

Table 2.4 shows the correctly classified percentage for each proposed method and, as a comparison, we included the result of the best submitted

Modality label		Training	
Code	Description	#	%
AN	angiography	11	1.1
CT	computed tomography	70	7.1
MR	magnetic resonance imaging	17	1.7
US	ultrasound	30	3.0
XR	X-ray	59	6.0
FL	fluorescence	44	4.5
EM	electronmicroscopy	16	1.6
GL	gel	50	5.1
HX	histopathology	208	21.1
PX	general photo	165	16.7
GR	gross pathology	43	4.4
EN	endoscopic imaging	10	1.0
RN	retinograph	5	0.5
DM	dermatology	7	0.7
GX	graphs	161	16.3
DR	drawing	43	4.4
3D	3D reconstruction	32	3.2
CM	compound figure	17	1.7
Total		988	100.0

Table 2.2: Modality labels at ImageCLEF 2011 and their distribution

run [48] for the ImageCLEF 2011 medical modality classification challenge as well.

The first three rows of Table 2.4 show the accuracy, where we use the proposed combined kernel (CK) method. As previously, we used the Jensen-Shannon kernel for each of the visual features and for the textual features we used a simple linear kernel.

The single most important improvement when using combined kernels is kernel normalisation (Eq 2.16). As it can be seen from the classification accuracies of the different setups, defining kernels for different features and combining them into one kernel without kernel normalisation, the classifi-

Feature Set	Kernel	Accuracy
SIFT-BoW+f_{hist}+f_{skin}+f_{mean}+f_{lines}	Jensen-Shannon	80.21
	χ^2	77.44
	Histogram Intersection	73.04
ASIFT-BoW+f_{hist}+f_{skin}+f_{mean}+f_{lines}	Jensen-Shannon	82.81
	χ^2	80.66
	Histogram Intersection	79.4
Fisher Vector [48]	SMLR	83.59

Table 2.3: Modality classification accuracy based on visual features. For the reference we have included the best performing run of the competition.

Method	Accuracy	Cls.Time (ms)
Weighted CK (#1)	**88.47**	189 ± 13
Unweighted CK (#2)	87.69	179 ± 1
Unnormalized CK (#3)	81.34	172 ± 2
HKM (#4) [**Vedaldi:2011ct**]	82.03	1.3
SLR with Fisher Vector [48]	86.91	N/A

Table 2.4: Impact of the proposed approaches on accuracy using heterogeneous features for medical image modality classification. For the reference we have included the best performing run of the competition. "CK" = Combined Kernels using non-linear SVM. HKM = Homogeneous Kernel Mapping using linear SVM [78]. The "Cls.Time" column shows the average classification time of a sample of the test set in milliseconds.

cation accuracy significantly drops.

As soon as the kernel normalisation is used, without even defining any weights for the kernels (i.e. $\forall m\ \beta_m = 1.0$), the average classification of a carefully selected—or by using an automatic algorithm configuration tool—kernel combination for the various features exceeds the best submitted run for the classification challenge.

The system's accuracy can be further improved by fine-tuning the β_m weights of the kernels. In this case the best accuracy can be achieved

Modality class	Ratio (%)		Method			
	train	test	#1	#2	#3	#4
3D : 3D render	3.2	4.4	**80.0**	77.78	64.45	60.0
AN: Angiography	1.1	0.9	77.78	77.78	55.56	55.56
CM: Compound figure	1.7	2.0	15.0	**20.0**	15.0	10.0
CT : Computed tomography	7.1	8.1	95.18	95.18	95.18	96.38
DM: Dermatology	0.7	1.5	6.67	6.67	0.0	0.0
DR: Drawing	4.4	7.2	**74.02**	68.83	68.83	67.53
EM: Electronmicroscope	1.6	1.8	44.44	**50.0**	5.0	5.56
EN : Endoscope	1.0	1.1	72.72	72.72	18.0	18.18
FL : Fluorescence	4.5	2.7	**100.0**	96.42	89.28	92.85
GL : Gel	5.1	4.9	**100.0**	98.0	92.0	96.0
GR: Gross pathology	4.4	3.1	34.37	**37.5**	25.0	28.12
GX: Graphics	16.3	16.8	98.83	98.83	97.67	96.51
HX: Histopathology	21.1	19.0	**99.48**	98.97	97.94	97.94
MR: MRI	1.7	2.0	70.0	**75.0**	20.0	45.0
PX: Photo	16.7	13.8	**95.74**	94.32	87.23	88.65
RN: Retiongraph	0.5	0.3	66.7	66.7	66.7	**100.0**
US : Ultrasound	3.0	4.0	**95.12**	92.68	85.36	85.36
XR: X-ray	6.0	6.5	**95.52**	94.03	88.06	88.06

Table 2.5: Performance of the proposed methods broken down for the individual classes.

by setting the weights of the linear kernels (the ones used for the textual features) to 0.5 each. To our knowledge, this is the best result reported to date on ImageCLEF 2011 medical modality classification.

By observing the per class accuracy of the different methods in Table 2.5 it is important to note that although the weightening improves the classification accuracy in case of eight classes compared to the uniformly weighted case, it accuracy degrades in four other classes. This degradation is typically in cases of classes with very few train and test samples.

And at last, the accuracy of the linear SVM using homogeneous kernel map of the features is neither near to our best accuracy nor managed to go beyond the accuracy of the best run of the challenge, but again one

should see that this result has been achieved by using a simple linear SVM classifier. Thus, the training and classification time is significantly lower than the non-linear SVM ones'. It is important to point out that with the homogeneous kernel mapping the performance is better than the combined kernel case without kernel normalisation, which again demonstrates the importance of using kernel normalisation.

CHAPTER 3

Organ Detection

Nem, az agyvelőm nem fáj. Fáj? Hát nem kietlenebb ez, mintha fájna? Inkább fájna. Ijesztőbb minden fájásnál, hogy valószínűtlen. Valószínűtlen, hogy az ember itt feküdjék az asztalon, felbontott koponyával, agyveleje a külvilágban - valószínűtlen, hogy itt fekhessék élve -, valószínűtlen, nem való ez, illetlen ez, hogy mégis él - nemcsak hogy él, ébren van és gondolkodik.

— Karinthy Frigyes, *Utazás a koponyám körül*

The segmentation of organs is one of the major tasks in medical imaging. Most of the segmentation algorithms operate locally, i.e. they depend on a manual placement of seed points in order to successfully segment the desired structure. Thus, an automatic organ segmentation system, which is capable to segment various organs without manual labour, requires an *organ detection* model that can automatically provide those seeding points for the Image Segmentation (IS) model.

Another possible and interesting application of organ detection is the efficient retrieval of selected parts of patient scans from radiological databases. In case a physician would like to examine a particular organ, the ability to

determine its position and extent automatically means it is not necessary to retrieve the entire scan but only a small part of it.

Currently, most of the organ detection methods are tailored to a specific application. For example, Kainmüller et al. [90] detect the liver by searching for the right lung lobe, which can be relatively easily detected by thresholding and voxel counting. The problem of such approaches is that they do not generalize well to other structures or image modalities. Learning-based approaches such as Discriminative Generalized Hough transform [151] and Marginal Space Learning (MLS) [109, 187] are more general and can be adapted to a wide variety of detection tasks by simply exchanging the training data.

Jung et al. [89] propose a learning-based organ detection approach for 3D medical images based on the Viola-Jones object detection algorithm [175]. They use a bootstrapping approach to automatically select important training data and propose several extensions to the original approach tailored to medical images, namely a new pure-intensity feature type, and a multi-organ detection that exploits spatial coherence in medical data.

Criminisi et al. [46] introduced a new parametrisation of the anatomy localisation task as a continuous multivariate parameter estimation problem. This is addressed effectively via non-linear regression, in the form of regression forests [45, 141]. The approach is fully probabilistic and, unlike other techniques [58, 188], maximizes the confidence of output predictions. The method yields salient anatomical landmarks, i.e. automatically selected "anchor" regions that help localize organs of interest with high confidence. The algorithm can localize both macroscopic anatomical regions, like abdomen, thorax, etc. and smaller scale structures like heart, left adrenal gland, etc. using a single model.

In this chapter we first introduce the reader with the concept of structured output prediction, in particular we focus on the generalisation of the Support Vector Machine (SVM) classifier, that allows training of a classifier for general structured output labels. In Section 3.2 we show how structured output prediction can be employed to construct a multi-organ detection model. Finally, we present a novel automatic IS framework based

on this multi-organ detector and Deformable Models (DMs).

3.1 Structured output prediction

Structured output prediction [136] is defined as task of predicting elements of a complex interdependent output space that correspond to a given input. Recent years have witnessed its increasing popularity in machine learning due to its ability to provide an elegant formulation for several applications in medical image analysis [13, 14, 185], computer vision [20, 150, 146], natural language processing [97, 108] and many other areas of research.

A central problem in structured output prediction is to learn the parameters of the prediction model. There are two paradigms for parameter estimation: probabilistic modeling and empirical risk minimisation. In the former paradigm, a parameter estimate is made by means of the marginal likelihood function of the observed data. Here the model is usually represented by means of a parameter point estimate [19, 95] or an approximation to the posterior distribution over the model parameters [19, 21, 125, 132]. While in case of empirical risk minimisation the parameters are learned by minimizing the regularized empirical risk, where the risk is measured by a user-specified loss function. In this thesis, we will focus on the empirical risk minimisation approach.

Research has been mostly focused on developing supervised learning techniques for structured prediction. It is assumed that each sample of the training has a fully observed ground-truth annotation. A well-known example of such a learner is the Structured Output Support Vector Machine (SOSVM) [165, 167].

3.1.1 Structural output support vector machine

Given an input vector $\mathbf{x} \in \mathcal{X}$, a SOSVM provides a linear prediction rule for the structured output $\mathbf{y} \in \mathcal{Y}$. Let us denote the joint feature vector of an input $\mathbf{x}$ and an output $\mathbf{y}$ by $\Psi(\mathbf{x}, \mathbf{y})$. This joint feature vector captures the relationship between the input and the output. In order to understand

the role of the joint feature vector let us consider an example application of SOSVM: detecting an object in an image. For convenience, let us assume that an image can contain at most one instance of the object. In this application, the input $\mathbf{x}$ is the image, the output $\mathbf{y}$ is the bounding box of the object for a positive image, i.e. an image containing the object and $\varnothing$ for a negative image, an image not containing the object. The joint feature vector $\Psi(\mathbf{x}, \mathbf{y})$ can be the Histogram of Oriented Gradients (HOG) descriptor of the bounding box specified by $\mathbf{y}$ for positive images, and $\mathbf{0}$ for negative images.

The prediction rule of an SOSVM parameterized by $\mathbf{w}$ can be formulated as the following optimisation problem:

$$\mathbf{y}(\mathbf{w}) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}}\, \mathbf{w}^\intercal \Psi(\mathbf{x}, \mathbf{y}) \tag{3.1}$$

where $\mathbf{w}^\intercal \Psi(\mathbf{x}, \mathbf{y})$ is the score of an output $\mathbf{y}$ for a given input $\mathbf{x}$. In the above introduced object detection example, given an image $\mathbf{x}$, the SOSVM will either return a bounding box $\mathbf{y}$ if there exists a bounding box with a score greater than 0, or it will return ϕ.

The problem of learning an SOSVM is to estimate the parameter $\mathbf{w}$ from a training data set. In this chapter, we will assume that the training data set is fully supervised, i.e. the complete ground-truth annotation of each training sample is known. Formally, we denote the training data set by $\mathcal{S}_{train} = \{\mathbf{s}_i = (\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, n\}$. Furthermore, we assume a user-specified loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, which is used to measure the risk of prediction. In other words, $\Delta(\mathbf{y}_1, \mathbf{y}_2)$ measures the difference between the outputs $\mathbf{y}_1$ and $\mathbf{y}_2$. The value of the loss is assumed to be non-negative and equal to 0 if and only if $\mathbf{y}_1 = \mathbf{y}_2$. In the above introduced application of SOSVM, an appropriate loss function could be one that returns 0 if both of the outputs belong to the same class—either both consider the image $\mathbf{x}$ to be positive, or both consider $\mathbf{x}$ to be negative—, and 1 otherwise.

The parameter $\mathbf{w}$ is learned by minimizing a regularized upper bound on the empirical risk, where the empirical risk is defined as

$$\min_{\mathbf{w}} \sum_i \Delta(\mathbf{y}_i, \mathbf{y}_i(\mathbf{w})) \tag{3.2}$$

where $\mathbf{y}_i(\mathbf{w})$ refers to the prediction made using the parameters $\mathbf{w}$ as specified in Ex. 3.1. More formally, the learning of parameter $\mathbf{w}$ involves solving the following convex Quadratic Programming (QP):

$$\begin{aligned} & \min_{\mathbf{w},\xi_i \geq 0} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ \text{s.t. :} \quad & \mathbf{w}^\intercal\left(\Psi(\mathbf{x}_i,\mathbf{y}_i) - \Psi(\mathbf{x}_i,\hat{\mathbf{y}}_i)\right) \geq \Delta(\mathbf{y},\hat{\mathbf{y}}_i) - \xi_i \quad \forall \hat{\mathbf{y}}_i \in \mathcal{Y}, i = 1,\ldots,n \end{aligned} \tag{3.3}$$

where $C \geq 0$ is a user-specified constant, and ξ_i is the slack variable corresponding to the sample $\mathbf{s}_i$. Intuitively, the above problem introduces a margin between the score of the ground-truth output and the score of any other output. The desired margin is proportional to the loss value between the ground-truth and the corresponding output.

Even though the above problem seems to be a difficult optimisation task due to a large number of constraints, there exist several polynomial-time algorithms that obtain a solution up to an arbitrary precision ϵ [165, 167, 153, 87].

3.2 Organ detection as a structured output prediction

Formally, let I be an image, $p_i = (x, y)$ the pixel location of part i, and t_i the mixture component of part i, where $i \in \{1,\ldots,K\}$, $p_i \in \{1,\ldots,L\}$ and $t_i \in \{1,\ldots,T\}$. t_i is the "type" of part i.

In order to determine the score of a configuration of parts, i.e. how the parts are distributed within the image and where exactly they are located, we first define a compatibility function for part types that factors into a sum of local and pairwise scores:

$$S(t) = \sum_{i \in V} b_i^{t_i} + \sum_{ij \in E} b_{ij}^{t_i,t_j}. \tag{3.4}$$

The parameter $b_i^{t_i}$ favors particular type assignments for part i, while the pairwise parameter $b_{ij}^{t_i,t_j}$ favors particular co-occurrences of part types. Let us define a $G = (V, E)$ for a K-node relational graph whose edges specify

which pairs of parts are constrained to have consistent relations. One can now write the full score associated with a configuration of part types and positions:

$$S(I,p,t) = S(t) + \sum_{i \in V} w_i^{t_i} \cdot \phi(I,p_i) + \sum_{ij \in E} w_{ij}^{t_i,t_j} \cdot \psi(p_i - p_j), \tag{3.5}$$

with $\phi(I,p_i)$ being a feature vector extracted from pixel location p_i in image I and $\psi(p_i - p_j) = [dx\ dx^2\ dy\ dy^2]^\intercal$, where $dx = x_i - x_j$ and $dy = y_i - y_j$ are the relative location of part i with respect to j.

Learning Assuming a supervised learning paradigm, with given labeled positive examples I_n, p_n, t_n (image, part, type) and negative examples I_n, one can define a structured prediction objective function. In order to do so, let us define $z_n = (p_n, t_n)$ and note that the scoring function 3.5 is linear in model parameters $\beta = (w,b)$, and so can be written as $S(I,z) = \beta \cdot \Psi(I,z)$. A model would be learned of the form:

$$\begin{aligned} &\underset{w,\xi_i \geq 0}{\operatorname{argmin}} \frac{1}{2}\beta^2 + C\sum_n \xi_n \\ \text{s.t. } &\begin{cases} \forall n \in \text{pos} & \beta \cdot \Psi(I_n, z_n) \geq 1 - \xi_n \\ \forall n \in \text{neg}, \forall z & \beta \cdot \Psi(I_n, z) \leq -1 + \xi_n \end{cases} \end{aligned} \tag{3.6}$$

The above constraint states that positive examples should score better than 1 (the margin), while negative examples, for all configurations of part positions and types, should score less than -1. The objective function penalizes violations of these constraints using slack variables ξ_n.

Inference corresponds to maximizing $S(I,p,t)$ over p and t. When the relational graph $G = (V,E)$ is a tree, this can be performed efficiently by dynamic programming. Let kids(i) be the set of children of part i in G. One can compute the message part i passes to its parent j by the following:

$$\begin{aligned} \text{score}_i(t_i, p_i) =& b_i^{t_i} + w_{t_i}^i \phi(I, p_i) + \sum_{k \in \text{kids}(i)} m_k(t_i, p_i) \\ m_i(t_j, p_j) =& \max_{t_i} b_{ij}^{t_i, t_j} + \\ & + \max_{p_i} \text{score}(t_i, p_i) + w_{ij}^{t_i, t_j} \cdot \psi(p_i - p_j), \end{aligned} \tag{3.7}$$

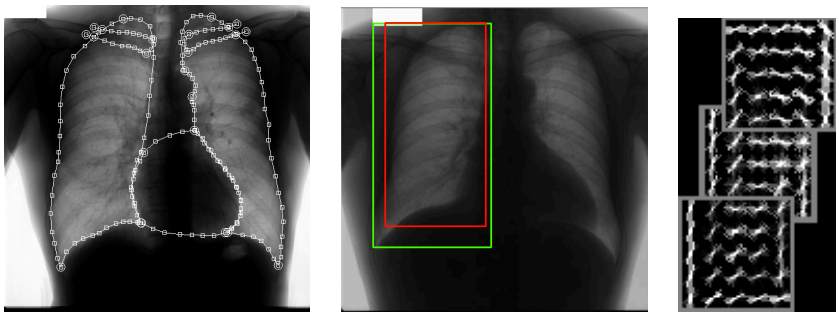
where the first equation computes the local score of part i, at all pixel locations p_i and for all possible types t_i, by collecting messages from the children of i. Ex. 3.7 computes for every location and possible type of part j, the best scoring location and type of its child part i. Once messages are passed to the root part ($i = 1$), $\text{score}_1(t_1, p_1)$ represents the best scoring configuration for each root position and type. One can use these root scores to generate multiple detections in image I by thresholding them and applying non-maximum suppression.

3.3 Experiments

In order to evaluate the organ detection model, we required a medical image data set with organ annotations. Since data set in this form was not publicly available we had to define our own. We used two popular medical image data sets for organ segmentation, which include the Ground Truth (GT).

3.3.1 SCR database: lungs segmentation in chest radiographs

The Segmentation in Chest Radiographs (SCR) database [63] has been established to facilitate comparative studies on segmentation of the lung fields, the heart and the clavicles in standard posterior-anterior chest radiographs. All chest radiographs are taken from the Japanese Society of Radiological Technology (JSRT) database [156], which is a publicly available database of 247 PA (that is, front view) chest radiographs. In each image the lung fields, heart and clavicles have been manually segmented to provide a standard reference (see Figure 3.1a).



(a) Manual segmentation of an image of the JSRT database. Lung fields, the heart, and the clavicles are delineated.

(b) Green rectangle is the ground truth, the red rectangle is the model's prediction.

(c) Three parts star mixture model for right lung detection.

Figure 3.1: Right lung detection in chest radiographs.

Using the model introduced in section 3.2 we defined a three parts mixture star model for detecting the right lung. For the root part position we choose to use the centroid of the segmentation contour and as for the two children parts' positions we used to uppermost and bottommost landmark points of the provided segmentation.

For the training set we used 78 examples of the SCR database as positive examples and 300 randomly selected images from the ImageCLEF 2011 [91] data set as negative examples. Figure 3.1c shows the learned three part mixture model.

In order to evaluate the learned model's accuracy we used the remaining 169 samples of the SCR data set. Based on the segmentation contours, one can easily define a bounding box for each organ. Using this as the ground truth and the output bounding box of our model we used the Jaccard index [81] to calculate the system's accuracy, which is defined in the following way:

$$J = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}, \tag{3.8}$$

where TP is true positive area (correctly classified as object), FP is false positive area (classified as object, but in fact background), FN is false negative area (classified as background, but in fact object), and TN is true

(a) Half resolution (b) Noisy (c) Combined modalities

Figure 3.2: Examples of right lung detection on random x-ray chest images.

negative area (correctly classified as background).

Using this measure the average accuracy for right lung detection was **69.4%**.

In order to test the system's robustness we have randomly selected X-ray chest images from Radiopaedia[1] as well as from Google's image search for *x-ray lung* keyword. Figure 3.2 shows three interesting examples from this data set. As one can see in figure 3.2c the model can detect multiple instances of the right lung within the same image as well.

3.3.2 Allen Brain Atlas

The Allen Brain Atlas (ABA) [3] contains a genome-scale collection of histological images (cellular resolution gene-expression profiles) obtained by In Situ Hybridisation of serial sections of mouse brains. There is great interest in automated methods to accurately, robustly, and reproducibly localize the hippocampus in brain images, after discoveries which established its role as an early biomarker for Alzheimer's disease and epilepsy [11].

The anatomical structure to segment was the hippocampus and the GT was created manually by an expert in molecular biology. Every image was manually segmented 5 times and, for each group of 5 manual segmentations, the consensus image was calculated and used as GT. The typical resolution of ABA images is about 15,000 × 7,000 pixels. Thus, in order to have an

[1] http://radiopaedia.org/tags/lung

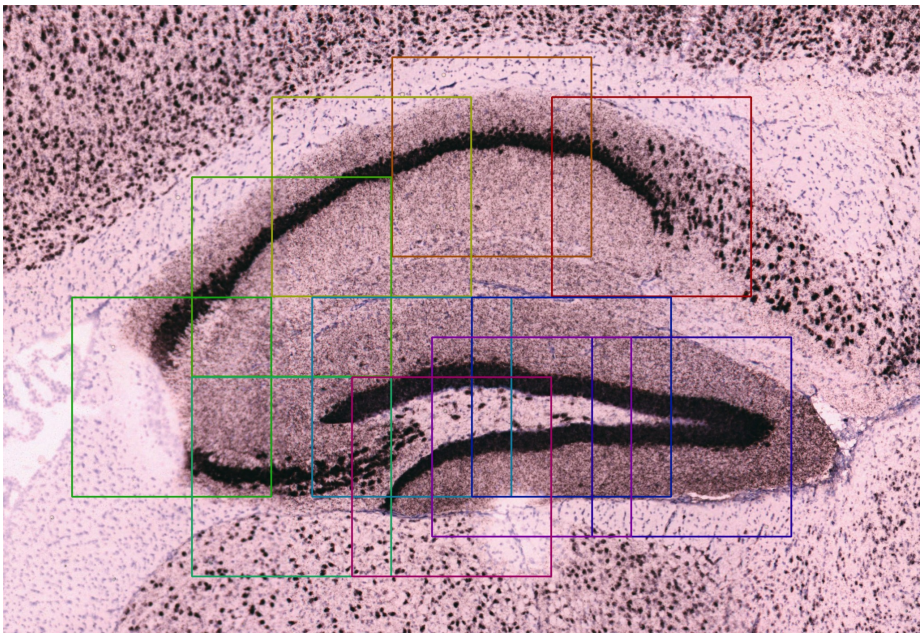
Figure 3.3: Section of a mouse brain from ABA data set. The red bounding box marks the hippocampus.

accurate, robust and fast system for segmenting the hippocampus without loosing information by rescaling the image, a system that could provide a rough location of the hippocampus within the image can both lower the computational time and raise accuracy—by avoiding false positives—of a fine grained segmentation algorithm.

For the organ detection model we have defined a graph of eleven nodes that covers the whole hippocampus (see Figure 3.4b). For the training set we used the five segmentation GT, and manually labeled the eleven parts of the hippocampus. As for the test set, we randomly selected 20 images of the data set. We used simple manual verification for testing the model's accuracy, i.e. checked whether the detected parts in the whole image are actually covering the hippocampus.

Only by using so few training examples we have created a model which is able to automatically find wherein the image the hippocampus is roughly located in all the test images. Figure 3.4a and 3.4b shows an example of the detected parts.

(a) The detected eleven parts of the hippocampus in the whole image.



(b) Cropped image of the hippocampus with the detected parts.

Figure 3.4: Allen Brain Atlas

3.4 A general framework for deformable models supervised guidance

In this section we present a novel automatic IS framework based on Machine Learning (ML) and DMs. The input of our system is a Dataset of Images (DI). Each image in the DI has an associated GT binary image which partitions the original image I in two regions: *foreground* (FG), delineating the segmentation target, and *background* (BG), such that $\mathrm{GT} = \mathrm{FG} \cup \mathrm{BG}$ and $\mathrm{FG} \cap \mathrm{BG} = \emptyset$. The output is a ML model or, rather, a set of models, able to segment a data set of images similar to the ones in DI, thus having generalisation capabilities. Being a general purpose framework, it could be applied to any kind of image, modality, and target structure.

The framework consists of several components and the connections among them. In this case, the sophistication comes from the choice of the components and the way they are interconnected. The opportunity of examining different designs allows the exploration of strategic possibilities that may prove effective in a particular application or image modality. The framework components are:

- *Localiser*: any method that is capable of detecting a Region Of Interest (ROI) within an image. Possible choices are, for instance, model-based detection systems or registration algorithms.
- *Deformable Model*: any DM model (e.g. Level Set (LS)) whose final position within the image domain represents the segmentation result.
- *Term set*: a set of image descriptors (like Haralick features, HOG or Local Binary Pattern (LBP)) and DM-related features (like local curvature). Basically any kind of relevant feature can be part of the term set.
- *Driver*: a ML-based classification or regression method whose output directly guides the DM evolution.

The framework components are not connected in a pipeline-like manner, but rather they are tightly interconnected, in a way also dependent on the

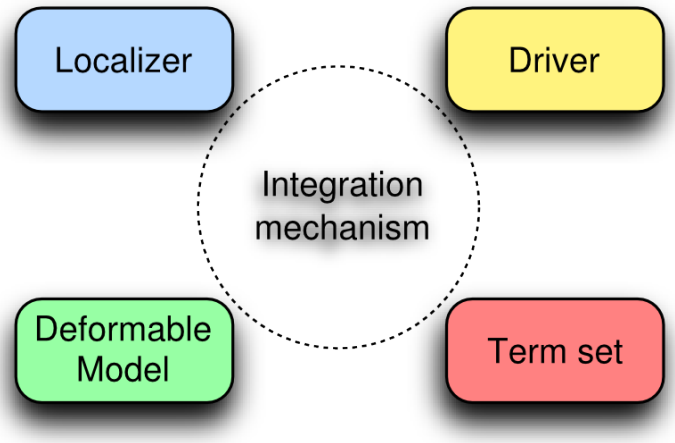


Figure 3.5: The framework overall scheme.

implementation. We call *integration mechanism* the way the framework components are connected to each other. Fig. 3.5 shows the framework general scheme.

A key aspect of the framework is the role of the *driver*. While in the vast majority of the works in literature DM evolution is tackled as an energy minimisation problem, in our proposal the driver directly guides the evolution of the model using a model derived from a ML method. The driver is intended as a general purpose ML tool able to drive the DM toward its final position, the target object. It makes decisions considering the values assumed by the features in the term set. These features comprise image related cues, the local and global status of the DM, or other information from different sources, as the output of the localiser. The prediction calculated by the driver will affect the evolution of the DM in a way dependent on the specific framework instantiation. In particular, it is strongly dependent on the nature of the DM employed.

As the driver is a general purpose, supervised ML tool, a training phase is needed. It has to be carried out by means of a specific procedure to infer a function from a labeled set of training examples. In supervised learning, each example is a pair consisting of an input vector and a desired output value, called the label. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used

for predicting new examples (see Section 1.2). However, in our case, the input of the framework, the DI, is a data set of images and associated GTs, not a standard plain set of vector-label pairs, as expected by many ML algorithms. Therefore, it is necessary to construct a proper data set of vector-label pairs, which we name Image Vector-Label Dataset (IVLD). Each example in the IVLD will be made up of a vector of the values the terms in the *term set* assume in a given location of the DM in the image, and a (continuous or discrete) label indicating the behavior the DM should show in that specific case.

In general, the values assumed by the terms in a given image point depend on both the image itself and the status of the DM in that point. This implies that the use of an evolving DM is mandatory to generate the IVLD. It is not possible to generate the IVLD *a priori*, directly from the image itself. Moreover, the meaning and the type of label are strictly dependent on the structure of the employed DM. For instance, geometric and parametric DMs will need a different class of labels to reflect the different ways they evolve. While the former evolve (that is, move) only along the line perpendicular to the contour (in a given point), the latter can move toward any direction.

To generate the IVLD, we exploit GT information by forcing the DM to evolve in a way that, in its final state, will make it perfectly cover FG, as defined earlier. In other words, each label in an IVLD example will be the one that is expected to guide the DM toward a segmentation identical to the GT. For instance, if the employed DM is a LS, the label would be positive (that is, the DM grows) for each point p of the LS being over the foreground FG while it would be negative (that is, the DM shrinks) if p is over the background BG. A more complex label set is expected in the parametric DM case, as each control point can, in principle, be moved toward any direction. Depending on the DM initialisation, different evolution trajectories can be produced. By performing the IVLD creation starting from different initialisations, we can generate examples from different areas of the images, increasing diversity in the IVLD.

Once the IVLD has been generated, the driver can learn a classifier

or model from it using any supervised ML technique. The output of the learning is, in an optimal scenario, a ML model with the ability to correctly determine labels for unseen instances. We call this the Guide Model (GM). During the prediction phase, that is, when unseen images are segmented, the *driver* constructs a feature vector for each point of the DM by calculating the values of the features of the *term set* (in the same way it was done for the creation of IVLD), then applies the GM to this vector and calculates the label responsible for the DM movement in that specific point. The process stops as soon as the termination criterion is satisfied. This condition depends on the specific DM employed.

Different choices are possible for each component of our framework. This fact provides flexibility, as it can be implemented in a variety of ways and degrees of sophistication. As the components are interconnected, the decision might affect the integration of the components to some extent.

In the framework, the role of the *localiser* is to detect a ROI within the image. While it is not strictly necessary to perform segmentation, it effectively reduces the search space, allowing the *driver* to focus on the relevant areas. Moreover, it provides the capability of discerning similar objects on the basis of their location in the image space (e.g. recognizing one of the two lungs). In any case, the information dispensed by the *localiser* can be exploited in different forms, depending on its nature. For instance, if the localiser was a part-based method, the provided information would be in the form of the coordinates (and, possibly, the size) of the model parts in the image space. Differently, a registration-based localiser would provide a somewhat more comprehensive kind of information in terms of shape definition and accuracy of the localisation. However, compared to a part-based model, a registration localiser could be more sensible to noise and high variance in shapes, apart from needing much higher computation times. Moreover, the choice of the localiser would influence the initialisation of the DM and the features of the *term set*. In fact, in the case of a registration-based localiser, the DM could be straightforwardly initialized with the result of the registration step. Moreover, a feature of the *term set* could be the minimum distance between the registration result and the

evolving DM. Conversely, in the case of a N part-based localiser, the DM could be initialized by dividing it in N chunks located over the localiser parts. In this case, a feature of the *term set* could be the distance between a point in the DM and the closest localiser part center.

The choice of the DM is also very relevant in the framework. On the one hand, geometric DMs provide the advantage of being able to easily change topology and, therefore, are suitable to segmentation problems with high variability in shape. On the other hand, parametric DMs are faster than their geometric counterparts and, while tackling topological changes less easily, they can effectively deal with strong shape constraints.

Being the *driver* a general purpose ML tool, a plethora of different learning algorithms is available (see Section 1.2). However, the nature of the employed DM can constrain the available drivers. A common LS, for instance, is guided by a certain speed function, with a real valued result. To calculate this velocity, a regressor would be more appropriate than a classifier. Differently, for parametric DMs the desired output could be the destination pixel of a DM control point. This problem can be dealt with a classification algorithm. Apart from what has been explained so far, there is no clear advantage in using a specific ML method, with a wide range of possibilities. However, given the possible large size of the IVLD, fast algorithms are preferred, in both training and prediction phases.

Finally, the features in the *term set* are strongly dependent on the characteristics of the DI. In fact, in some data sets, a set of features is more discriminative than others. For example, if the goal is segmenting structures in satellite images, the colour would be very relevant, as some structures (e.g. forests) are best described by it. Conversely, in medical imaging, where colour is mostly absent, other features such as edges and textures are best suited at discriminating different tissues and organs.

In general, the choice of the most appropriate components is dependent on the IS problem at hand. Being able to easily replace the framework components to tailor the system to specific data sets is one of the key strengths of the proposed framework.

3.5 A specific implementation of the framework for medical image segmentation

As said earlier, there is a vast plethora of design possibilities when implementing a segmentation system within the proposed framework. To illustrate this capability, we tailored the system to segment diverse structures within different medical image modalities. Therefore we integrated the components focusing on edge and texture image descriptors along with shape. We chose a part-based localiser able to capture the variability of the forms and quickly recognize different parts of the target structures. We opted for the Shi LS (see Section 1.3) [155], a flexible, fast DM, able to handle complex topologies and simple to control. In fact, its evolution algorithm only considers the sign of the speed function value, thus allowing to treat that calculation as a classification problem. Hence, we chose an extended, quick, accurate and easy to train classifier, Random Forest (RF) [30] (see Section 1.2.2.4). The decision to choose fast components was motivated by the large size of one of the data sets to be considered.

In the following sections, we provide insight for the design of every component in the framework to customize it to our medical IS problem.

3.5.1 Localiser

The localisation method for medical images are mostly tailored for a specific application. For our proposed framework it is more desired to have a more general, universal localiser that could be used for locating any pattern type in an image, after customising it to the specific problem.

The method introduced in Section 3.2 is a good candidate for such a localiser.

3.5.2 Deformable Model

As already introduced in the previous section, the output of the driver, that is, the meaning and number of prediction labels, depends on the employed DM.

We decided to use a simple and fast LS implementation: the Shi approximation [155] (see Section 1.3). This LS, while retaining the same attractive qualities of standard LSs in terms of accuracy and flexibility, is way faster than them and is easy to control. In fact, with this LS, only two values of the speed function F are significant with respect to the curve evolution: positive and negative. In the former case, the LS will grow (in the direction normal to the contour, as usual), while in the latter it will shrink. Since the absolute value of the speed is irrelevant, we can effectively treat the calculation of the speed function as a binary classification problem, with the obvious advantage of reducing complexity. If a different LS was to be used, the absolute value would be, indeed, relevant. In this case, the driver would rather provide a real value and the calculation of the speed would be more appropriately tackled as a regression problem. Apart from the calculation of the speed function, which in our framework is performed by the driver, every other aspect of the LS evolution is kept as in its original proposal [155], including the *smoothing cycle*.

3.5.3 Terms

As shown in Section 1.1, many different features can be used to design term for the LS evolution. Apart from edges, that are local in this case, the texture information can be considered at different levels. We distinguish among statistics calculated over the entire image and over local portions of it, centered on the LS contour. Indeed, the border of the DM allows us to calculate the texture features also distinguishing between inside and outside the contour, at local level. Finally, other terms are relative to the shape of the contour, both at global and local level. These are due to the localiser and the LS local appearance, respectively. In the following sections we will define which image descriptors and shape related features are included in

our reference implementation.

3.5.3.1 Haralick

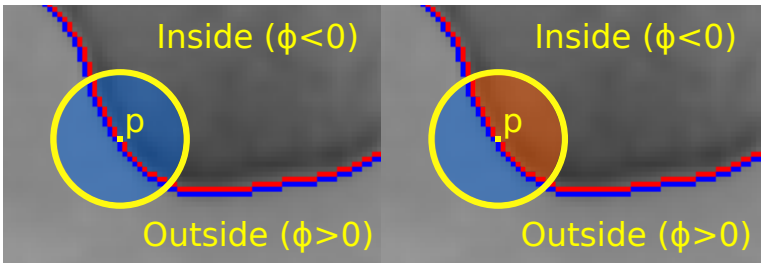
In Section 1.1.1, we introduced the Haralick image descriptor. There, a GLCM offset distance of 1 was chosen. It was shown in literature [6] that the higher these values, the higher the values dispersion around the main diagonal. This leads to a random-like matrix, caused by the reduction of the spatial correlation among pixels. We decided, therefore, to keep the offset distance value to 1.

The first order statistical parameters consider point-wise values. Their computational complexity is dependent on the computation of the intensity histogram, which is linearly dependent on the size of the area on which the calculation is performed.

A common image depth in textural analysis is 256 gray levels (1 byte per pixel). For the calculation of second order statistics, this would result in a 256×256 Gray Level Co-occurrence Matrix (GLCM) matrix. Hence, apart from the area size, complexity depends on the square of image depth. This is different from first order statistics, as they have a linear dependence on that value. Moreover, such a large matrix would probably be composed of a large number of 0-valued cells, making it a bad approximation of a probability density. Therefore, we decided to quantize intensity information reducing the number of levels to 32, giving rise to 32×32 GLCMs.

The Haralick features values assume different results depending on the areas the calculation is performed on. Thus, we implemented three different procedures to calculate the Haralick statistics which are defined as follows.

Plain Haralick This is the most straightforward option. In this case, we define a *closed ball* of radius r, centered at a given point p of the LS contour. We calculate the Haralick statistics introduced in Section 1.1.1 on the area inside the border of the ball, that is, the light blue area of Fig. 3.6(a). To take into account image features at different scales, we define two balls of different sizes, r_{small} and r_{big}. The values of these two constants are parameters of our algorithm.



(a) Plain Haralick (b) Split Haralick

Figure 3.6: The plain and split Haralick terms.

Split Haralick This term is similar to the previous one. However, in this case, we calculate the Haralick statistics in two separate areas inside the ball: inside and outside of the LS, that is, the orange and light blue areas of Fig. 3.6(b), respectively.

Global Haralick This term is similar to the split Haralick one. However, in this case, the features are calculated on the whole internal and external areas of the LS.

3.5.3.2 Local Chan and Vese

This is a local version [102] of the widely employed Chan and Vese term [36]. The formulation is $\text{cv}_{\text{local}} = (I - \mu_{\text{ext}})^2 - (I - \mu_{\text{int}})^2$, where I is the intensity value of the pixel p, and μ_{ext} and μ_{int} are the mean intensity values of the image areas enclosed by the same balls used for the Haralick terms, outside or inside of the LS, respectively.

3.5.3.3 Gabor

In our implementation we consider the image response to the application of 8 different filters, one for each direction. The size of the complex wave employed for the calculation is fixed at 21 pixels, for a total of 8 different features.

3.5.3.4 LBP

A LBP is a string of bits obtained by binarising a local neighborhood of pixels with respect to the brightness of the central pixel. With a neighborhood size of 3×3 pixels, the LBP at location (x, y) is a string of 8 binary values. Therefore, in this case an LBP is a string of 8 bits and so there are 256 possible LBPs. These are usually too many for a reliable statistics (histogram) to be computed. Therefore, the 256 patterns are further quantized into a smaller number of patterns. Considering a uniform quantisation, there is one quantized pattern for each LBP that has exactly a transition from 0 to 1 and one from 1 to 0 when scanned in anti-clockwise order, plus one quantized pattern comprising the two uniform LBPs, and one quantized pattern comprising all the other LBPs. This yields a total of 58 quantized patterns. In all cases, the employed cell size has been 8 pixels. In our implementation the values of the features in the term set are the same for the pixels in the same cell.

3.5.3.5 HOG

The HOG features are widely used for object detection. HOG decomposes an image into small squared cells, computes a histogram of oriented gradients in each cell, normalizes the result using a block-wise pattern, and returns a descriptor for each cell. In our implementation we used the variant introduced in [57]. The main difference is that this variant computes both directed and undirected gradients as well as a four-dimensional texture-energy feature, but projects the result down to 31 dimensions. Also in this case, the employed cell size has always been 8 pixels. In our implementation the values of the features in the term set are the same for the pixels in the same cell.

3.5.3.6 Deformable model-related and local shape

These features are calculated at the same scales (i.e. the same balls) we do for the Haralick ones. They are as follows.

- *Pseudo curvature.* This is a curvature-like feature and is defined as $\text{curv} = \frac{|\text{int}|-|\text{ext}|}{|\text{int}|+|\text{ext}|}$, with $\text{curv} \in [-1, 1]$, where $|\text{ext}|$ and $|\text{int}|$ are the number of pixels in the image areas enclosed by the ball and outside or inside of the LS, respectively.

- *Local perimeter.* This is the length of the LS contour in the image area inside of the given ball. Since the Shi LS that we use in our implementation is made up of two lists, we define this feature as the average of the length of the two lists (inside the ball).

- the Shi LS list in which p is contained: it can be either +1, if $p \in L_{\text{in}}$, or -1, in case $p \in L_{\text{out}}$.

3.5.3.7 Node priors

This term is intended as a way to inject some prior knowledge relative to the structures to be segmented. Given a point p of the LS contour, the node priors term is defined as the relative shift, in the x and y directions, between p and each of the central points of the localiser nodes. Mathematically, it is formulated as:

$$\text{np} = \{x_p - x_{n_1}, y_p - y_{n_1}, x_p - x_{n_2}, y_p - y_{n_2}, \cdots, x_p - x_{n_m}, y_p - y_{n_m}\},$$

where x_p and y_p are the x and y coordinates in the image plane of the point p, respectively, while x_k, y_k are the coordinates of the center point of localiser node k and m is the total number of parts of the localiser model.

3.5.4 Driver

As explained in Section 3.4, the driver is a model derived from a general purpose ML method able to perform supervised learning. Given the nature of the DM we decided to use, we have chosen the driver to be a binary classifier. Among many possible choices, RF (see Section 1.2.2.4) is a very attractive one. Apart from being the most extended classifier ensemble found in literature [96], it is quite fast and easily parallelizable, a strong advantage when dealing with large data sets. Decision trees are not affected

by the modulus of the feature values, a property that other classifiers (like SVM) do not share. Then, being RF a kind of decision tree, it is not necessary to perform any sort of scaling of the values of the used features. This fact sensibly contributes to simplify the implementation of our system.

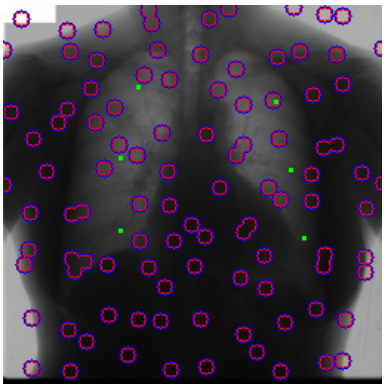
3.5.5 Integration mechanism

As already introduced, generating the IVLD is a key point in the proposed framework. To do so, we had to make some decisions about how to derive it from DI. First of all, the LS must be initialized in the image plane. We decided to perform three different initialisations as follows:

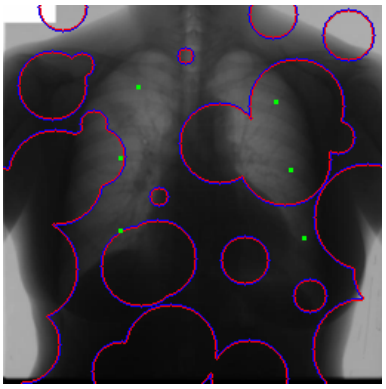
- from small ellipses;
- from big ellipses;
- from the localiser described in Section 3.5.1.

In the first two cases, the LS is initialized as a uniform grid of ellipses in the image plane. The position of each ellipse is shifted by a small random amount in the x and y directions. In the *small ellipses* case, the grid has a 10×10 size and all the ellipses have the same size: $x_e = 0.05 \times X$ and $y_e = 0.05 \times Y$, where x_e and y_e are the ellipse's x and y axes, respectively, and X and Y are the image dimensions. In the *big ellipses* case, the grid has a 5×5 size and the size of the ellipses is uniformly sampled in $[5, \min(X, Y)/8]$. Of course, the LS can gracefully merge some of the ellipses, in case of overlap. Finally, in the third case, the LS is initialized as circles centered in the output points as provided by the localiser. The size of the circles is a fraction of the localiser window square. Fig. 3.7 shows an example of the three LS initialisations.

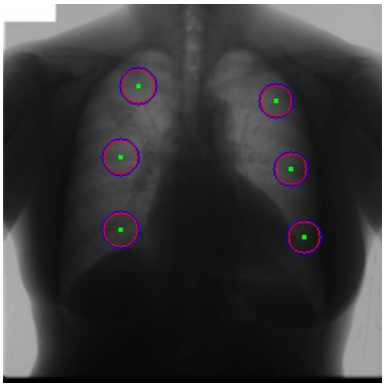
Once the LS has been initialised, it has to evolve toward its final state, corresponding to the target object position in the image plane. Since the GT images are available, the LS evolves taking into account this information.



(a) Initialisation with small ellipses



(b) Initialisation with large ellipses



(c) Initialisation with localiser

Figure 3.7: LS initialisation by means of: (a) small ellipses, (b) big ellipses, (c) localiser. The localiser output points are highlighted in green.

In particular, the LS speed function for a point p is

$$S(p) = \begin{cases} +1, & \text{if } p \in FG \\ -1, & \text{if } p \in BG. \end{cases}$$

This speed function implies that the LS contour will grow (in the direction perpendicular to the contour) in the areas that are included in the target object, while it will shrink in the areas that are included in the background. Fig. 3.8 shows an example of this procedure.

The IVLD generation procedure operates as follows. For each point in the LS contour, at each iteration, the term set values and the corresponding

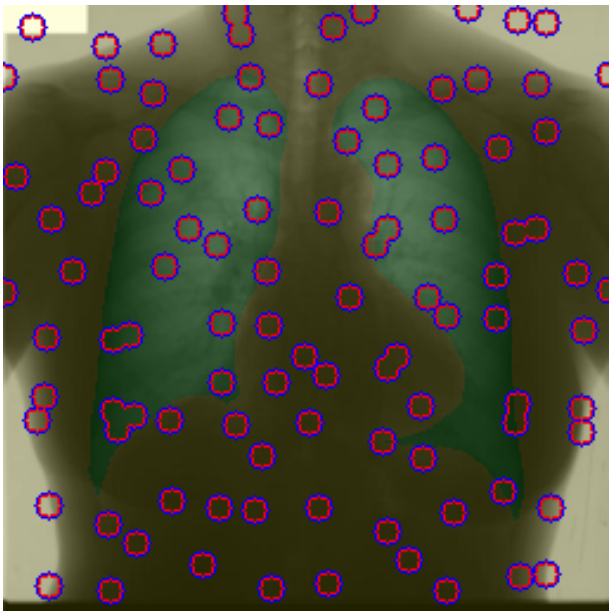


Figure 3.8: The LS speed function for the generation of the IVLD. The LS will grow in the green areas while it will shrink in the yellow one.

speed, grow (+1) or shrink (-1) are calculated. So, each example $e(p)$ in the IVLD is described as:

$$e(p) = \{t_1(p), t_2(p), \cdots, t_m(p), l(p)\}. \tag{3.9}$$

$$l(p) = \begin{cases} +1, & \text{if } p \in FG \\ -1, & \text{if } p \in BG, \end{cases} \tag{3.10}$$

where p is a point of the LS contour, $t_k(p)$ is the $k-$th term in the term set, and m is the term set size.

Since IVLD can grow indiscriminately, even in the case of small DIs, we insert a given example $e(p)$ in IVLD only if the following two conditions hold:

- p has not been inserted previously;
- $\mathbf{r} \bmod \left[k\left(b(p)+1\right)\right] < f_{\text{saveTerm}}$,

where $\mathbf{r}$ is a random variable uniformly distributed in the interval $[0, \text{RAND\_MAX}]$, k is a constant, f_{saveTerm} is a proper threshold, and $b(p)$ is a bias function,

dependent of p. The $b(p)$ function aims at rising the probability of the point p to be included in the IVLD if p is close to the object contour. It is defined as $b(p) = d(p, C)$, where C is the set of the points belonging to the target object contour and $d(\cdot)$ is the Euclidean distance function. Moreover, the values of $b(p)$ are normalized in the interval $[0, 255]$ to ensure that its values are not dependent of the image size. Finally, to take into account which points were already inserted in the IVLD, we employ the array $S(p)$. This array can have three values:

$$S(p) = \begin{cases} \text{gray}, & \text{if } p \notin \text{IVLD} \\ \text{white}, & \text{if } p \in \text{IVLD and } l(p) = +1 \\ \text{black}, & \text{if } p \in \text{IVLD and } l(p) = -1. \end{cases}$$

Fig. 3.9 shows the data structures needed to perform the construction of the IVLD.

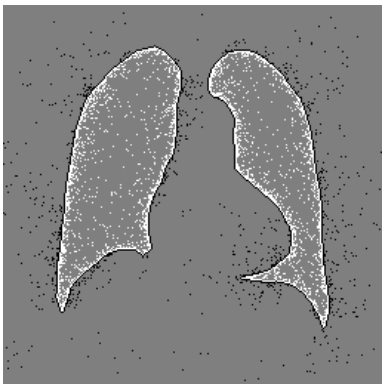
In order to maximize the number of possible examples on the IVLD, we can perform the evolution procedure several times, starting from different initialisations. We perform up to e_{small}, e_{large}, and $e_{\text{localiser}}$, evolution procedures starting from, respectively, small ellipses, large ellipses, and localiser output.

As the foreground and background segments in images often differ in size, in most cases the instances of one class significantly outnumber the ones of the other class. So, due to the kind of ML task tackled, this becomes an imbalanced classification problem [39, 179]. Hence, the last step in the IVLD creation is the balancement of the number of instances for each class. This is a common practice in ML and it has been proven that class balancement improves the accuracy of the classifiers [38, 12, 163]. To do so, we apply a simple undersampling procedure [74, 163] by randomly removing $N_{\text{maj}} - N_{\text{min}}$ instances of the majority class from the IVLD, where N_{maj} and N_{min} are the number of instances of the majority and minority classes, respectively.

A key aspect of the implementation of our system is the use of a classifier ensemble. In fact, instead of employing just one classifier to learn from the IVLD, we use N_{parts} different classifiers, where N_{parts} is the number of parts

(a) The target object contour

(b) $b(p)$



(c) $S(p)$

Figure 3.9: The data structures to perform the construction of the IVLD: (a) the target object contour, (b) $b(p)$, i.e. the normalized distance to the contour, (c) $S(p)$, i.e. the points already introduced in IVLD.

in the localiser model (see Section 3.5.1). Since each part is located in roughly the same area in each image, the rationale of using multiple drivers is reducing the learning space of each driver, therefore increasing the overall accuracy of the system. To associate each example $e(p)$ to the appropriate driver, we calculate the distance of p to each localiser part center. $e(p)$ will be then inserted to the IVLD associated to the closest driver. Algorithms 2 and 3 show the overall IVLD creation procedure. As showed in 2, the whole GT guided LS evolution procedure is repeated multiple times for the small and big ellipses initialisation. Since these initialisations employ a random component, the initialisations are different each time. Performing

these procedures multiple times allows to increase the diversity in the IVLD, leading to better generalisation in the optimal case.

```
   Data: The DI dataset
   Result: The IVLD dataset
 1 foreach I ∈ DI do
 2 |   for i = 0; i < n_se; ++i do
 3 |   |   initSmallEllipses(I);
 4 |   |   evolveIVLD();
 5 |   end
 6 |   for i = 0; i < n_be; ++i do
 7 |   |   initBigEllipses(I);
 8 |   |   evolveIVLD();
 9 |   end
10 |   initLocaliser(I);
11 |   evolveIVLD();
12 |   for i = 0; i < N_parts; ++i do
13 |   |   balanceIVLD(i);
14 |   end
15 end
```

Algorithm 2: Overall IVLD creation procedure.

Once the N_{parts} IVLDs have been defined, each RF classifier is trained with the associated IVLD. The output are N_{parts} different classification models. These models are used in the prediction phase to guide the LS when segmenting unseen images.

In the prediction phase, the LS is initialized using the output points of the localiser, as described earlier. For each point of the LS contour, the values of the terms are calculated. These values form the vector to be predicted by the associate driver, giving as output the action to be performed. The usual Shi LS stopping conditions hold.

```
   Data: The DI dataset
   Result: The IVLD dataset
 1 repeat
 2     foreach point p in level set contour do
 3         r = rand();
 4         if r mod [k(b(p) + 1)] < f_saveTerm AND p ∉ IVLD then
 5             e = calculateTermValues(p);
 6             d = getAssociatedDriver(p);
 7             add e to IVLD(d);
 8         end
 9     end
10     moveLeveSetContour();
11 until level set converges;
```

Algorithm 3: evolveIVLD(): IVLD for a single image, given a single initialisation.

3.6 Experiments

To test the performance of our proposal, we applied the developed algorithm to the previously introduced two medical image data sets in Section 3.3. In both cases the evaluation metric employed is the standard Jaccard index (see Ex. 3.8).

Since we compare against different sets of competitors, we deal with the experimental design and analysis of the results separately, in the following two sections.

3.6.1 SCR database: lungs segmentation in chest radiographs

In SCR database for each image the lung fields, heart and clavicles have been manually segmented to provide a standard reference. Here, we only test and compare on the lungs part.

3.6.1.1 Experimental design

As in [63], the 247 cases in the SCR database were split in two folds. One fold contained all 124 odd numbered images in the SCR database. The other fold contained the 123 even numbered images. Images in one fold were segmented with the images in the other fold as training set, and *vice versa*.

The parameters employed by our algorithm on this data set are shown in Table 3.1.

We compare our proposal against the segmentation results of the 12 algorithms detailed in [63]. In that work, a comprehensive explanation of the competitors is provided, as well as the numerous parameters involved. An overall description of each of the 12 algorithms is given as follows.

Human observer Two observers segmented the objects in each image: a medical student and a computer science student specialized in medical image analysis. The segmentations of the first observer were taken as gold

Parameter	Value
f_{saveTerm}	5
e_{small}	5
e_{large}	5
$e_{\text{localiser}}$	1
terms	NPdistance, Haralick, splitHaralick, globalHaralick, HOG, LBP, Gabor
r_{big}	15
r_{small}	5
examples per class	200000
Classifier ensemble size	400

Table 3.1: The parameters used by our system on the SCR data set.

standard, while the ones produced by the second observer were used to compare computerized methods with human performance.

Mean shape It serves as a reference method, since it just calculates the mean shape of each object in a way independent of the actual image contents.

Active Shape Model The ASM scheme consists of three elements: a global shape model; a local, multi-resolution appearance model; and a multi-resolution search algorithm. The implementation the authors employed is introduced in [41].

In [41] suitable values are suggested for all the parameters in the ASM scheme. They are listed in [63] and they have been used in the current experiments, referred to as *ASM default*. After performing pilot experiments varying the parameters range, the authors in [63] kept the overall best setting, referred as *ASM tuned*.

Active Appearance Models The AAM segmentation and image interpretation method [42] is an evolution of the ASM. AAM uses the same input as ASM, a set of training images in which a list of corresponding points has been indicated. The major difference to ASM is that an AAM considers all objects pixels, compared to the border representation from ASM, in a

combined model of shape and appearance. The search algorithm is also different. The tested implementation was based on the freely available C++ AAM implementation described in [162]. This algorithm is referred as *ASM default*.

AAM with whiskers To include information about the contour edges into the texture model, this AAM is characterized by the presence of contour normals pointing outwards on each object, denoted whiskers. It is referred as *ASM whiskers*.

Refinement of AAM Search Results This algorithm, referred as *ASM whiskers BFGS*, endows AAMs with a global search to avoid local minima of the cost function. It was optimised by a quasi-Newton method using the BFGS (Broyden, Fletcher, Goldfarb and Shanno) update of the Hessian [59].

Pixel classification Pixel classification (PC) is an established technique for IS. In PC, a training and a test stage can be distinguished. The training stage consists of choosing a working resolution, selecting a number of samples (positions) in each training image, and computing a set of features (the input) for each sample. An output is associated with each sample. This output lists to which class this position belongs. The last step is training a classifier with the input feature vectors and the output.

The test stage consists of computing the features for each pixel in the images (in the test set), the optional application of the transformation to each feature vector, and, finally, running the trained classifier on the feature vector to obtain the posterior probabilities that the pixel belongs to each class.

Furthermore, a *post processing* step is applied to reduce the grainy appearance at object boundaries. Subsequently the largest connected object is selected, and holes in this object are filled. This algorithm is referred as *PC post-processed*.

Hybrid approaches Since different segmentation methods providing complementary information were tested in [63], the authors also developed three hybrid segmentation schemes.

The first one, referred as *hybrid voting scheme*, considers the output of different methods. Since all methods can output hard classification labels for each pixel, the hybrid voting scheme takes the classification labels of the best performing ASM, AAM and PC scheme and assigns pixels to objects according to majority voting [94, 100].

A different approach is to take the output of one method as input for another scheme. An obvious approach is to use the posterior probabilities for each pixel as obtained from the PC method and convert these into an image where different objects have different gray value. This "probability image" is used as input for the ASM segmentation method (this is referred to as the *hybrid ASM/PC method*) and the AAM segmentation method (the *hybrid AAM/PC method*).

3.6.1.2 Results and discussion

A summary of the numeric results obtained by our proposal, along with the other 12 algorithms, is shown in Table 3.2. A selection of the resulting segmentations is shown in Fig. 3.10 while Fig. 3.11 shows the distributions of the results in boxplot form.

In general, the left lung is more difficult to segment than the right lung because of the presence of the stomach below the diaphragm, which may contain air, and the heart border, which can be difficult to discern [63].

First of all, ASM and AAM produced satisfactory results in most cases, with the tuned and whiskers variants outperforming their default counterparts. However, occasionally the segmentation proved to be problematic, especially for the left lung. In fact, in those cases where the lung and the heart are close and their borders are fuzzy, ASM and AAM followed different edges, providing unsatisfactory results. In particular, the AAM default algorithm provided a very low minimum value of its results distribution, as shown in the one before the last row of Table 3.2.

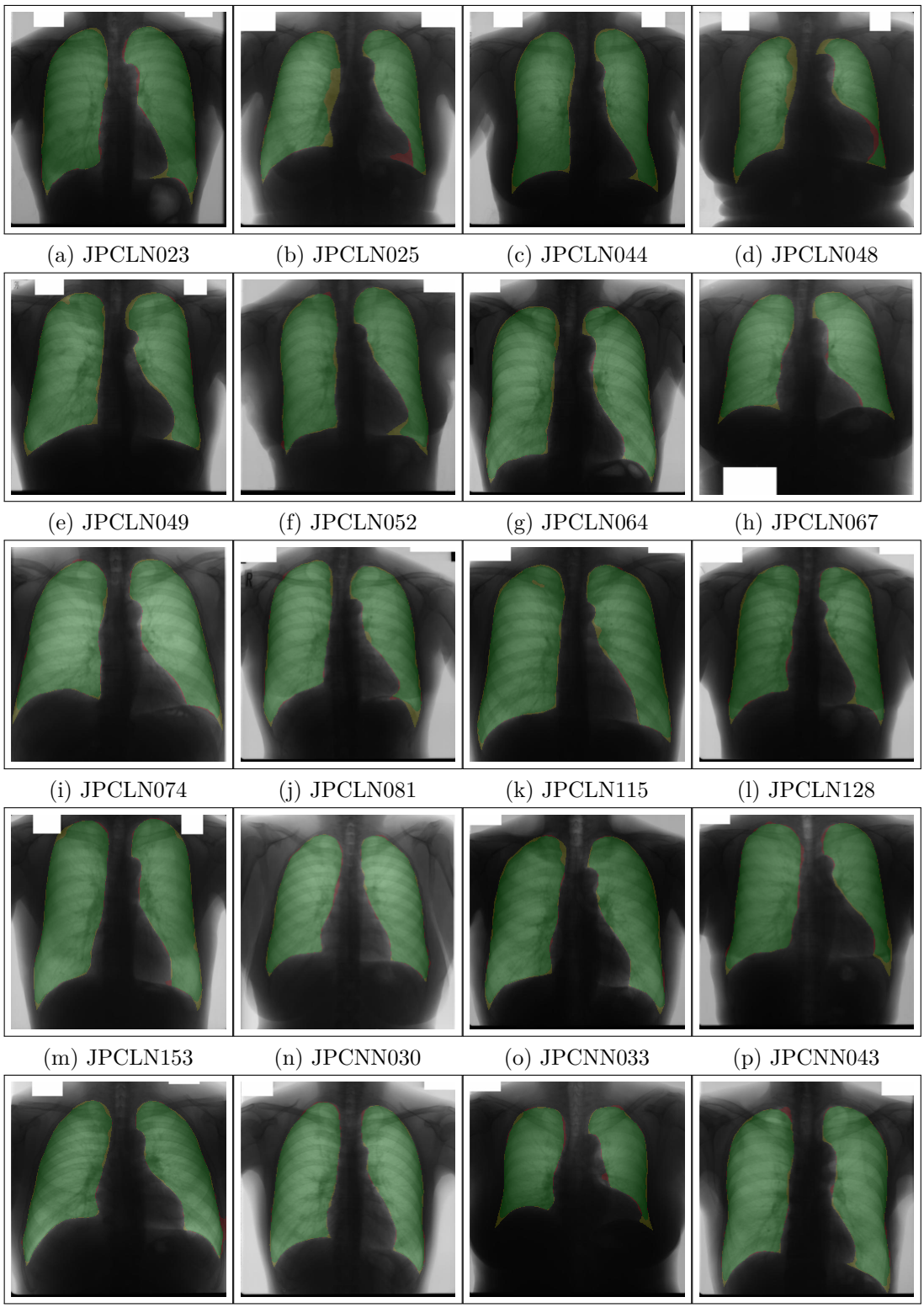
On the other hand, the PC algorithm approximated well the lung shapes and produced very satisfactory results, close to the human observer or even better, in the case of the post-processed PC. This is probably due to the key importance of texture analysis in the considered SCR data set. The hybrid systems that use the PC output for ASM and AAM outperformed the direct usage of ASM and AAM. This can be explained by the fact that PC works very well for lung segmentation and thus provides reliable input to ASM and AAM [63]. Moreover, the hybrid voting algorithm obtained the best results on this data set by combining the best performing ASM, AAM and PC schemes.

As clearly expected, the mean shape algorithm obtained the worst results while the human observer occasionally deviated from the gold standard, probably due to interpretation errors. Indeed, the human observer ranked fourth, according to the shown results (see Table 3.2).

Finally, the results obtained by our system are encouraging. It outperformed every non-hybrid proposal, ranking second according to the accuracy measure and being one of the three methods outperforming the human observer. Only the *Hybrid voting* method provided slightly better results. Most of the images were successfully segmented by our proposal, with just a few lower quality cases. In these cases, the results were affected by oversegmentation, that is, the DM leaked the object borders and ended up overlapping part of the background. This is probably mostly due to the "liquid" nature of the LS DM. Given the general scope of our proposal, we used almost the same components for both SCR and Allen Brain Atlas problems (see next subsection), avoiding the implementation of specific components. In particular, we are not employing a "hard" shape prior, strongly enforcing resemblance between the DM current shape and the prior, which could even improve the accuracy for the specific case of the SCR data set.

3.6.2 Allen Brain Atlas

As previously mentioned (see Section 3.3.2) the ABA data set contains a genome-scale collection of histological images (cellular resolution gene-



(a) JPCLN023 (b) JPCLN025 (c) JPCLN044 (d) JPCLN048

(e) JPCLN049 (f) JPCLN052 (g) JPCLN064 (h) JPCLN067

(i) JPCLN074 (j) JPCLN081 (k) JPCLN115 (l) JPCLN128

(m) JPCLN153 (n) JPCNN030 (o) JPCNN033 (p) JPCNN043

(q) JPCNN047 (r) JPCNN064 (s) JPCNN080 (t) JPCNN091

Figure 3.10: Some results of our proposal on the SCR lungs data set. Green is TP, red is FP, yellow is FN, and transparent is TN.

Lungs	$\mu \pm \sigma$	Min	Q1	Median	Q3	Max
Hybrid voting	0.949 ± 0.020	0.818	0.945	0.953	0.961	0.978
Our proposal	0.947 ± 0.023	0.803	0.943	0.952	0.961	0.974
PC post-processed	0.945 ± 0.022	0.823	0.939	0.951	0.958	0.972
Human observer	0.946 ± 0.018	0.822	0.939	0.949	0.958	0.972
PC	0.938 ± 0.027	0.823	0.931	0.946	0.955	0.968
Hybrid ASM/PC	0.934 ± 0.037	0.706	0.931	0.945	0.952	0.968
Hybrid AAM/PC	0.933 ± 0.026	0.762	0.926	0.939	0.95	0.966
ASM tuned	0.927 ± 0.032	0.745	0.917	0.936	0.946	0.964
AAM whiskers BFGS	0.922 ± 0.029	0.718	0.914	0.931	0.94	0.961
ASM default	0.903 ± 0.057	0.601	0.887	0.924	0.937	0.96
AAM whiskers	0.913 ± 0.032	0.754	0.902	0.921	0.935	0.958
AAM default	0.847 ± 0.095	0.017	0.812	0.874	0.906	0.956
Mean shape	0.713 ± 0.075	0.46	0.664	0.713	0.768	0.891

Table 3.2: The results achieved on the SCR data set by the 13 segmentation algorithms (Jaccard index).

expression profiles) obtained by In Situ Hybridisation of serial sections of mouse brains.

In this experiment we employed only section of the images in the ABA data set, namely only the ROI covering the hippocampus. The resolution of the ROIs is about 2,500 × 2,000 pixels, which we rescaled to 600 × 400 pixels.

3.6.2.1 Experimental Design

We compare the results of our proposal with the ones obtained in [121]. In that research, the authors included both deterministic and non-deterministic methods in the comparison, as well as classic and very recent proposals. In that paper, the authors considered a 22 images subset of the ABA data set, composed of 10 training images and 12 test images. They evaluated the algorithms on the whole 22 images data set. To compare with them, we provide results over the whole data set. However, we also compare the algorithms over the test partition only[2]. The parameters employed by our algorithm on this data set are shown in Table 3.3. Notice that they are

[2]We asked the authors of [121] for the results on each image and we recalculated the statistics separately for the two subsets.

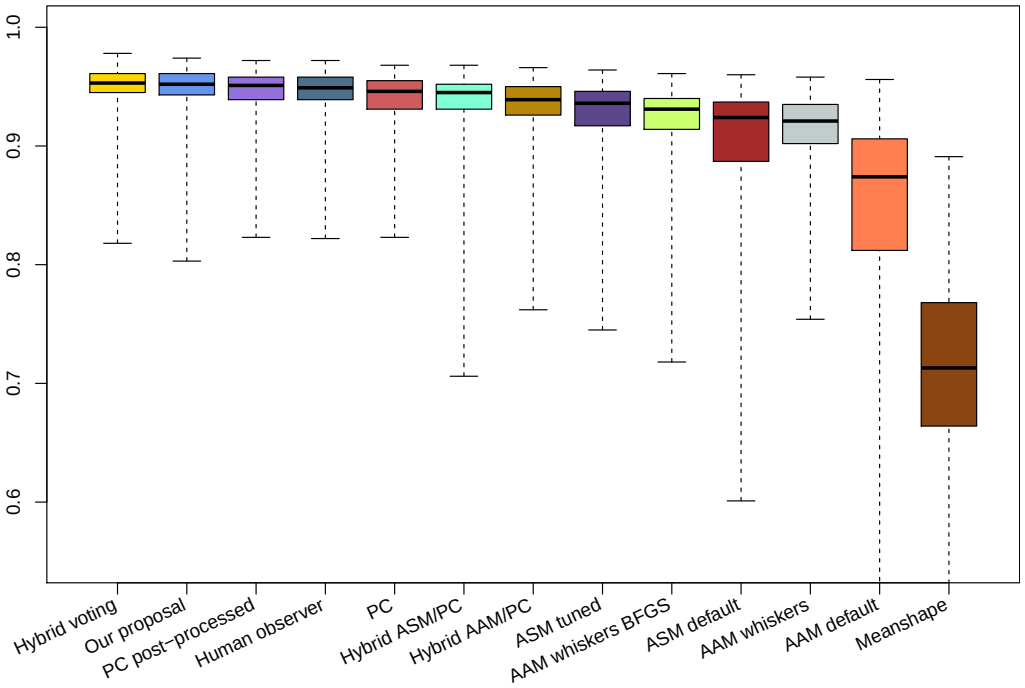


Figure 3.11: Results obtained on the SCR data set.

exactly the same ones considered for the previous SCR database (see Table 3.1 but for the f_{saveTerm} case.

Parameter	**Value**
f_{saveTerm}	4
e_{small}	5
e_{large}	5
$e_{\text{localiser}}$	1
terms	NPdistance, haralick, splitHaralick, globalHaralick, HOG, LBP, Gabor
r_{big}	15
r_{small}	5
examples per class	200000
Classifier ensemble size	400

Table 3.3: The parameters used by our system on the ABA data set.

An overall description of each of the 8 competitor algorithms is given as follows.

HybridLS This technique [121] is a hybrid LS approach for medical IS. This geometric DM combines region- and edge-based information with the prior shape knowledge introduced using deformable image registration. The proposal consists of two phases: training and test. The former implies the learning of the LS parameters by means of a Genetic Algorithm (GA) (see Section 1.5). The latter is the proper segmentation, where another evolutionary algorithm, in this case Scatter Search (SS) (see Section 1.5.2), derives the shape prior. Finally, HybridLS has the ability to learn optimal parameter settings for every specific data set. Provided a training set of already segmented images of the same class, the parameters are learned using a classic ML approach: configurations of parameters are tested on the training data and the results are compared with the GT to assess their quality.

Active Shape Models ASMs (and Iterative Otsu Thresholding Method) refined using RF (ASM + RF) [123, 122]. This method uses a medial-based shape representation in polar coordinates with the objective of creating simple models that can be managed in an easy and fast manner. Such a parametric model is moved and deformed by an EA, DE [51], according to an intensity-based similarity function between the model and the object itself. Finally, RF is applied to expand the segmented area to the regions that were not properly localized.

Soft Thresholding Soft Thresholding (ST) [2] is a recent deterministic method based on relating each pixel in the image to the different regions via a membership function, rather than through hard decisions. Such a membership function is derived from the image histogram.

Atlas-based deformable segmentation This method [169], called DS, refers to the atlas-based segmentation procedure used in HybridLS to compute the prior. This is actually a stand-alone segmentation method, therefore it is included in the experimental study as a representative of registration-based segmentation algorithms.

Geodesic Active Contours This method [34], referred as GAC, employs the Shi LS (Section 1.3) and the speed function of Geodesic Active Contour (GAC) (Section 1.3.0.1). In this work, two implementations of GAC have been tested. The first one uses as initial contour the whole image while the second one, called DSGAC, employs the segmentation obtained using DS to create the initial contour of the geometric DM.

Chan&Vese Level Set Model This technique [36], referred as Chan-Vese (CV), employs the Shi LS (Section 1.3) and the CV speed function (Section 1.3.0.1). As in GAC, two implementations have been tested. The first one uses the whole image as initial contour while the second employs the segmentation result obtained by DS as the LS initial contour.

3.6.2.2 Results and discussion

A summary of the numeric results obtained by our proposal, along with the other 8 algorithms, is shown in Tables 3.4 and 3.5, for the whole data set case and for the test set only, respectively. The resulting segmentations are shown in Fig. 3.12 (test set) and Fig. 3.13 (training set). On the other hand, Fig. 3.14 shows the distributions of the results in boxplot form for the whole data set while Fig. 3.15 depicts the same information but for the test set only.

ABA	$\boldsymbol{\mu \pm \sigma}$	**Min**	**Q1**	**Median**	**Q3**	**Max**
Our Proposal	0.845 ± 0.140	0.398	0.819	0.884	0.940	0.960
Hybrid	0.806 ± 0.109	0.321	0.796	0.849	0.869	0.884
ASM+RF	0.797 ± 0.061	0.461	0.770	0.812	0.836	0.876
DS	0.787 ± 0.108	0.342	0.760	0.829	0.852	0.877
DSGAC	0.674 ± 0.172	0.147	0.589	0.709	0.808	0.873
ST	0.597 ± 0.192	0.185	0.501	0.632	0.729	0.825
DSCV	0.538 ± 0.203	0.058	0.332	0.618	0.695	0.804
CV	0.460 ± 0.242	0.088	0.213	0.567	0.648	0.827
GAC	0.528 ± 0.192	0.146	0.395	0.564	0.660	0.833

Table 3.4: The results achieved by the 9 segmentation algorithms on the whole ABA data set.

ABA	$\mu \pm \sigma$	Min	Q1	Median	Q3	Max
Hybrid	0.800 ± 0.106	0.478	0.774	0.850	0.866	0.881
ASM+RF	0.790 ± 0.056	0.599	0.754	0.793	0.836	0.876
DS	0.774 ± 0.118	0.427	0.744	0.828	0.847	0.873
Our Proposal	0.764 ± 0.146	0.398	0.728	0.823	0.839	0.899
DSGAC	0.659 ± 0.130	0.351	0.590	0.676	0.748	0.856
ST	0.583 ± 0.225	0.185	0.440	0.647	0.751	0.825
DSCV	0.548 ± 0.193	0.058	0.531	0.628	0.671	0.712
CV	0.516 ± 0.187	0.154	0.404	0.606	0.644	0.677
GAC	0.521 ± 0.176	0.195	0.426	0.538	0.641	0.807

Table 3.5: The results achieved by the 9 segmentation algorithms on ABA test set only.

The DS method has been one of the best-performing algorithms, ranking fourth or third over the two partitions. This is probably due to the high relevance of the shape information in this data set. In fact, both DSCV and DSGAC, two algorithms relying on registration-based initialisation, ranked better than their standard counterparts.

Overall, DSGAC delivered an acceptable performance, ranking fifth in both partitions. This is remarkable as the regular GAC ranked last in both partitions, a fact that can be explained by the high sensitivity of this algorithm to its initialisation.

DSCV ranked seventh in both partitions, performing significantly worse than DSGAC. This is plausibly due to the nature of the energy function, which is probably too global for the data set at hand, even when initialized close to the target object contour. However, it outperformed the plain CV method, which achieved a bad performance, ranking second to last in both partitions.

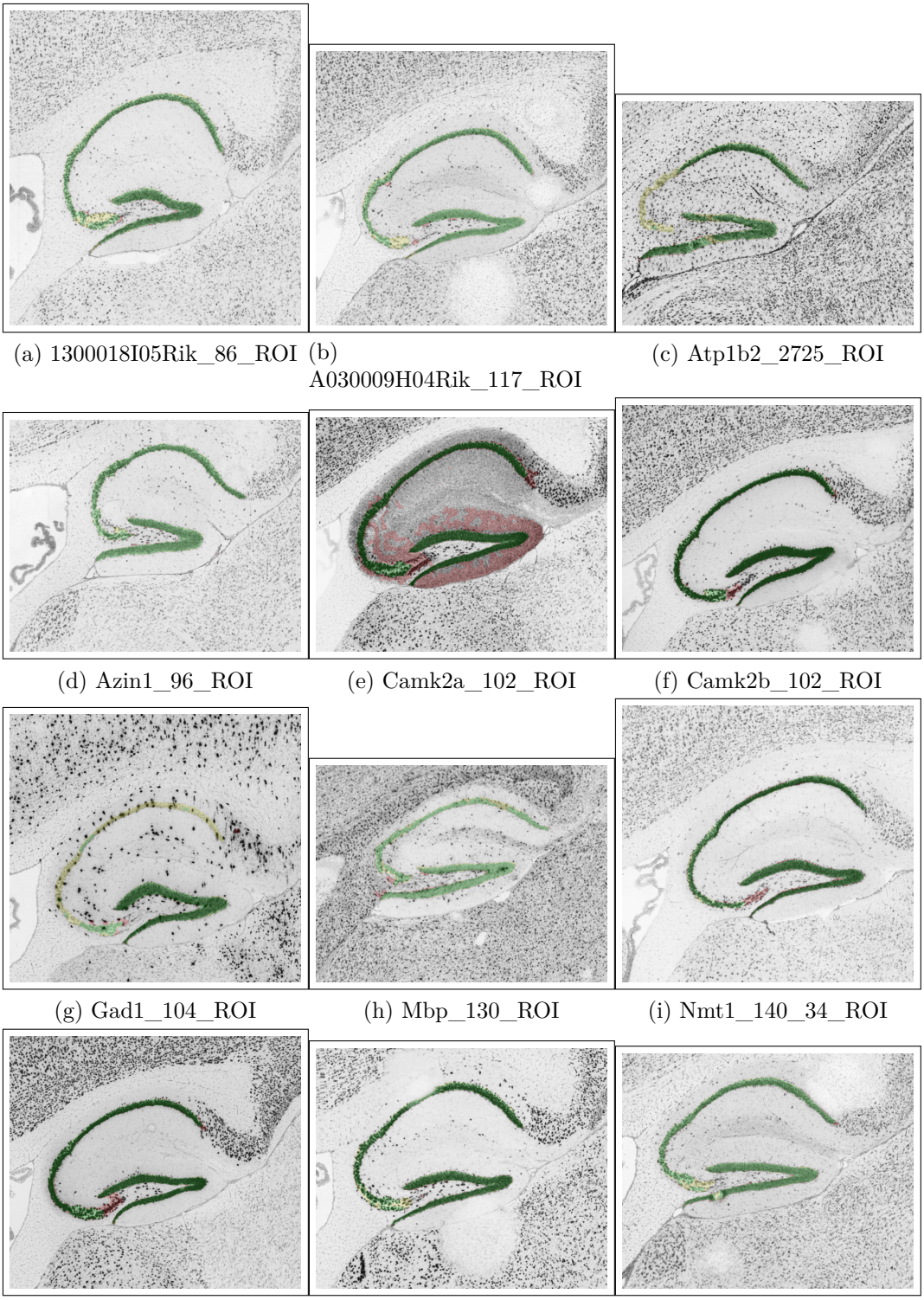
ST ranked sixth in both partitions, consistently between DSGAN and DSCV performance. This is a remarkable result as it outperformed a method employing registration-based initialisation (DSCV) and two extended LSs, CV and GAC. However, being ST based on the image histogram only, it showed limited ability to cope with complex scenarios.

ASM+RF obtained some of the best results on this data set, ranking third or second on the two partitions. However, it is fair to underline its

ad-hoc nature, since it was developed specifically for the ABA data set. Moreover, it is not able to manage topological changes in a natural way as geometric DMs can do.

HybridLS has obtained the best results on the test-only partition of this data set and it ranked second when measuring its performance on the whole data set. This is presumably due to the combination of a well performing prior (i.e. the DS method) and the simultaneous use of visual image cues along with a flexible LS implementation. However, its drawbacks are its long execution times [121] and its complexity, as the method relies on an elaborate registration-based initialisation and a separate optimisation phase to perform parameter learning.

Finally, the results obtained by our proposal are encouraging, as its accuracy is close to the best methods in the comparison. In particular, it ranked fourth on the test partition (with very close performance to the best three algorithms) and first on the whole data set. In fact, most of the images were successfully segmented by our proposal. Some of them, however, present under or over-segmentations. While the former are probably due to the faint borders in some areas of the target structures or very similar textures across different structures, the latter are presumably caused by the somewhat arbitrary (at least to a non expert eye) borders of the lower part of the upper structure of the Hippocampus. In particular, Fig. 3.12(e) shows the worst result. In this case, the DM leaks and expands until the gray area around the target structure. The reason behind this behavior is plausibly attributable to the absence of similar cases in the small training set considered.



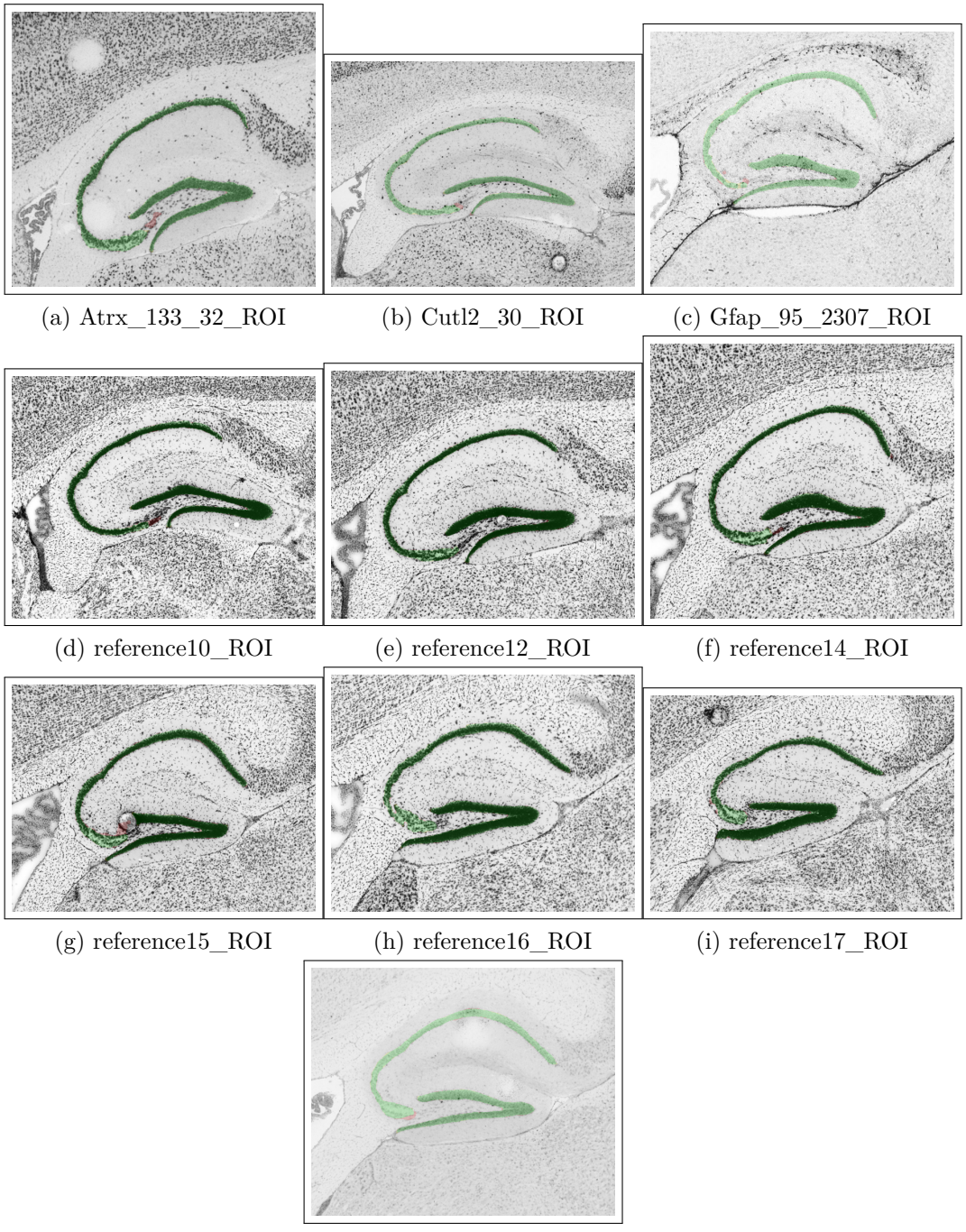
(a) 1300018I05Rik_86_ROI (b) A030009H04Rik_117_ROI (c) Atp1b2_2725_ROI

(d) Azin1_96_ROI (e) Camk2a_102_ROI (f) Camk2b_102_ROI

(g) Gad1_104_ROI (h) Mbp_130_ROI (i) Nmt1_140_34_ROI

(j) Slc17a7_120_ROI (k) Tubb3_114_ROI (l) Wars_109_ROI

Figure 3.12: Results of our proposal on the ABA data set (test set). Green is TP, red is FP, yellow is FN, and transparent is TN.



(a) Atrx_133_32_ROI (b) Cutl2_30_ROI (c) Gfap_95_2307_ROI

(d) reference10_ROI (e) reference12_ROI (f) reference14_ROI

(g) reference15_ROI (h) reference16_ROI (i) reference17_ROI

(j) Zim1_117_ROI

Figure 3.13: Results of our proposal on the ABA data set (training set). Green is TP, red is FP, yellow is FN, and transparent is TN.

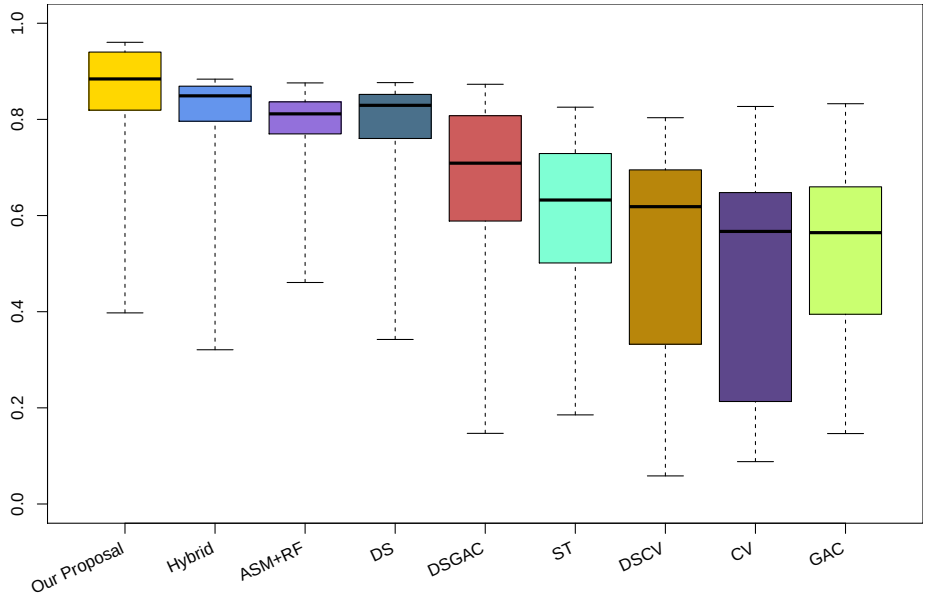


Figure 3.14: Results obtained on the ABA data set on the whole ABA data set.

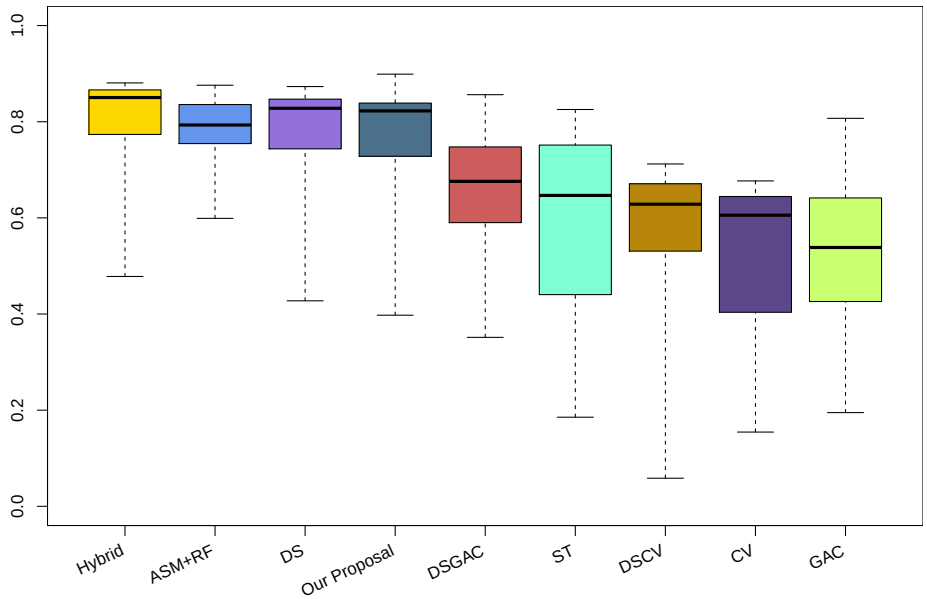


Figure 3.15: Results obtained on the ABA data set on ABA test set only.

CHAPTER 4

Conclusions and future work

***Spike**: "So that's the story. And what was the real lesson? Don't leave things in the fridge."*

— Cowboy Bebop, *Toys in the Attic*

4.1 Conclusions

We proposed a general framework for image modality classification. The flexible framework proposed in this thesis works with a task dependent interconnection between the components. It consists of a *Feature Extractor* that works with a set of user defined algorithms to create feature vectors from the images. The *Feature Encoder* uses a user defined method to fuse the required feature vectors. The purpose of this is to aggregate the feature vectors to form a low dimensionality feature space where discrimination is more feasible. For this supervised and unsupervised Machine Learning (ML) methods are used. The *Classifier* component—as the name suggests—does the actual classification. A fourth component is responsible for optimising the parameters of the other components, to reach an optimal solution performing *Hyper-Parameter Optimisation* or model selection. Interconnection of the components depends on the selected features, low dimensional global

features can be directly fed to the Classifier, while other feature vectors are aggregated with the Feature Encoder.

We proposed an accurate, flexible, automatic, and general purpose system for Image Segmentation (IS) employing ML techniques to perform direct guidance of the Deformable Model (DM) contour. We provided a reference implementation of the proposed system using Level Sets (LSs) as DM, an ensemble classifier as ML method, the previously introduced part-based organ detector as localiser, and a proper set of image visual descriptors. Finally, we tested it over two medical imaging data sets, achieving encouraging results. The introduced method was competitive with other state-of-the-art medical IS algorithms while requiring minimal human intervention.

- *Propose a ML-based generic framework for IS using DMs.* We proposed a generic, automatic IS framework able to learn the segmentation model from example data. Our framework is composed of four main components: (i) the localiser, to detect the image ROI (ii) the DM (iii) the set of terms, that is the image features we take into account (iv) and the driver, a general purpose ML-based method to perform classification or regression in order to automatically adapt the DM. The main feature of the system is that the driver directly guides the DM contour adjustment to the object, instead of relying on an optimisation procedure. A fifth component, the integration mechanism, takes into account how the other four components connect to each other. The presented framework shows a high degree of flexibility and can be tailored to different IS problems.

- *Develop an ML-based IS framework implementation for medical imaging.* To prove the feasibility of the proposed ML-based IS framework, we provided a specific implementation for medical IS. We chose the organ detector introduced in this thesis. Then, we decided to employ a fast LS implementation, the Shi approximation, as DM. With the aim of capturing the variability of the image characteristics, we grouped a large set of image- and model-related features to define the term set. Besides, we chose Random Forest (RF), a fast, accurate

and flexible decision tree classifier ensemble as driver. Finally, the integration mechanism was designed to fully take advantage of the characteristics of the four components while combining them.

The work carried out for the last three objectives of this PhD dissertation will result in a paper to be submitted to a medical imaging journal in the short future.

- *Analyze the performance of the proposed methods.*

4.2 Future works

- *Use of different DMs for the implementation of the ML-based IS framework.* As pointed out in Section 3.4, our framework is customizable and independent of the choice of the specific DM to be considered for the segmentation task.

- *Consideration of a different driver for the implementation of the ML-based IS framework.* RF, the ML method of choice in the provided reference implementation of the ML-based IS framework, showed an interesting performance. However, given the current state of development of the ML field, a vast amount of alternatives can be considered. Among them, Rotation Forest [149] has some attractive characteristics. In fact, it employs an embedded Principal Component Analysis [88] to perform automatic feature selection. This is a desirable property when dealing with problems with a large amount of features as the one at hand. A different alternative could be one of the numerous flavors of the vastly employed Support Vector Machine (SVM) classifier [27, 37, 44, 47]. In this case, we would opt for a fast implementation, to keep the processing time low. Finally, depending on the nature of the DM employed in the framework, the use of a regressor (instead of a classifier) could be a more appropriate choice, as mentioned in the previous item. In that case, the use of ML methods of a different nature would be required.

- *Incorporation of new features for the implementation of the ML-based IS framework*. To further improve the accuracy of the current implementation, the results provided by advanced edge detectors could be introduced in the term set. With this purpose in mind, the use of the attractive edge detection algorithm described in [119] could be a good choice

- *Benchmarking of a different localizer for the implementation of the ML-based IS framework*. Apart from testing different part-based localizers, registration-based localizers could also be successfully introduced in the current implementation. In particular, the SS-based image registration algorithm proposed in [170] represents an attractive option as it has been employed to perform IS in [124] with a significant accuracy.

- *Testing the ML-based IS framework on different medical imaging problems*. The presented ML-based IS framework implementation can also be applied to the large number of different medical IS problems found in literature. Among them, active and upcoming MICCAI grand challenges [1] represent one of the best choices for testing on open problems in medical IS.

[1] `www.grand-challenge.org`

Bibliography

[1] M. A. Aizerman, E. M. Braverman, and L. I. Rozoner. "Theoretical foundations of the potential function method in pattern recognition learning". In: *Automation and Remote Control* 25 (1964), pp. 821–837.

[2] S. Aja-Fernandez, G. Vegas-Sanchez-Ferrero, and M.A. Martin Fernandez. "Soft thresholding for medical image segmentation". In: *Proceedings of the International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2010, pp. 4752–4755.

[3] Allen Institute for Brain Science. *Allen Reference Atlases*. `http://mouse.brain-map.org`. 2004–2006.

[4] E. Alpaydin. *Introduction to Machine Learning*. 2nd. The MIT Press, 2010. ISBN: 026201243X, 9780262012430.

[5] Shun-ichi Amari and Si Wu. "Improving support vector machine classifiers by modifying kernel functions". In: *Neural Networks* 12.6 (1999), pp. 783–789.

[6] L. Ascari, C. Morandi, P. Carrai, and G. Adorni. "Studio di un sistema di percezione della qualità per immagini riscalate contenti trame". MA thesis. Università degli Studi di Parma - Facoltà di Ingegneria, 1999.

[7] T. Back, D. B. Fogel, and Z. Michalewicz, eds. *Handbook of Evolutionary Computation*. 1st. Bristol, UK, UK: IOP Publishing Ltd., 1997. ISBN: 0750303921.

[8] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*. Vol. 463. ACM press New York, 1999.

[9] Lu Bai and EdwinR. Hancock. "Graph Kernels from the Jensen-Shannon Divergence". English. In: *Journal of Mathematical Imaging and Vision* 47.1-2 (2013), pp. 60–69. ISSN: 0924-9907. DOI: 10.1007/s10851-012-0383-6.

[10] Prasanna Balaprakash, Mauro Birattari, and Thomas Stützle. "Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement". In: *Hybrid Metaheuristics*. Springer Berlin Heidelberg, 2007, pp. 108–122.

[11] J. W. Bartlett, L. A. van de Pol, C. T. Loy, R. I. Scahill, C. Frost, P. Thompson, and N. C. Fox. "A meta-analysis of hippocampal atrophy rates in Alzheimer's disease". In: *Neurobiology of aging*. Elsevier, 2009, pp. 1711–1723.

[12] G. E. Batista, R. C. Prati, and M. C. Monard. "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 20–29. ISSN: 1931-0145. DOI: 10.1145/1007730.1007735.

[13] N K Batmanghelich, B Taskar, and C Davatzikos. "Generative-Discriminative Basis Learning for Medical Imaging". In: *Medical Imaging, IEEE Transactions on* 31.1 (2012), pp. 51–69.

[14] Pierre-Yves Baudin, Danny Goodman, Puneet Kumar, Noura Azzabou, Pierre G Carlier, Nikos Paragios, and M Pawan Kumar. "Discriminative Parameter Estimation for Random Walks Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013* 8151.Chapter 28 (2013), pp. 219–226.

[15] S. Belongie, J. Malik, and J. Puzicha. "Matching Shapes". In: *Proceedings of the International Conference on Computer Vision*. Vol. 1. 2001, pp. 454–461.

[16] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization”. In: *The Journal of Machine Learning Research* 13.1 (Mar. 2012), pp. 281–305.

[17] Mauro Birattari, Thomas Stützle, Luis Paquete, Klaus Varrentrapp, et al. “A Racing Algorithm for Configuring Metaheuristics.” In: *GECCO*. Vol. 2. Citeseer. 2002, pp. 11–18.

[18] C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[19] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

[20] M B Blaschko, A Vedaldi, and A Zisserman. “Simultaneous Object Detection and Ranking with Weak Supervision.” In: *NIPS* (2010).

[21] David M. Blei and Michael I. Jordan. “Variational inference for Dirichlet process mixtures”. In: *Bayesian Analysis* 1 (2005), pp. 121–144.

[22] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. “Latent dirichlet allocation”. In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 993–1022. ISSN: 1532-4435. DOI: `http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993`.

[23] P. P. Bonissone. “Soft computing: the convergence of emerging reasoning technologies”. In: *Soft Computing* 1.1 (1997), pp. 6–18.

[24] P. P. Bonissone, Y. Chen, K. Goebel, and P. Khedkar. “Hybrid Soft Computing Systems: Industrial and Commercial Applications”. In: *Proceedings of the IEEE, Special Issue on Computational Intelligence*. Vol. 87. 9. 1999, pp. 1641–1667.

[25] Anna Bosch, Andrew Zisserman, and Xavier Munoz. “Image classification using random forests and ferns”. In: (2007).

[26] Anna Bosch, Andrew Zisserman, and Xavier Muñoz. “Scene classification via pLSA”. In: *Computer Vision–ECCV 2006*. Springer, 2006, pp. 517–530.

[27] Bernhard E Boser, Isabelle Guyon, and Vladimir Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *Computational Learing Theory*. 1992, pp. 144–152.

[28] Sabri Boughorbel, Jean-Philippe Tarel, and Nozha Boujemaa. "Generalized histogram intersection kernel for image recognition". In: *Image Processing, 2005. ICIP 2005. IEEE International Conference on*. Vol. 3. IEEE. 2005, pp. III–161.

[29] A.L. Boulesteix, S. Janitza, J. Kruppa, and I. R. König. "Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics". In: *WIREs Data Mining Knowl Discov* 2.6 (2012), pp. 493–507. DOI: 10.1002/widm.1072.

[30] L. Breiman. "Random Forests". In: *Maching Learning* 45 (1 2001), pp. 5–32.

[31] Christopher J C Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". In: *Data Min. Knowl. Discov.* 2.2 (June 1998), pp. 121–167.

[32] J Canny. "A Computational Approach to Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8.6 (June 1986), pp. 679–698. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767851.

[33] V. Caselles. "Geometric models for active contours". In: *Proceedings of the 1995 International Conference on Image Processing (Vol. 3)-Volume 3 - Volume 3*. ICIP '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 3009–. ISBN: 0-8186-7310-9.

[34] V. Caselles, R. Kimmel, and G. Sapiro. "Geodesic Active Contours". In: *International Journal of Computer Vision* 22.1 (1997), pp. 61–79.

[35] D Chai and K N Ngan. "Face segmentation using skin-color map in videophone applications". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 9.4 (1999), pp. 551–564.

[36] T. Chan and L. Vese. "Active Contours without Edges". In: *IEEE Transactions on Image Processing* 10 (2 2001), pp. 266–277.

[37] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*. 2001.

[38] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.

[39] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. "Editorial: special issue on learning from imbalanced data sets". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 1–6. ISSN: 1931-0145. DOI: 10.1145/1007730.1007733.

[40] O Chum, J Philbin, J Sivic, and M Isard. "Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval". In: *2007 IEEE 11th International Conference on Computer Vision*. Rio de Janeiro, Brazil: IEEE, Oct. 2007, pp. 1–8. ISBN: 978-1-4244-1630-1. DOI: 10.1109/ICCV.2007.4408891.

[41] T. F. Cootes and C. J. Taylor. "Statistical models of appearance for medical image analysis and computer vision". In: (2001), pp. 236–248. DOI: 10.1117/12.431093.

[42] TF Cootes, GJ Edwards, and CJ Taylor. "Active appearance models". In: *IEEE Trans. on Pattern Analysis And Machine Intelligence* 23.6 (2001), 681–685.

[43] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. 1^{st}. Vol. 19. Advances in Fuzzy Systems–Applications and Theory. World Scientific Publishing Co. Pte. Ltd., 2001.

[44] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (1995), pp. 273–297.

[45] A. Criminisi, J. Shotton, D. Robertson, and E. Konukoglu. "Regression forests for efficient anatomy detection and localization in CT studies". In: *MCV'10: Proceedings of the 2010 international MICCAI conference on Medical computer vision: recognition techniques and applications in medical imaging*. Springer-Verlag, 2010.

[46] A Criminisi et al. "Anatomy Detection and Localization in 3D Medical Images". In: *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013, pp. 193–209.

[47] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[48] G. Csurka, S Clinchant, and Guillaume Jacquet. "XRCE's Participation at Medical Image Modality Classification and Ad-hoc Retrieval Tasks of ImageCLEF 2011". In: *CLEF 2011 working notes*. Amsterdam, The Netherlands, 2011.

[49] Marco Cuturi, Kenji Fukumizu, and Jean-Philippe Vert. "Semigroup Kernels on Measures". In: *The Journal of Machine Learning Research* 6 (Dec. 2005).

[50] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection". In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. CVPR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 886–893. ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.177.

[51] S. Das and P.N. Suganthan. "Differential Evolution: A Survey of the State-of-the-Art". In: *IEEE Trans. on Evolutionary Computation* 15 (2011), pp. 4–31.

[52] R. Datta, D. Joshi, J. Li, and J. Z. Wang. "Image retrieval: Ideas, influences, and trends of the new age". In: *ACM Computing Surveys* 40.2 (2008), 5:1–5:60. ISSN: 0360-0300. DOI: 10.1145/1348246.1348248.

[53] J. G. Daugman. "Two-dimensional spectral analysis of cortical receptive field profiles". In: *Vision Research* 20.10 (1980), pp. 847–856. DOI: 10.1016/0042-6989(80)90065-6.

[54] J. G. Daugman. “Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters”. In: *Journal of the Optical Society of America A: Optics, Image Science, and Vision* 2.7 (1985), pp. 1160–1169.

[55] RO Duda. “Use of the Hough transformation to detect lines and curves in pictures”. In: *Communications of the ACM* (1972).

[56] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.

[57] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. “Object detection with discriminatively trained part-based models”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.9 (2010), pp. 1627–1645.

[58] M. Fenchel, S. Thesen, and A. Schilling. “Automatic labeling of anatomical structures in MR FastView images using a statistical atlas”. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008*. Springer, 2008, pp. 576–584.

[59] R. Fletcher. *Practical methods of optimization; (2nd ed.)* New York, NY, USA: Wiley-Interscience, 1987. ISBN: 0-471-91547-5.

[60] W. T. Freeman, K. Tanaka, J. Ohta, and K. Kyuma. “Computer vision for computer games”. In: *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG ’96)*. FG ’96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 100–. ISBN: 0-8186-7713-9.

[61] William T. Freeman and Michal Roth. “Orientation Histograms for Hand Gesture Recognition”. In: *In International Workshop on Automatic Face and Gesture Recognition*. 1994, pp. 296–301.

[62] Murray Gell-Mann and Constantino Tsallis. “Nonextensive entropy-Interdisciplinary applications”. In: *Nonextensive Entropy-Interdisciplinary Applications, by Edited by Murray Gell-Mann and C Tsallis, pp. 440. Oxford University Press, Apr 2004. ISBN-10: 0195159764. ISBN-13: 9780195159769* 1 (2004).

[63] B. van Ginneken, M.B. Stegmann, and M. Loog. "Segmentation of anatomical structures in chest radiographs using supervised methods: a comparative study on a public database". In: *Medical Image Analysis* 10.1 (2006), pp. 19–40.

[64] F. Glover. "Heuristics for integer programming using surrogate constraints". In: *Decision Sciences* 8.1 (1977), pp. 156–166.

[65] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN: 0201157675.

[66] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. 2nd. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001. ISBN: 0201180758.

[67] A.B.A. Graf, A.J. Smola, and S. Borer. "Classification in a normalized feature space using support vector machines". In: *Neural Networks, IEEE Transactions on* 14.3 (2003), pp. 597–605.

[68] Kristen Grauman and Trevor Darrell. "The pyramid match kernel: Discriminative classification with sets of image features". In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. Vol. 2. IEEE. 2005, pp. 1458–1465.

[69] Mryka Hall-Beyer. *The GLCM Tutorial Home Page*. `http://www.fp.ucalgary.ca/mhallbey/tutorial.htm`.

[70] R. M. Haralick. "Statistical and structural approaches to texture". In: *Proceedings of the IEEE* 67.5 (1979), pp. 786–804. ISSN: 0018-9219. DOI: `10.1109/PROC.1979.11328`.

[71] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. "Textural Features for Image Classification". In: *IEEE Transactions on Systems, Man, and Cybernetics* 3.6 (1973), pp. 610–621. ISSN: 0018-9472. DOI: `10.1109/TSMC.1973.4309314`.

[72] Jan Havrda and František Charvát. "Quantification method of classification processes. Concept of structural a-entropy". In: *Kybernetika* 3.1 (1967), pp. 30–35.

[73] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[74] Haibo He and Edwardo A. Garcia. "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284.

[75] Ralf Herbrich and Thore Graepel. "A PAC-Bayesian margin bound for linear classifiers". In: *Information Theory, IEEE Transactions on* 48.12 (2002), pp. 3140–3150.

[76] William R Hersh, Henning Müller, Jeffery R Jensen, Jianji Yang, Paul N Gorman, and Patrick Ruch. "Advancing Biomedical Image Retrieval: Development and Analysis of a Test Collection". In: *Journal of the American Medical Informatics Association* 13.5 (2006), pp. 488–496. DOI: 10.1197/jamia.M2082. eprint: http://jamia.bmj.com/content/13/5/488.full.pdf.

[77] J. Holland. "Adaptation in natural and artificial systems". In: *Artificial Intelligence* (1989).

[78] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and S Sundararajan. "A dual coordinate descent method for large-scale linear SVM". In: *ICML '08: Proceedings of the 25th international conference on Machine learning*. ACM, July 2008.

[79] Frank Hutter. "Automated configuration of algorithms for solving hard computational problems". PhD thesis. University of British Columbia, 2009.

[80] F. Idris and S. Panchanathan. "Review of Image and Video Indexing Techniques". In: *Journal of Visual Communication and Image Representation* 8.2 (1997), pp. 146–166. DOI: 10.1006/jvci.1997.0355.

[81] Paul Jaccard. "The Distribution of the Flora in the Alpine Zone". In: *New Phytologist* 11.2 (1912), pp. 37–50. ISSN: 0028646X. DOI: 10.2307/2427226.

[82] A Jain. "Image retrieval using color and shape". In: *Pattern Recognition* 29.8 (Aug. 1996), pp. 1233–1244.

[83] J[yh-Shing] R. Jang, C[huen-Tsai] Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Prentice-Hall, 1997.

[84] Tony Jebara, Risi Kondor, and Andrew Howard. "Probability Product Kernels". In: *The Journal of Machine Learning Research* 5 (Dec. 2004), pp. 819–844.

[85] H. Jegou, H. Harzallah, and C. Schmid. "A contextual dissimilarity measure for accurate and efficient image search". In: *Computer Vision and Pattern Recognition, 2007, IEEE Conference on, (CVPR '07)*. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.382970.

[86] Thorsten Joachims. "Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06". In: *the 12th ACM SIGKDD international conference*. New York, New York, USA: ACM Press, 2006, pp. 217–226.

[87] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. "Cutting-plane training of structural SVMs". In: *Machine Learning* 77.1 (Oct. 2009).

[88] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[89] Florian Jung, Matthias Kirschner, and Stefan Wesarg. "A Generic Approach to Organ Detection Using 3D Haar-Like Features". In: *Bildverarbeitung für die Medizin 2013*. Springer, 2013, pp. 320–325.

[90] Dagmar Kainmüller, Thomas Lange, and Hans Lamecker. "Shape constrained automatic segmentation of the liver based on a heuristic intensity model". In: *Proceedings of the MICCAI Workshop 3D Segmentation in the Clinic: A Grand Challenge*. 2007, pp. 109–116.

[91] Jayashree Kalpathy-Cramer, Henning Müller, Steven Bedrick, Ivan Eggel, Alba Garcia Seco de Herrera, and Theodora Tsikrika. "The CLEF 2011 medical image retrieval and classification tasks". In: *CLEF 2011 working notes*. Amsterdam, The Netherlands, 2011.

[92] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi. "Conformal curvature flows: From phase transitions to active vision". In: *Archive for Rational Mechanics and Analysis* 134 (3 1996). 10.1007/BF00379537, pp. 275–301. ISSN: 0003-9527.

[93] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. "Optimization by simmulated annealing". In: *science* 220.4598 (1983), pp. 671–680.

[94] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. "On combining classifiers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.3 (1998), pp. 226–238.

[95] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN: 0262013193, 9780262013192.

[96] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004. ISBN: 0471210781.

[97] G Kundu, V Srikumar, and D Roth. "Margin-based Decomposed Amortized Inference". In: *Urbana* (2013).

[98] M. Laguna and R. Marti. *Scatter Search: Methodology and Implementations in C*. Norwell, MA, USA: Kluwer Academic Publishers, 2002. ISBN: 1402073763.

[99] Abolfazl Lakdashti and M. Moin. "A New Content-Based Image Retrieval Approach Based on Pattern Orientation Histogram". In: *Computer Vision/Computer Graphics Collaboration Techniques*. Ed. by André Gagalowicz and Wilfried Philips. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2007, pp. 587–595.

[100] L. Lam and C.Y. Suen. "Application of majority voting to pattern recognition: An analysis of its behavior and performance". In: *IEEE Transactions on Systems, Man, and Cybernetics* 27 (1997), pp. 553–568.

[101] PW Lamberti, AP Majtey, A Borras, Montserrat Casas, and A Plastino. "Metric character of the quantum Jensen-Shannon divergence". In: *Physical Review A* 77.5 (2008), p. 052311.

[102] Shawn Lankton, Student Member, and Allen Tannenbaum. "A.: Localizing region-based active contours". In: *IEEE Transactions on Image Processing* (2008), pp. 2029–2039.

[103] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2006, pp. 2169–2178.

[104] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. "Efficient sparse coding algorithms". In: *Advances in neural information processing systems*. 2006, pp. 801–808.

[105] Tai Sing Lee. "Image Representation Using 2D Gabor Wavelets". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18 (1996), pp. 959–971.

[106] Thomas Leung and Jitendra Malik. "Representing and Recognizing the Visual Appearance of Materials Using Three-dimensional Textons". In: *Int. J. Comput. Vision* 43.1 (June 2001), pp. 29–44. ISSN: 0920-5691. DOI: 10.1023/A:1011126920638.

[107] Jing Li and Nigel M. Allinson. "A comprehensive review of current local features for computer vision". In: *Neurocomputing* 71.10-12 (2008), pp. 1771–1787. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2007.11.032.

[108] Shen Li, João V Graça, and Ben Taskar. "Wiki-ly supervised part-of-speech tagging". In: *EMNLP-CoNLL '12: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, July 2012.

[109] Haibin Ling, S Kevin Zhou, Yefeng Zheng, Bogdan Georgescu, Michael Suehling, and Dorin Comaniciu. "Hierarchical, learning-based automatic liver segmentation". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008, pp. 1–8.

[110] M López-Ibánez, J Dubois-Lacoste, and T Stützle. "The irace Package: Iterated Racing for Automatic Algorithm Configuration". In: (2011).

[111] David G Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[112] David G. Lowe. "Object Recognition from Local Scale-Invariant Features". In: *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–. ISBN: 0-7695-0164-8.

[113] S Maji, A C Berg, and J Malik. "Classification using intersection kernel support vector machines is efficient". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[114] AP Majtey, A Borrás, Montserrat Casas, PW Lamberti, and A Plastino. "JENSEN–SHANNON DIVERGENCE AS A MEASURE OF THE DEGREE OF ENTANGLEMENT". In: *International Journal of Quantum Information* 6.supp01 (2008), pp. 715–720.

[115] AP Majtey, PW Lamberti, and DP Prato. "Jensen-Shannon divergence as a measure of distinguishability between mixed quantum states". In: *Physical Review A* 72.5 (2005), p. 052310.

[116] R. Malladi, J. Sethian, and B. Vemuri. "Shape Modeling with Front Propagation: A Level Set Approach". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 17 (2 Feb. 1995), pp. 158–175. ISSN: 0162-8828. DOI: 10.1109/34.368173.

[117] S. Marčelja. "Mathematical description of the responses of simple cortical cells∗". In: *Journal of the Optical Society of America* 70.11 (1980), pp. 1297–1300. DOI: 10.1364/JOSA.70.001297.

[118] Oded Maron and Andrew W Moore. "The racing algorithm: Model selection for lazy learners". In: *Lazy learning*. Springer, 1997, pp. 193–225.

[119] David R Martin, Charless C Fowlkes, and Jitendra Malik. "Learning to detect natural image boundaries using local brightness, color, and texture cues". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26.5 (2004), pp. 530–549.

[120] André F T Martins, Noah A Smith, Eric P Xing, Pedro M Q Aguiar, and Mário A T Figueiredo. "Nonextensive Information Theoretic Kernels on Measures". In: *The Journal of Machine Learning Research* 10 (Dec. 2009), pp. 935–975.

[121] P. Mesejo, A. Valsecchi, L. Marrakchi-Kacem, S. Cagnoni, and S. Damas. "Biomedical Image Segmentation using Geometric Deformable Models and Metaheuristics". In: *Computerized Medical Imaging and Graphics* (2013).

[122] Pablo Mesejo, Roberto Ugolotti, Stefano Cagnoni, Ferdinando Di Cunto, and Mario Giacobini. "Automatic Segmentation of Hippocampus in Histological Images of Mouse Brains using Deformable Models and Random Forest". In: *Proc. of Symposium on Computer-Based Medical Systems, CBMS '12*. 2012.

[123] Pablo Mesejo, Roberto Ugolotti, Ferdinando Di Cunto, Mario Giacobini, and Stefano Cagnoni. "Automatic Hippocampus Localization in Histological Images using Differential Evolution-Based Deformable Models". In: *Pattern Recognition Letters* 34.3 (2013), pp. 299–307.

[124] Pablo Mesejo, Andrea Valsecchi, Linda Marrakchi-Kacem, Stefano Cagnoni, and Sergio Damas. "Biomedical image segmentation using

geometric deformable models and metaheuristics". In: *Computerized Medical Imaging and Graphics* (2014).

[125] Thomas Minka and Zoubin Ghahramani. "Expectation Propagation for Infinite Mixtures". In: *Advances in Neural Information Processing Systems (NIPS): Workshop on Nonparametric Bayesian Methods and Infinite Models*. 2003.

[126] Thomas M. Mitchell. *Machine Learning*. 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN: 0070428077, 9780070428072.

[127] Jean-Michel Morel and Guoshen Yu. "ASIFT: A New Framework for Fully Affine Invariant Image Comparison". In: *SIAM Journal on Imaging Sciences* 2.2 (Apr. 2009).

[128] Jean-michel Morel and Guoshen Yu. *On the consistency of the SIFT method*. Tech. rep. 2008.

[129] Henning Müller, Nicolas Michoux, David Bandon, and Antoine Geissbuhler. "A Review of Content-Based Image Retrieval Systems in Medical Applications - Clinical Benefits and Future Directions". In: *International Journal of Medical Informatics* 73.1 (2003), pp. 1–23.

[130] David Mumford and Jayant Shah. "Optimal approximations by piecewise smooth functions and associated variational problems". In: *Communications on Pure and Applied Mathematics* 42.5 (1989), pp. 577–685.

[131] Alexander Nareyek. "Choosing search heuristics by non-stationary reinforcement learning". In: *Metaheuristics: Computer decision-making*. Springer, 2004, pp. 523–544.

[132] Radford M. Neal. "Markov Chain Sampling Methods for Dirichlet Process Mixture Models". In: *Journal of Computational and Graphical Statistics* 9.2 (2000), pp. 249–265.

[133] John A Nelder and Roger Mead. "A simplex method for function minimization". In: *The computer journal* 7.4 (1965), pp. 308–313.

[134] Nils J. Nilsson. "Probabilistic Logic". In: *Artificial intelligence* 28.1 (1986), pp. 71–87.

[135] D Nister and H Stewenius. "Scalable Recognition with a Vocabulary Tree". In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. 2006, pp. 2161–2168.

[136] Sebastian Nowozin and Christoph H Lampert. "Structured Learning and Prediction in Computer Vision". In: *Foundations and Trends® in Computer Graphics and Vision* 6.3–4 (Mar. 2011).

[137] Francesca Odone, Annalisa Barla, and Alessandro Verri. "Building kernels from binary strings for image matching". In: *Image Processing, IEEE Transactions on* 14.2 (2005), pp. 169–180.

[138] Timo Ojala, Matti Pietikäinen, and David Harwood. "A comparative study of texture measures with classification based on featured distributions". In: *Pattern Recognition* 29.1 (1996), pp. 51–59.

[139] Yew-Soon Ong, Meng Hiot Lim, and Xianshun Chen. "Research frontier: memetic computation-past, present & future". In: *IEEE Computational Intelligence Magazine* 5.2 (2010), pp. 24–31. ISSN: 1556-603X. DOI: 10.1109/MCI.2010.936309.

[140] Edgar Osuna, Robert Freund, and Federico Girosi. *Support Vector Machines: Training and Applications*. Tech. rep. AIM-1602. 1997.

[141] Olivier Pauly, Ben Glocker, Antonio Criminisi, Diana Mateus, Axel Martinez Möller, Stephan Nekolla, and Nassir Navab. "Fast multiple organ detection and localization in whole-body MR Dixon sequences". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011*. Springer, 2011, pp. 239–247.

[142] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[143] Witold Pedrycz. *Fuzzy control and fuzzy systems (2nd, extended ed.)* Taunton, UK, UK: Research Studies Press Ltd., 1993. ISBN: 0-86380-131-5.

[144] A Pentland, R W Picard, and S Sclaroff. "Photobook: Content-based manipulation of image databases". In: *International Journal of Computer Vision* 18.3 (1996), pp. 233–254.

[145] J Philbin, O Chum, M Isard, J Sivic, and A Zisserman. "Lost in quantization: Improving particular object retrieval in large scale image databases". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[146] L Pishchulin, A Jain, M Andriluka, T Thormahlen, and B Schiele. "Articulated people detection and pose estimation: Reshaping the future". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE Computer Society, 2012, pp. 3178–3185.

[147] J Platt. *Sequential minimal optimization: A fast algorithm for training support vector machines*. Tech. rep. 1998.

[148] Michael JD Powell. "A direct search optimization method that models the objective and constraint functions by linear interpolation". In: *Advances in optimization and numerical analysis*. Springer, 1994, pp. 51–67.

[149] J.J. Rodriguez, L.I. Kuncheva, and C.J. Alonso. "Rotation Forest: A New Classifier Ensemble Method". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.10 (2006), pp. 1619–1630. ISSN: 0162-8828. DOI: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2006.211.

[150] Carsten Rother. "Regression Tree Fields — An efficient, non-parametric approach to image labeling problems". In: *CVPR '12: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR*. IEEE Computer Society, June 2012.

[151] Heike Ruppertshofen, Cristian Lorenz, Sarah Strunk, Peter Beyerlein, Zein Salah, Georg Rose, and Hauke Schramm. "Discriminative Generalized Hough Transform for Localization of Lower Limbs". In: *Journal of Computer Science–Research & Development* (2010).

[152] Gerard Salton and Christopher Buckley. "Term-weighting approaches in automatic text retrieval". In: *Information processing & management* 24.5 (1988), pp. 513–523.

[153] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM". In: *ICML '07: Proceedings of the 24th international conference on Machine learning*. ACM, June 2007.

[154] Y. Shi and W. C. Karl. "A fast level set method without solving PDEs". In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. 2005, pp. 97–100.

[155] Yonggang Shi and W. C. Karl. "A Real-Time Algorithm for the Approximation of Level-Set-Based Curve Evolution". In: *Trans. Img. Proc.* 17.5 (May 2008), pp. 645–656. ISSN: 1057-7149. DOI: 10.1109/TIP.2008.920737.

[156] J Shiraishi et al. "Development of a digital image database for chest radiographs with and without a lung nodule: receiver operating characteristic analysis of radiologists' detection of pulmonary nodules." In: *AJR. American journal of roentgenology* 174.1 (2000), pp. 71–74.

[157] K. Siddiqi, Y. B. Lauzière, A. Tannenbaum, and S. W. Zucker. "Area and Length Minimizing Flows for Shape Segmentation". In: *IEEE Computer Society Conference on Pattern Recognition and Image Processing*. 1997, pp. 621–627.

[158] Josef Sivic and Andrew Zisserman. "Video Google: A Text Retrieval Approach to Object Matching in Videos". In: *9th IEEE International Conference on Computer Vision (ICCV 2003)*. IEEE Computer Society, 2003, pp. 1470–1477. ISBN: 0-7695-1950-4.

[159] Alex J Smola and Bernhard Schölkopf. *Learning with kernels*. Citeseer, 1998.

[160] Cees G M Snoek, Marcel Worring, and Arnold W M Smeulders. "Early versus late fusion in semantic video analysis". In: *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*. New York, New York, USA: ACM Request Permissions, Nov. 2005, pp. 399–402.

[161] Sören Sonnenburg et al. “The SHOGUN machine learning toolbox”. In: *The Journal of Machine Learning Research* 11 (2010), pp. 1799–1802.

[162] Mikkel B. Stegmann, Bjarne K. Ersbøll, and Rasmus Larsen. “FAME - A Flexible Appearance Modelling Environment.” In: *IEEE Transactions on Medical Imaging* 22.10 (2003), pp. 1319–1331.

[163] Y. Sun, A.K.C. Wong, and M. Kemel. “Classification of imbalanced data: A review.” In: *International Journal of Pattern Recognition and Artificial Intelligence* 23.4 (2009), pp. 687–719.

[164] Michael J Swain and Dana H Ballard. “Color indexing”. In: *International journal of computer vision* 7.1 (1991), pp. 11–32.

[165] Ben Taskar, Daphne Koller, and Carlos Guestrin. “Max-margin Markov networks”. In: *NIPS*. Ed. by Youngmin Cho Lawrence Saul and B Schölkopf. 2003, pp. 25–32.

[166] Ken Thompson. “Programming Techniques: Regular Expression Search Algorithm”. In: *Commun. ACM* 11.6 (June 1968), pp. 419–422. ISSN: 0001-0782. DOI: 10.1145/363347.363387.

[167] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. “Support vector machine learning for interdependent and structured output spaces”. In: *ICML ’04: Proceedings of the twenty-first international conference on Machine learning*. ACM, July 2004.

[168] M. Tuceryan and A. K. Jain. “Texture Analysis”. In: *The Handbook of Pattern Recognition and Computer Vision (2nd Edition)* (1988).

[169] Andrea Valsecchi, Sergio Damas, José Santamaría, and Linda Marrakchi-Kacem. *Intensity-based Image Registration using Scatter Search*. Tech. rep. AFE 2012-14. Submitted to Artificial Intelligence in Medicine. 2012.

[170] Andrea Valsecchi, Sergio Damas, José Santamaría, and Linda Marrakchi-Kacem. “Intensity-based image registration using scatter search”. In: *Artificial intelligence in medicine* 60.3 (2014), pp. 151–163.

[171] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.

[172] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995. ISBN: 0-387-94559-8.

[173] A. Vedaldi and B. Fulkerson. *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. `http://www.vlfeat.org/`. 2008.

[174] Andrea Vedaldi and Andrew Zisserman. "Efficient additive kernels via explicit feature maps". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.3 (2012), pp. 480–492.

[175] Paul Viola and Michael J. Jones. "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. 2001, pp. I–511.

[176] Li Wang and Dong-Chen He. "Texture classification using texture spectrum". In: *Pattern Recognition* 23.8 (1990), pp. 905–910.

[177] Xiaoyu Wang, T X Han, and Shuicheng Yan. "An HOG-LBP human detector with partial occlusion handling". In: *Computer Vision, 2009 IEEE 12th International Conference on*. 2009, pp. 32–39.

[178] Yao Wang, Zhu Liu, and Jin-Cheng Huang. "Multimedia content analysis-using both audio and visual clues". In: *Signal Processing Magazine, IEEE* 17.6 (2000), pp. 12–36.

[179] Gary M. Weiss. "Mining with rarity: a unifying framework". In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 7–19. ISSN: 1931-0145. DOI: `10.1145/1007730.1007734`.

[180] Jianchao Yang, Kai Yu, Yihong Gong, and T Huang. "Linear spatial pyramid matching using sparse coding for image classification". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1794–1801.

[181] A. Yezzi, S. Kichenassamy, A. Kumar, P. Olver, and A. Tannenbaum. “A geometric snake model for segmentation of medical imagery”. In: *IEEE Trans. on Medical Imaging* (1997), pp. 199–209.

[182] L.A. Zadeh. “Fuzzy sets”. In: *Information and Control* 8.3 (1965), pp. 338–353.

[183] L.A. Zadeh. “Soft Computing and Fuzzy Logic”. In: 11.6 (1994), pp. 48–56.

[184] L.A. Zadeh. “What is soft computing?” In: *Soft Computing* 1.1 (1997), p. 1.

[185] Wojciech Zaremba, M Pawan Kumar, Alexandre Gramfort, and Matthew B Blaschko. “Learning from M/EEG data with variable brain activation delays”. In: *IPMI’13: Proceedings of the 23rd international conference on Information Processing in Medical Imaging*. Berlin, Heidelberg: Springer-Verlag, June 2013, pp. 414–425.

[186] Wanlei Zhao, Yu-Gang Jiang, and Chong-Wah Ngo. “Keyframe retrieval by keypoints: Can point-to-point matching help?” In: *Image and Video Retrieval*. Springer, 2006, pp. 72–81.

[187] Yefeng Zheng, Adrian Barbu, Bogdan Georgescu, Michael Scheuering, and Dorin Comaniciu. “Four-chamber heart modeling and automatic segmentation for 3-D cardiac CT volumes using marginal space learning and steerable features”. In: *Medical Imaging, IEEE Transactions on* 27.11 (2008), pp. 1668–1681.

[188] Shaohua Kevin Zhou, Jinghao Zhou, and Dorin Comaniciu. “A boosting regression approach to medical anatomy detection”. In: *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*. 2007, pp. 1–8.