University of São Paulo
Institute of Mathematics and Statistics
Bachelor degree in Computer Science

Augusto César Monteiro Silva

Image Operator Learning Based on Local Features

São Paulo
December 2017

This work is partially supported by São Paulo Research Foundation (FAPESP) grant nº 2017/09137-0

Image Operator Learning
Based on Local Features

Final monograph of the course
MAC0499 - Supervised Graduate Project.

Supervisor: Prof. Dr. Roberto Hirata Jr.
Cosupervisor: Dr. Igor dos Santos Montagner

São Paulo
December 2017

Abstract

Image operators have a great deal of applications in many different areas, ranging from document analysis to medical imaging and so forth. However, the manual design of such operators is very difficult and time consuming, while also being highly prone to human errors. Therefore, an automatic design of image operators is extremely desirable and many previous works have been done on this task. One interesting approach is the automatic design of W-operators that are locally defined within a determined region. The main disadvantage of previous methods of automatic constructing a W-operator is a limitation on the size of the predetermined region. We propose a novel approach on the stablished method of learning a W-operator from pairs of input-output images, with the goal of enabling the use of larger windows on the learning method. We do this by extracting a feature from the region, so we can summarize the important information from it while also discarding unnecessary values. We validate our assumptions by comparing operators learned by the already stablished method with the operators learned using our implemented features. The results shows that, although the stablished method (described in this monograph as the Raw feature) is a well-rounded learning technique for every dataset, specific combinations of feature and dataset can produce similar accuracy while significantly decreasing the feature vector size on the training method.

Keywords: image processing, morphological operators, feature extraction, image operator learning.

Contents

List of Abbreviations

DFT	Discrete Fourier Transform
LBP	Local Binary Pattern
DRIVE	Digital Retinal Images for Vessel Extraction
DIBCO	Document Image Binarization Competition
TRIOS	Training Image Operators from Sample
HOG	Histogram of Oriented Gradients

Chapter 1

Introduction

One of the main applications of image processing in several areas revolve around image operators that transform an image into another, such as segmentation or binarization of an object of interest. For instance, some diseases, such as diabetic retinopathy, can be detected and classified by the amount of ramification in the blood vessels of the retina of a patient. However, detecting the blood vessels from a digital image of a retina is a hard task and the result differs even amongst specialists. Therefore, the design of an image operator that performs the segmentation of the blood vessels in these images can help the diagnosis of such diseases.

Amidst these operators, a particular interesting and extensive field of research is the study on Mathematical Morphology, and more specifically, the morphological operators (Soille (2002)). Although these operators are generally used on binary images, they are not restricted to it. Famous examples of such operators are erosion and dilation of images.

Manual design of morphological operators is a hard task, and it is usually done through a trial and error approach, consuming a great amount of time and effort from the specialist. It is also completely dependent on the expertise of the specialist, which makes it prone to human errors. In order to prevent that, many researches were done on automating the task of designing an image operator from a set of pairs of images that represents an input and an output.

One of the methods of learning image operators can be seen in the pioneering work by Junior Barrera (1997), and many works were done on improving such framework. This method consists on learning an image operator by defining a local function that, given a region on the input image, returns the value of a pixel in the output image. This technique requires the definition of a window W that is used as the region of input to the local function. One of the main disadvantages of this method is that it restricts the size of W, due to the proposed process being unfeasible on a large W (Nina (2009)). Therefore, the learned local function cannot have information on large regions of the image.

In this monograph we propose a novel approach that enables the use of large regions on the process of learning the image operator, without falling on the curse of dimensionality. Our approach is to extract some useful information from the region while discarding unneeded ones. By doing so, the dimensionality of the training process can be heavily diminished. We also tried to improve the accuracy of the previous used technique with this feature extraction step.

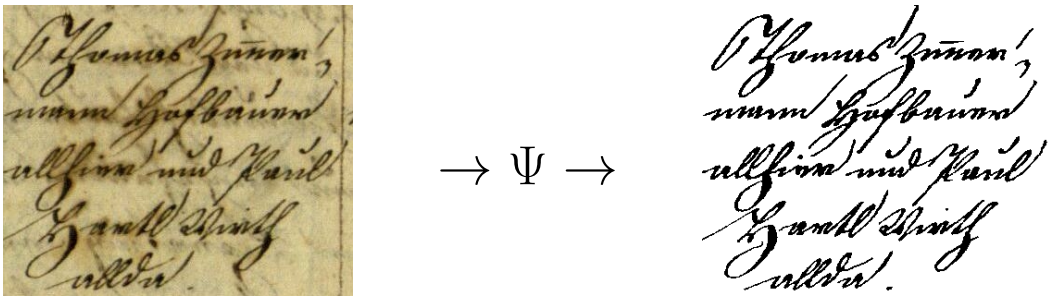
Chapter 2 describes in more details the method of learning image operators used in this project, while also introducing the necessary mathematical background and the notation used throughout this monograph. After that, Chapter 3 describe all of the 6 features implemented and tested in this project and it also provides references for each one of them, so the interested

reader can easily investigate them further. Chapter 4 depicts the experiments made with all features, describing the datasets used, the code, the language and the library used, and all metrics calculated in order to compare each one of the features in each dataset. It also contains all results of every feature in each dataset. Finally, Chapter 5 contains our final remarks on this project, and provides some thoughts on possible future works on this area.

Chapter 2

Image Operator Learning

In this project, we are concerned about the problem of learning, from pairs of input-output images, how to transform the input image into the output one. This problem is modeled as the learning of an image operator that, when applied to an input image, performs the desired task. An example of such pair of images can be seen in Figure 2.1.



(a) *Input image* **(b)** *Output image*

Figure 2.1: *Example of input-output images*

The method used in this project(Montagner *et al.* (2016b)) is based on the pioneering work by Junior Barrera (1997), where the learning of image operators is modeled as the learning of local transformations. More specifically, we are concerned about learning morphological operators that are translation invariant and locally defined within a window W. Operators that respect those two properties are called W-operators.

Automatic design of a W-operator is still a challenging problem and many researches are being done about novel methodologies that perform this task or improvement in already stablished techniques, like Montagner *et al.* (2016a) and Dornelles e Hirata (2015). The method used in this project is based on extracting information from a neighboring region of a pixel, and then, classifying that pixel with that information. We restrict ourselves to the case where the output image is binary, because those image operators can be simpler to train while still representing important transformations, such as segmentation and object detection. This method is further described in section 2.2.

A problem on this technique arises when we use large sizes of window W, as the most common information extracted from the region is a vector with the intensity values of every pixel within that neighborhood. Therefore, a large window leads to large feature vectors, and, the higher the dimensionality of the feature vector, more infeasible the training of the classifier becomes.

Many attempts to overcome this issue were proposed throughout the years. Enhancements on the selection of the window W (Dornelles e Hirata (2015)), use of different machine

learning techniques, like Convolutional Neural Networks (D. Julca-Aguilar e Hirata (2017)) and multilevel training (Nina (2009)), are some examples of such improvements.

In this work, we propose a novel approach where the information extracted from that region creates a smaller feature vector than the original one. To achieve this, we build a set of different feature extractors, each one based on a technique that is commonly used on image processing or computer vision systems, and then apply it to every pixel on the image. The classifier, then, is trained on the resulting feature vectors.

2.1 Background

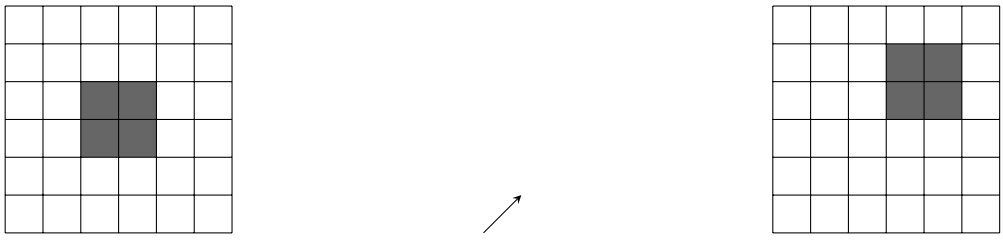
Although grayscale digital images are only defined on a finite set of points, we assume that the image function f is defined on the discrete grid $\mathbb{E} = \mathbb{Z}^2$. Therefore, a grayscale digital image with pixel depth k can be described by a function $f : \mathbb{E} \to K$, where $K = \{0, 1, ..., k-1\}$. The set of all digital images with gray-levels in K is denoted as $K^{\mathbb{E}}$, so $K^{\mathbb{E}} = \{f | f : \mathbb{E} \to K\}$.

An image operator is a function Ψ of the form $\Psi : K^{\mathbb{E}} \to K^{\mathbb{E}}$, i.e., is a function that takes an image as input and transforms it into another image. Note that the set of digital binary images is contained in the set of grayscale images. Thus, this definition allows transformations of gray level images into binary ones. Accordingly, the value of $[\Psi(f)](p)$ is the gray level intensity value of the pixel on position p of the transformed image.

In this project, we are concerned about W-operators, thus, we need to define translation invariance and local definition propertios of an operator Ψ. For more information on morphological operators, the reader can refer to Heijmans (1994).

Translation Invariance

The translation of an image f by q is denoted f_q and is defined as $f_q(p) = f(p - q)$. An example of the translation of a binary digital image by a vector q can be seen in Figure 2.2.



(a) *Image* f **(b)** *Vector* $q = (1, 1)$ **(c)** *Image* f_q

Figure 2.2: *Example of translation*

Operator Ψ is said to be translation invariant if, for any $f \in K^{\mathbb{E}}$ and $q \in \mathbb{E}$

$$[\Psi(f)]_q = \Psi(f_q). \tag{2.1}$$

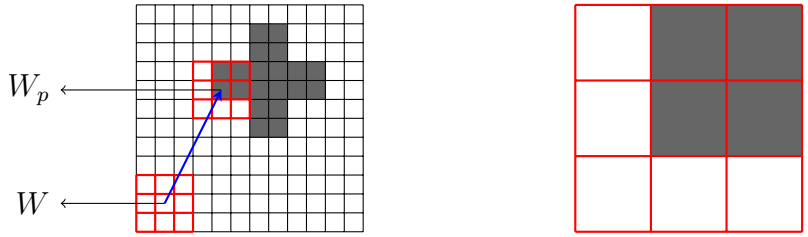
In other words, operator Ψ is translation invariant if applying Ψ on image f translated by q results in the same image as applying Ψ on image f and then translating the resulting image by q. Detection of external or internal contours are examples of operators that are translation invariant.

Local Definition

An operator Ψ is locally defined if there exists a limited region $W \subset \mathbb{E}$ such that

$$[\Psi(f)](p) = [\Psi(f|_{W_p})](p) \quad \forall p \in \mathbb{E} \tag{2.2}$$

where $f|_{W_p}$ is defined as the image where every pixel within region W_p has the same value as the correspondent pixel in f, and every pixel that is not in W_p has value 0. The region W_p is the window W translated by p.



(a) *Window W and W_p* **(b)** $f|_{W_p}$

Figure 2.3: *Local definition*

In other words, the value of each pixel in the resulting image $\Psi(f)$ only depends on the surrounding neighborhood W on image f.

W-operators

W-operators are interesting for image operator learning because any W-operator Ψ can be characterized by a local function $\psi : K^W \to K$, as shown in Heijmans (1994) Thus, for each pixel p of image f:

$$[\Psi(f)](p) = \psi(f_{-p}|_W) \tag{2.3}$$

Therefore, the problem of designing a W-operator can be reduced to the problem of estimating a local function that performs the same transformation. Even more, estimating a local function can be seen as a pattern classification problem, where the pattern is constrained to W and the classes are the possible values on the output image (0 or 1, in the case of binary output images).

2.2 Learning Method

In order to formulate the learning process, it is necessary to define some function to be optimized, like an error or a cost function. Thus, we are going to define the Mean Absolute Error (MAE) of an operator and the optimal operator for this error function. Then, we describe the training approach used to estimate this optimal operator from the pairs of images on the training set. The interested reader can refer to Junior Barrera (1997) for further information.

2.2.1 Mean Absolute Error (MAE)

We assume that an image f is a realization of a random process F. Thus, an input-output pair (f, g) is a realization of a random process pair (F, G) with some joint distribution

$P(F, G)$. Under these assumptions, the objective of the learning process is to find an operator Ψ that, when applied to F, returns a random process $\Psi(F)$ that is as close to G as possible. We can calculate this distance between $\Psi(F)$ and G by the MAE. Considering ψ as being the local function that characterizes the operator Ψ, the MAE can be calculated at any point $p \in \mathbb{E}$ by:

$$MAE(\Psi) = E[|\psi(f|_{W'_p}) - g(p)|] \tag{2.4}$$

As W-operators are translation invariant and locally defined, $f|_{W'_p}$ can be represented by a random vector X_p and $g(p)$ as a random variable y_p. Considering the stationarity of those random process the position p is irrelevant, and equation 2.4 can be written as:

$$MAE(\Psi) = E[|\psi(X) - y|] \tag{2.5}$$

To minimize the distance between $\Psi(F)$ and G we need to estimate a function ψ that minimizes the MAE in equation 2.5. Since we are concerned about binary output images, the possible values of y are 0 and 1. Then, let us expand equation 2.5 in order to determine the optimal function ψ for the binary case.

$$\begin{aligned} MAE(\psi) &= E[|\psi(X) - y|] \\ &= \sum_{(X,y)} P(X, y)(|\psi(X) - y|) \\ &= \sum_{(X,y)} P(X)P(y|X)(|\psi(X) - y|) \\ &= \sum_{X} P(X)[P(0|X)\psi(X) + P(1|X)(1 - \psi(X))] \end{aligned} \tag{2.6}$$

It is clear, from equation 2.6, that the optimal estimator ψ that minimizes the MAE is the following local function:

$$\psi(X) = \begin{cases} 1, & \text{if } P(1|X) > P(0|X) \\ 0, & \text{if } P(0|X) > P(1|X) \\ 0 \text{ or } 1, & \text{otherwise} \end{cases} \tag{2.7}$$

Therefore, if we know the joint probability distribution of the input-output processes (F, G), defining an estimator that minimizes the MAE measure is straightforward. However, given only a set of input-output images, the joint probability distribution is not known. Hence, our approach is to estimate these probability distributions by the given pairs of images.

2.2.2 Training Process

The method used to estimate this local function consists of three main steps: Feature Extraction, Observed Patterns Decision and Generalization. Each one of these steps is described considering as an example the input-output images of Figure 2.4, and a 3x3 window W centered at the pixel's position.

(a) *Input* **(b)** *Output*

Figure 2.4: *Input-output pair of images*

Feature Extraction

Let (f, g) be pair of images from the given input-output process. For every image f of the training set, slide window W centered at each pixel of the image and record a feature extracted from that window jointly with the value of the correspondent pixel in image g. In other words, considering Φ a feature extractor, i. e., a function of the form $\Phi : K^W \to \mathbb{R}^n$ where n is defined by the feature extractor, record the pair $(\Phi(f_p|_W), g(p))$ for each pixel p of image f.

In the example (figure 2.5) function Φ just returns the values of each pixel within window W.

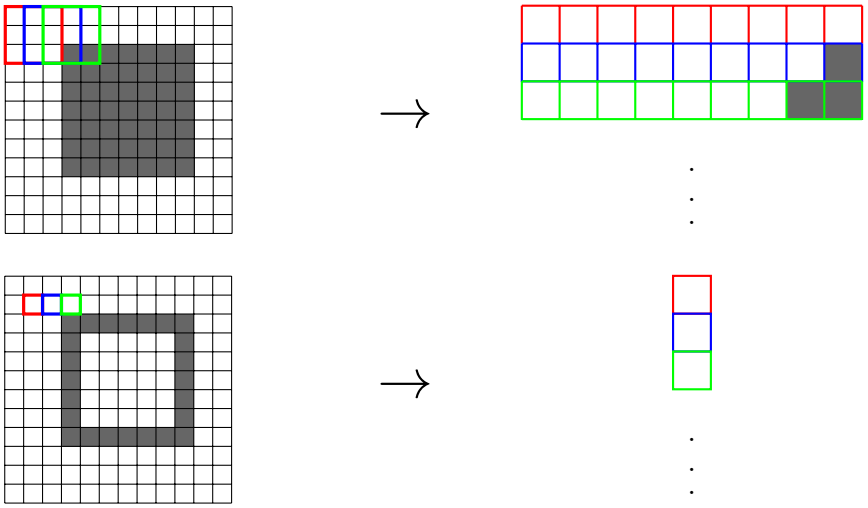


Figure 2.5: *Example of Feature Extraction step. Each color indicates a training sample* (X, y)

Observed Patterns Decision

For each recorded pattern $X = \Phi(f_p|_W)$, decide the value of $\hat{y} = \hat{\psi}(\Phi(f_p|_W))$ to be:

$$\hat{y} = \hat{\psi}(X) = \begin{cases} 1 & \text{if} \quad \hat{P}(y = 1|X) > \hat{P}(y = 0|X) \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

where $\hat{P}$ is the estimated joint distribution.

If all possible patterns were observed, the operator ψ could be fully determined by equation 2.8. Usually that is not the case, therefore, a method to generalize the estimated operator $\hat{\psi}$ is necessary.

An example of this step on the image learning process can be seen in figure 2.4.

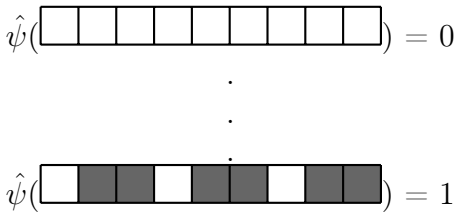


Figure 2.6: *Example of estimated* $\hat{\psi}$

Generalization

Generalizing operator $\hat{\psi}$ can be done by different techniques. Some previous works, such as Nina (2009) used algorithms for minimization of boolean functions while others, like D. Julca-Aguilar e Hirata (2017), approached this as a classification problem using machine learning techniques. The latter is also used in this work.

Learning the image operator ψ is modeled as a binary classification problem, where the inputs are the feature extracted from the first step and the expected outputs are the ones decided in the second step. Therefore, any machine learning technique for solving a classification problem can be used. In this work, we used Decision Trees.

Figure 2.7 shows the entire pipeline of the image operator learning process.

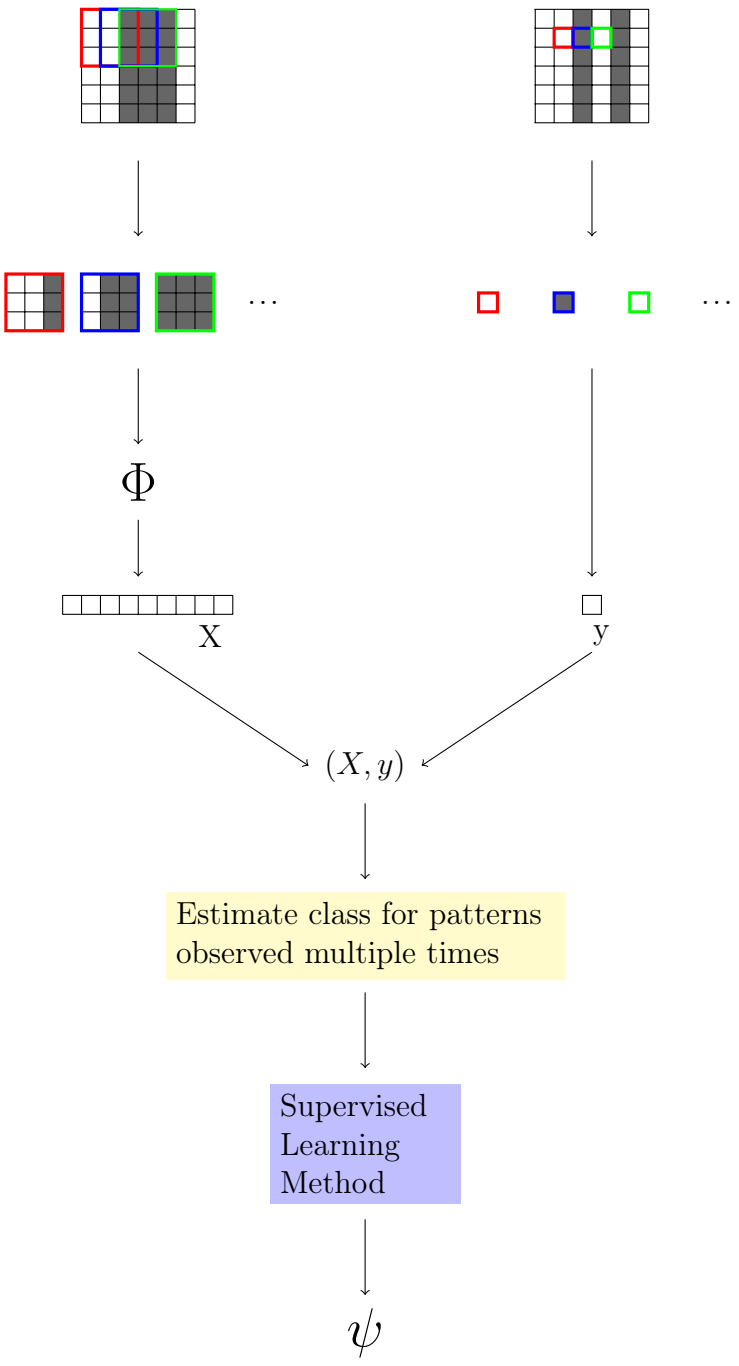


Figure 2.7: *Pipeline of learning the image operator*

Chapter 3

Local Features

As stated in Chapter 2, feature extraction is an important step in machine learning algorithms. Extracting the most important information from the input can impact significantly the accuracy of the classifier, can reduce the dimensionality of the feature vector and it can provide a better understanding of the underlying process that generated the data.

Feature extraction and feature selection in machine learning are important areas of research and many techniques to find the most relevant ones from an input were proposed throughout the years. Consider the following example that shows how different features can impact machine learning algorithms.

Suppose that we have a dataset that can be classified in two different categories and, for each sampled data, a pair of real values (x, y) can be extracted from it. Suppose also that plotting this pair (x, y) for each sample data generates the graph on Figure 3.1. In this dataset, no linear classifier can separate both categories in an efficient manner.

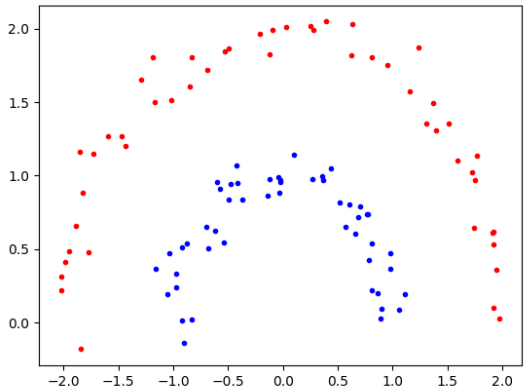


Figure 3.1: *Example where a linear classifier cannot separate both classes*

If, instead of directly using the values extracted from the dataset, we calculate the correspondent value in polar coordinates for each pair (x, y), our dataset generates the graph on Figure 3.2. Unlike the first figure, this graph can perfectly separate both categories in our dataset.

Therefore, using a different feature vector to represent the data in this example will increase significantly the accuracy of our classifier. For other aspects and further research in the area of feature selection and feature extraction the reader can refer to Guyon e Elisseeff (2003).

As W-operators are locally defined within a finite neighbourhood W, we restrict ourselves

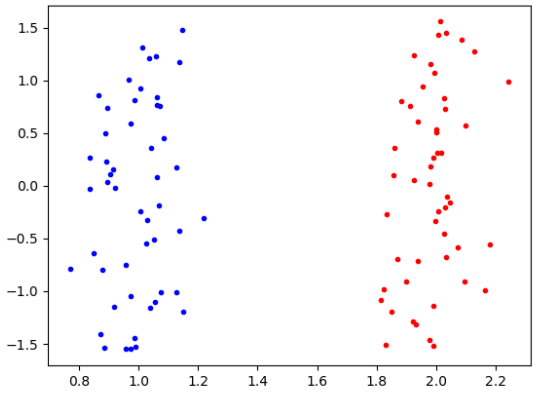


Figure 3.2: *Figure 3.1 after a change in coordinates, where a linear classifier can easily separate both classes*

to the extraction of local features, i. e., features that only extract information from within this window W. Therefore, an important hyperparameter of the learning process of this project is the size and shape of the window W. Aside from that, each feature can have its own hyperparameters.

The rest of this Chapter is dedicated to explain the 6 features that were implemented and the main idea and theory behind each one of them. Results of our experiments and comparisons between each feature are displayed on Chapter 4.

3.1 Feature Raw

The feature extractor that extracts the Raw feature was already implemented at TRIOSlib. This feature consists of extracting the intensity values of surrounding pixels within the window W, and it builds a feature vector by flattening this window. An example of this extraction can be seen in Fig. 3.3.

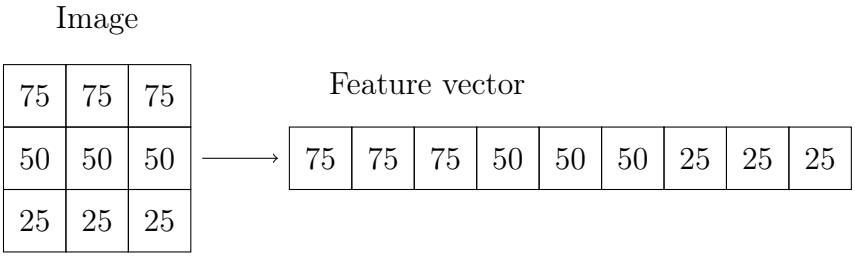


Figure 3.3: *Example of the Raw feature extractor*

This feature vector was the most used in previous works related to W-operator learning. Therefore, the operators learned using this feature extractor will be used as a baseline to our experiments on Chapter 4.

One major problem with this feature is that it limits the size of W, since it consists of every single pixel within the window. Therefore, the use of large windows become computationally impractical.

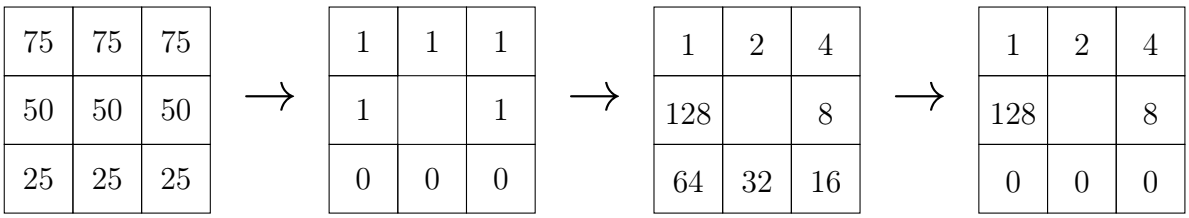
3.2 Feature Local Binary Pattern (LBP)

A different feature implemented in this project is based on the texture descriptor Local Binary Pattern (LBP), proposed by Ojala *et al.* (1994). In his article, Ojala evaluates the performance of many different texture descriptors in classification problems, some descriptors that were already used in various applications and some new approaches such as the LBP descriptor. The latter is based on a simplified version of the texture analysis model based on texture units, introduced by Wang e He (1990), in which the main idea is that 2D surface texture can be represented by two measures: local spatial pattern and gray scale contrast.

The original LBP, as described by Ojala in 1994, labeled each pixel using the following algorithm:

1. Threshold the 3x3 neighborhood of the pixel by it's intensity value
2. Multiply the thresholded pixels by the weight given to each correspondent pixel
3. Sum all the values of the thresholded neighborhood multiplied by the weights to obtain the texture unit

Figure 3.4 shows an example of the LBP descriptor algorithm. In the example, image 3.4a is thresholded into image 3.4b and then multiplied by image 3.4c. The result image is shown in 3.4d and the texture unit of the central pixel is the sum of all pixels in the result image, which is 143.



(a) *Image* **(b)** *Thresholded* **(c)** *Weigths* **(d)** *Result*

Figure 3.4: *Main algorithm of LBP descriptor*

LBP is gray scale invariant and locally defined, therefore it can be useful in our approach to learn W-operators. One of the disadvantages of this descriptor is that it only recognizes textures patterns that can be seen in a 3x3 neighborhood. This descriptor is also rotation variant, which can be undesirable in many applications. To surpass those problems, many different LBP variations were created throughout the years. For further details, the reader can refer to Pietikäinen *et al.* (2011), where it can be found an extensive list of LBP variations and many applications of this descriptor.

In our approach, we use the original LBP descriptor to create a feature vector for each pixel in the image. First, we use the LBP main algorithm to create a second image, where the value of each pixel is the correspondent texture unit. Then, for each pixel, we extract the corresponding W neighborhood in the image created by the LBP descriptor. The feature vector for that pixel is the flattened version of this neighborhood W.

3.3 Feature Moments

Both previous features extract information pixel by pixel within the window W. The Raw feature uses the intensity values of every pixel within the neighbourhood W and the

LBP feature is extracted by calculating a descriptor for every pixel inside the region. The Moments feature takes a different approach, based on image moments. Let us first describe what is an image moment:

The n^{th}-order moment of a continuous function $f(x, y)$ is defined as:

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy \tag{3.1}$$

where $n = p + q$. Therefore, if we consider the image as a function of two variables (x, y) and discretize the integral by replacing them with summations, the image moments of n^{th}-order can be defined as:

$$M_{pq} = \sum_{x=0}^{\infty} \sum_{y=0}^{\infty} x^p y^q f(x, y). \tag{3.2}$$

Moments of a function are particularly interesting because of the uniqueness theorem, proved by Hu (1962). This theorem states that, if the function $f(x, y)$ is piecewise continuous and has nonzero values only in the finite part of the xy-plane, then moments of all orders exists and the moments sequence is uniquely determined by $f(x, y)$ and, conversely, the function $f(x, y)$ is uniquely determined by the moments sequence.

Image moments can be used to calculate some properties of the image, such as area and centroid. The area of a binary image can be calculated by the zero-th moment, and the centroid in the x-axis and y-axis can be calculated by $\bar{x} = \frac{\mu_{10}}{\mu_{00}}$ and $\bar{y} = \frac{\mu_{01}}{\mu_{00}}$ respectively. Other properties of the image such as *skewness* and *kurtosis* can also be calculated by using moments of higher orders, but this is out of the scope of this project.

In our project we created a feature extractor that is based on the moments of an image. For each pixel, this feature extractor calculates all the moments of order n and lower, where n is a parameter previously given to the feature extractor with $n \geq 1$. To acquire translation-invariance, this feature extractor considers the position of the central pixel as being $(0, 0)$. Therefore, the pixel on the left of the central one is at position $(0, -1)$, the pixel right below the central is at position $(-1, 0)$ and so forth. The feature extractor also calculates the x-axis and y-axis of the centroid, and creates the following feature vector:

$$M = [f(x, y), \bar{x}, \bar{y}, \mu_{00}, \mu_{01}, ..., \mu_{pq}] \tag{3.3}$$

The following figure presents an example of our feature extractor based on moments:

75	75	75
50	50	50
25	25	25

(a) *Image*

$\mu_{00} = 450$
$\mu_{01} = -375$
$\mu_{10} = 225$
:

(b) *Moments*

$$\bar{x} = \frac{\mu_{10}}{\mu_{00}} = -0.83$$

$$\bar{y} = \frac{\mu_{01}}{\mu_{00}} = 0.5$$

(c) *Centroids*

50	-0.83	0.5	450	...

(d) *Feature Vector*

Figure 3.5: *Moments Feature*

In the example, Figure 3.5a shows a window W from an example image, fig. 3.5b displays the calculated moments of order 0 and 1, 3.5c the calculated centroid and the feature vector created for the central pixel of the example image is displayed on Figure 3.5d.

3.4 Feature Fourier

A signal is defined as a function that varies in time or space. The Fourier Transform is a transformation that takes as input a signal and decomposes that signal into sums of sines and cosines. This output is called the representation of the signal in the frequency domain. More information on the use of Fourier Transform in images can be seen in Gonzalez e Woods (2002).

Considering f as a signal, its Fourier Transform is defined as:

$$\begin{aligned} F(u) &= \int_{-\infty}^{\infty} f(x)e^{-i2\pi ux}dx \\ &= \int_{-\infty}^{\infty} f(x)(cos(2\pi ux) - isin(2\pi ux))dx \end{aligned} \tag{3.4}$$

where each point u represents a particular frequency contained in the spatial domain and the value of $F(u)$ represents how much the frequency u is present in function f. This operation is commonly used in signal processing and can be also applied in image processing, if we consider the image as a signal of two dimensions that varies in the space domain. The generalization of the Fourier Transform for two dimensions is:

$$\begin{aligned} F(u,v) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-i2\pi(ux+vy)} \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)(cos(2\pi(ux+vy)) - isin(2\pi(ux+vy)))dxdy \end{aligned} \tag{3.5}$$

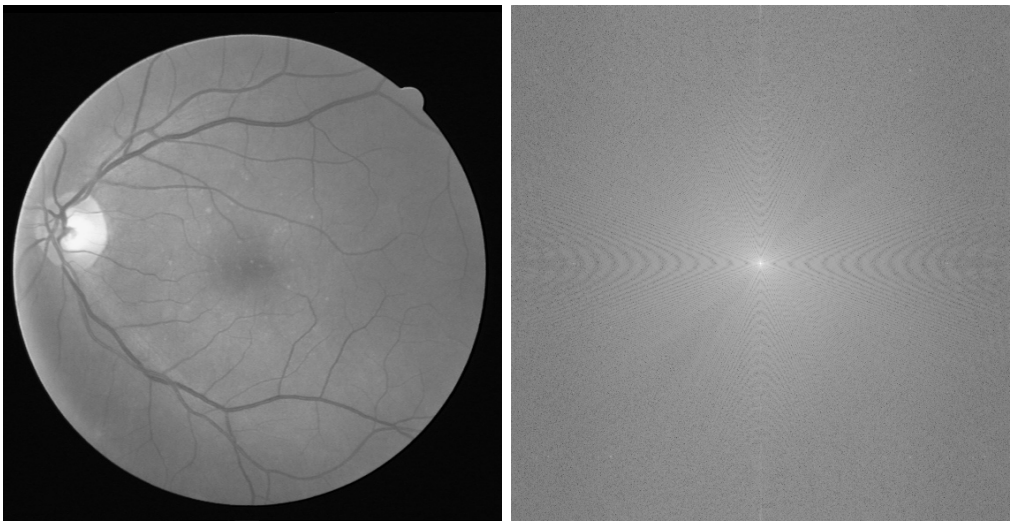
Digital images are defined only on discrete points, therefore, function f of equation 3.5 is not continuous and the Fourier Transform of it needs to be sampled. Substituting the integrals for summations in equation 3.5 results in the DFT (Discrete Fourier Transform), that can be used to find the representation of a digital image in the frequency domain.

$$F(u,v) = \sum_{j=0}^{N}\sum_{k=0}^{M} f(j,k)e^{-i2\pi(uj+vk)} \tag{3.6}$$

Nyquist-Shannon sampling theorem applies when using the sampled version of the Fourier Transform (further details on this theorem and on sampling theory can be found on Proakis e Manolakis (2007)). This theorem states that, given a function $f(t)$ with no frequencies higher than B, this function is completely determined by discrete points spaced by $1/(2B)$. An important consequence of this theorem is that it gives a threshold, called Nyquist frequency, where any frequency higher than that threshold cannot be perfectly represented in the sampling. Therefore, given an image of size NxM, vertical frequencies higher than $N/2$ and horizontal frequencies higher than $M/2$ cannot be represented in the Fourier Transform.

An example of the DFT applied in an image can be seen in Figure 3.6. Figure 3.6a shows an image from the DRIVE dataset(described in more details in Chapter 4), and Figure 3.6b shows the same figure in the frequency domain. In the figure of the frequency domain, each pixel represents a frequency, and its intensity value represents how much of that frequency is present in the original image. For a better visualization, the pixel representing the frequency zero is shifted to the center of the image.

We create a feature extractor based on the Fourier Transform in the following manner.



(a) *Image from DRIVE dataset* **(b)** *Spectrum of magnitude*

Figure 3.6: *Example of an image in the spatial and frequency domains*

For each pixel $p = (x, y)$ and a predetermined window W centered at (x, y), calculate the Fourier Transform of that region. For each coefficient c_{uv} found, where $c_{uv} = a + bi$ calculate the absolute value ($|c_{uv}| = \sqrt{a^2 + b^2}$). Then, create a feature with the first $NM/4$ coefficients, from c_{00} to $c_{\frac{N}{2}\frac{M}{2}}$, resulting in:

$$F = [|c_{00}|, |c_{01}|, |c_{02}|, ..., |c_{0\frac{N}{2}}|, |c_{10}|, |c_{11}|, ..., |c_{\frac{N}{2}\frac{M}{2}}|]. \tag{3.7}$$

Because of the Nyquist-Shannon sampling theorem, we only use the coefficients for the frequencies between 0 and $N/2$ and between 0 and $M/2$.

3.5 Feature Sobel

Previous researches in computer vision focused mainly in the detection of edges and corners. One of the most basic and most used edge detectors is the Sobel operator. The Sobel operator uses two kernels 3x3, defined as follow:

1	2	1
0	0	0
-1	-2	-1

(a) *Horizontal Sobel mask*

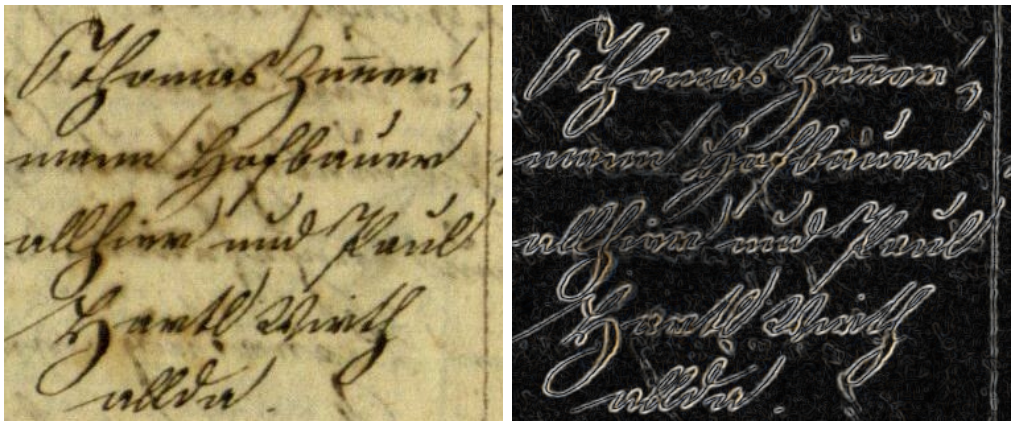
1	0	-1
2	0	-2
1	0	-1

(b) *Vertical Sobel mask*

Figure 3.7: *Sobel masks*

This masks are convolved with the image, each one creating a different output. Let G_y be the resulting image from the convolution of the horizontal sobel mask with the input image and G_x be the resulting image from the convolution with the vertical sobel mask, the result of the Sobel operator is calculated as:

$$S = \sqrt{G_x^2 + G_y^2} \tag{3.8}$$



(a) *DIBCO image* **(b)** *Result image*

Figure 3.8: *Example of the Sobel Operator*

An example of the Sobel operator can be seen in Figure 3.8. Figure 3.8a shows an image from the DIBCO dataset (described in Chapter 4), and Figure 3.8b shows the result of the application of the Sobel Operator in image 3.8a.

Our feature extractor based on this technique applies the Sobel operator in the whole image, and, for each pixel, extract the values of the pixels on the surrounding region and creates a feature vector with these values.

3.6 Feature Histogram of Oriented Gradients (HoG)

Histogram of Oriented Gradients (HoG) is a feature descriptor commonly used in object detection. It was first described in a pattent application by McConnell (1986), but its use in computer vision was only popularized by Dalal e Triggs (2005), where the feature descriptor was used for detection of pedestrian in images.

Extraction of HoG from an image consists of three steps. First, the gradient of the image is computed through the point discrete derivative masks in horizontal and vertical directions (Figure 3.9).The image is then divided in cells. For each cell, all the pixels within it calculates a weighted vote for an orientation, and these votes are accumulated in orientation bins that are evenly spread over 0 to 180 degrees. The last step is the normalization, where cells are grouped into blocks and each block is normalized separately. The HoG descriptor of a region on the image is the orientation bins of all the cells within that region.

-1	0	1

(a) *Horizontal derivative mask*

1
0
-1

(b) *Vertical derivative mask*

Figure 3.9: *Derivative masks*

We considered the region W too small to be further divided into blocks, therefore, our approach to create a feature extractor based on HoG is to calculate the gradient of the image, and, then, for each pixel, calculate the orientation bins in the surrounding region W.

After that, the bins were normalized, and the result is our feature vector. We tested two different normalizations methods, both described in equation 3.9.

$$\begin{aligned} \text{L1-sqrt:} \quad f &= \begin{cases} \sqrt{\frac{v}{||v||_1}} & \text{if} \quad ||v||_1 > 0 \\ 0 & \text{otherwise} \end{cases} \\ \text{L}\infty\text{:} \quad f &= \begin{cases} \frac{v}{max(v)} & \text{if} \quad max(v) > 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3.9}$$

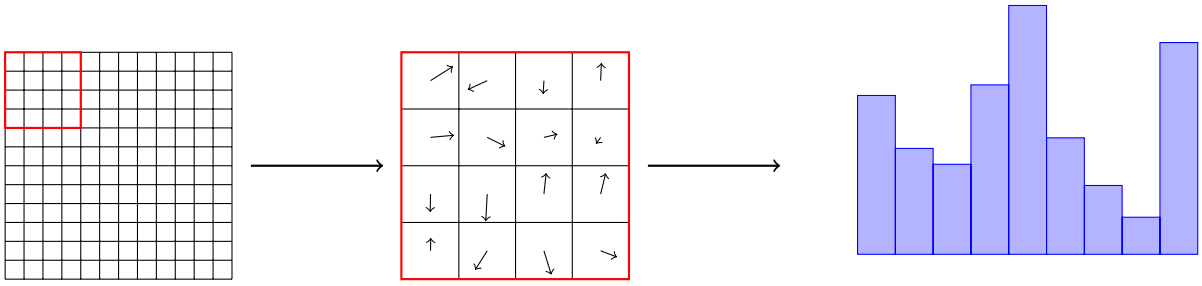


Image Orientation of pixels Orientation bins

Figure 3.10: *Overview of the HoG feature extractor*

Chapter 4

Experiments

The implementation of the features described in Chapter 3 were made on top of TRIOSlib, a toolbox implemented on Cython that contains state-of-art techniques on image operator learning. All of the features were tested on four different datasets, two that contains binary images and two of grayscale images. More details of the TRIOSlib library and of the datasets can be seen in sections 4.1 and 4.2.

The experiments were done with these two hyperparameters:

1. Window size: 3x3, 5x5, 7x7, 9x9 and 11x11
2. Classifier: Decision Trees

For each combination of these hyperparameters, the image operator was trained with the 6 implemented features in the training set of a specific dataset. Afterwards, the learned operator was applied to every image of the test set and the resulting image was saved. Every pixel of each image was then compared to the ground truth and classified in one of these four different classes:

- True positive: If both the classified pixel and the corresponding ground truth pixel have the value 1.
- True negative: If both the classified pixel and the corresponding ground truth pixel have the value 0.
- False positive: If the pixel was classified as 1 but the ground truth pixel has value 0.
- False negative: If the pixel was classified as 0 but the ground truth pixel has value 1.

The amount of pixels in each one of these four different classes (hereafter referred to as TP, TN, FP and FN, respectively) were used to calculate the five different evaluation metrics, that is, Precision, Recall, Specificity, F1-score and the Mean Absolute Error (MAE). All of these evaluation metrics are described in more detail on Section 4.3.

4.1 Training Image Operators from Samples (TRIOSlib)

TRIOSlib is a library implemented in Cython that contains state-of-art methods in image operator learning. This library was developed by many researchers throughout the years in the eScience group of the Institute of Mathematics and Statistics of the University of São Paulo.

TRIOSlib is open source and the full code is available at http://github.com/trioslib/trios. Currently in its second version, this library is implemented in Cython in order to have the expressiveness of Python while still having the efficiency of C. It also has the advantage of being easily integrated with many of today's most popular scientific library such as NumPy and SciPy.

There were a few feature extractors implemented in the library, but the Raw feature was the mainly used one in previous works. Therefore, this feature will be used in the experiments as a baseline for all of our results.

More details of this library are available at https://trioslib.github.io/, and a tutorial about using TRIOSlib with some experiments and results can be found at Montagner *et al.* (2016b).

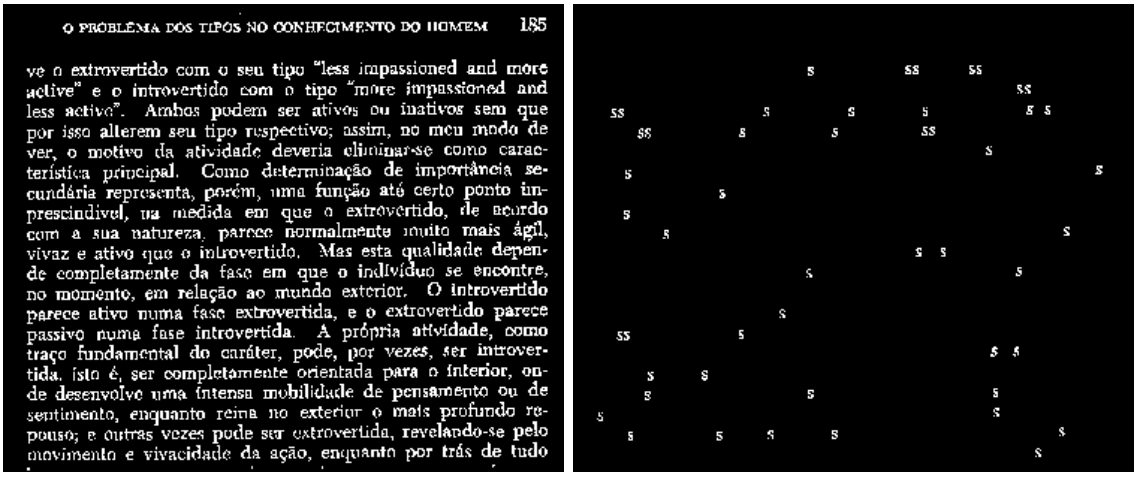
4.2 Datasets

All of the datasets used in these experiments are public and can be downloaded from the main page of this project. The datasets used were:

- Dataset Character Segmentation;
- Dataset Text Segmentation;
- Digital Retinal Images for Vessel Extraction;
- Document Image Binarization Competition.

The rest of this section describes each dataset in more detail and displays some sample images from each one.

4.2.1 Dataset Character Segmentation (CharS)



(a) *Input image* **(b)** *Output image*

Figure 4.1: *Input-output pair from the CharS dataset*

This dataset was obtained from scanned images of the book *"Tipos Psicológicos"*, from Jung (1967). It consists of 20 pages such as the one in Figure 4.1a and the correspondent 20 images of output such as in Figure 4.1b. In the experiments of this project, 10 different

images were used to the training process of the operator and the other 10 images were used in test the learned operator. Previous results of image operator learning using this dataset can be seen in Montagner *et al.* (2016b). In the rest of this monograph this dataset will be referred as *CharS*.

The main goal of this dataset is to extract the 's' characters in all of the images. An example of a typical input-output pair is displayed in Figure 4.1. Figure 4.1a shows a page of Jung's book and Figure 4.1b displays the same page, but with only the lower case 's' characters appearing.

CharS is the simplest dataset amongst the four used as it contains only binary images of a relative small size (375 x 313 pixels). Therefore, the training processes for this dataset are usually fast and very accurate.

4.2.2 Dataset Text Segmentation (TexRev)

(a) *Input* **(b)** *Output*

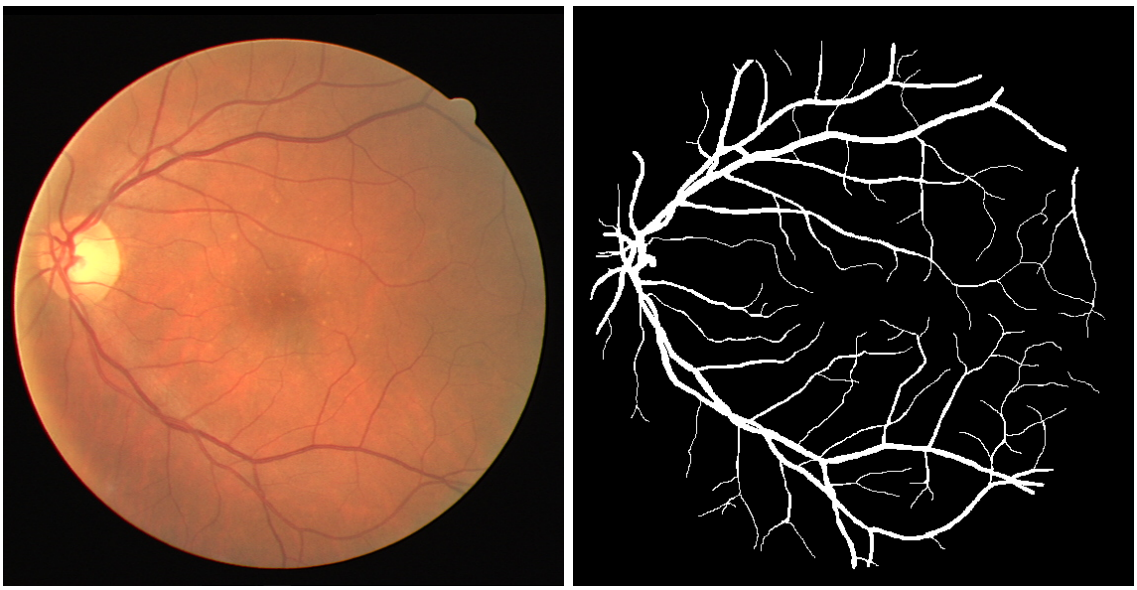
Figure 4.2: *Input-output pair of the TexRev dataset*

The dataset used for text segmentation (referred, from now on, as TexRev) was obtained from the Veja magazine *"Computador - o micro chega às casas"* from December of 1995. The main objective of this dataset is to be able to eliminate every part of an image that is not a text. Figure 4.2 displays an example of an input-output pair of this dataset.

Like the *CharS*, this dataset contains only binary images of different sizes, but smaller than 800 x 1100 pixels. While this dataset contains larger images than *CharS*, *TexRev* consists of only 10 pairs of input-output images. For the experiments, 5 images were used to train the operator and the remaining 5 were used to test the learned image operator.

This dataset was also used on Montagner *et al.* (2016b) and some previous results in operator learning using the raw feature are displayed there. However, in order to perform a more robust comparison between all different features, the experiments on Section 4.4 were all done using the same version of TRIOSlib instead of using previous obtained results.

4.2.3 Digital Retinal Images for Vessel Extraction (DRIVE)



(a) *Original image* **(b)** *Ground-Truth*

Figure 4.3: *Input-output pair of the DRIVE dataset*

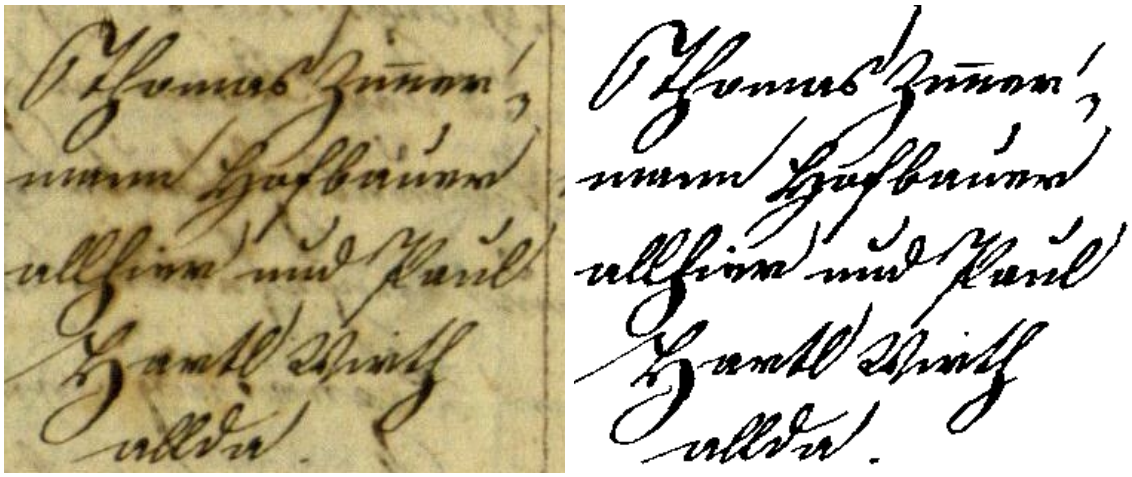
Digital Retinal Images for Vessel Extraction database (DRIVE, for short) is a well established dataset used for segmentation of blood vessels in retinal images and can be found in many state-of-art researches in this area. This dataset was constructed by Staal *et al.* (2004) to compare different methods of retinal vessel segmentation. The blood vessels of each image were manually segmented and the resulting image is treated as our ground truth. Figure 4.3 displays an example of an image and its ground truth that can be found on this database.

The DRIVE database is composed of 40 colored images, each one with a size of 565 x 584 pixels. To train the image operator we used 10 different images and to test the operator we used 20 other images. Even though the DRIVE database contains colored images, we decided to use a grayscale version of it to train our operators in order to be compliant to our previous defined features.

It's important to note that the DRIVE dataset, unlike the other ones, doesn't need to train and classify the whole image because of the existent margin between the circular retina and the rectangular matrix of the image. The DRIVE database constructed by Staal contains a mask for each image that defines where the retina is within it, thus, our image operator was trained and tested within the region determined by the masks.

4.2.4 Dataset Document Image Binarization Competition (DIBCO)

The Document Image Binarization Competition (DIBCO) is an annual contest, organized in conjunction with ICDAR (International Conference on Document Analysis and Recognition) and ICFHR (International Conference on Frontiers in Handwriting Recognition) that aims to identify the progress that are made in document image binarizations. Each year, the organizers of the contest provide sample images of documents and its correspondent ground truth image. The registered participants, then, submit their method of binarization back to the organizers, where the methods will be evaluated. After that, the organizers publicly



(a) *Original image* **(b)** *Ground-Truth*

Figure 4.4: *Input-output pair of the DIBCO dataset*

release the candidate methods, the testing set and the evaluation software. Previous results of the DIBCO contests can be seen in Ntirogiannis *et al.* (2014) and Pratikakis *et al.* (2016).

Figure 4.4 displays an example of a typical handwritten document image and its ground truth that can be found on DIBCO's training and testing set. This dataset contains images of varying sizes, ranging from around 900 x 600 to 2900 x 1060 pixels. For the experiments of this project, images from the 2016 and 2014 contests were taken, totalizing 10 images in the training set (6 from 2016's contest and 4 from the contest of 2014) and 7 images on the test set (4 from 2016 and 3 from 2014).

Just like the DRIVE dataset, DIBCO contains colored images but we decided to use the grayscale versions of it. This dataset was the largest one used in this experiments in terms of sizes of the image, thus, it was the most expensive in terms of memory and processing time.

It's important to note that the output of DIBCO's images have different values than the other dataset, i.e., the positive classification has value 0 and the negative has value 1. In order to have a proper meaning for the evaluation metrics, the values of FP and TP were changed with FN and TN, respectively.

4.3 Metrics

In each one of our experiments we calculated five different evaluation metrics, the precision, recall, sensitivity, F1-score and the mean absolute error. This section is dedicated to describe all these different metrics used. Here, classifying a pixel positively means that the value predicted by our operator was 1 (0 in the case of DIBCO's dataset), and a negative value means that our operator predicted the pixel as 0 (1 in the DIBCO's case).

4.3.1 Precision

The precision (also called positive predictive value) of a classification algorithm is calculated as:

$$P = \frac{TP}{TP + FP}. \tag{4.1}$$

This value shows the percentage of correctly positive predicted values over all the values that the classifier predicted as positive. If the precision of a classifier is 1, it means that, every pixel classified as positive is really positive, but doesn't mean anything about the amount of positive pixels on the image.

Precision is a really important evaluation metric, but its value alone can be very misleading. Suppose, for instance, that a classifier correctly predicts one pixel of positive value, but wrongly classifies every other pixel of the image as negative. The precision of this classifier will be 1, but it's not a really good classifier. Therefore, other metrics are needed to get a more accurate evaluation of the classifiers.

4.3.2 Recall

The recall of a classification algorithm is calculated as follows:

$$R = \frac{TP}{TP + FN} \tag{4.2}$$

The recall value shows the percentage of correctly positive classified pixels from all the positive pixels in the ground truth. A high value of recall means that the classifier can predict most of the positive values, but it doesn't inform anything about the negative values or the wrong positive predictions.

Just like the precision, the recall metric can be misleading. Suppose that a classifier predicts every pixel of the image as a positive one. Therefore there will be no false negative pixels and the value of recall will be 1.

4.3.3 Specificity

Specificity is an evaluation metric calculated as:

$$S = \frac{TN}{TN + FP} \tag{4.3}$$

The specificity value, sometimes called as true negative rate, calculates the percentage of correctly negative classified pixels, i. e., the amount of pixels that are truly negative among all the pixels that the classifier predicted as negative. This metric is similar to the recall value, but, instead of calculating the percentage of positive pixels, it calculates the percentage of negative pixels.

4.3.4 F1-Score

F1-Score is an interesting metric that calculates an harmonic average of the precision and recall, in order to minimizes the misleading aspects of those two metrics. It is calculated as follows:

$$F_1 = 2 * \frac{P * R}{P + R} \tag{4.4}$$

where P is the precision and R is the recall value.

The F_1 value reaches its best at 1 and its worst at 0, where 1 means perfect values of precision and recall.

4.3.5 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is one of the most common measures of accuracy. It measures the percent of miscalculated values among of them in the predicted image. In other words, is calculated as:

$$MAE = \frac{FP + FN}{FP + FN + TP + TN} \tag{4.5}$$

MAE is a very important evaluation metric as it calculates a percent of the wrong predictions. One of the problems of this metric appears when the resulting image has a bias towards one of the values, positive or negative. For instance, the output images of the *CharS* dataset have considerably more amount of negative values than positive ones. If a classifier predicts all of the pixels as being of negative value, it will have a good value of MAE, but it will be a useless classifier.

While the other metrics have it's perfect state (that is, the value that a perfect classifier would get) at 1, the opposite occurs with the MAE value, where the perfect classifier would have a MAE value of 0.

4.4 Results

Experiments showed a large disparity between the results of different datasets, i. e., features that resulted in a great performance in some datasets did not perform very well in the others. For that reason, the results of this section are separated by datasets.

CharS

CharS is a dataset with relative small images and the patterns to be recognized are not large. The goal of this dataset is to extract, from a binary textbook image, all the 's' characters. As the image is relatively small, deciding if the pixel belongs to a 's' character can be done by using only a small region around it. Therefore, it's expected that the features will perform really well even with small windows sizes.

Figure 4.5 displays a chart with the MAE in the Y-axis and the size of one side of window W in the X-axis (that is, 3 in the X-axis means that, using a 3x3 window resulted in an operator with the correspondent MAE on the Y-axis).

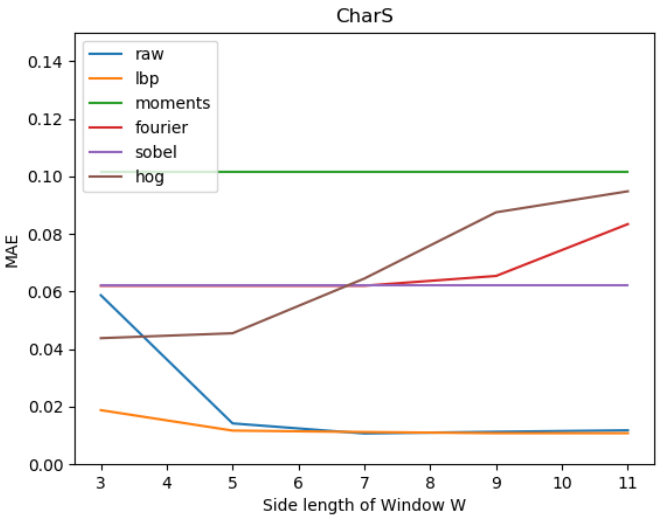


Figure 4.5: *Results of CharS*

The best features were Raw and LBP, and increasing the window size didn't improve that MAE value significantly. That was expected as the texture of the characters are small enough to be described by the LBP.

Although the MAE values on the chart indicate that all features performed really well in this dataset, the value of the Sobel feature is misleading. The operator learned with this

Window Size	Feature Extractor	MAE	Precision	Recall	Specificity	F1-value
3x3	Raw	0.0587	0.7647	0.0751	0.9985	0.1365
	Lbp	0.0188	0.8741	0.8136	0.9923	0.8424
	Moments	0.0998	0.2350	0.2732	0.9417	0.2521
	Fourier	0.0620	0.0000	0.0000	1.0000	0.0000
	Sobel	0.0620	0.0000	0.0000	1.0000	0.0000
	Hog	0.0443	0.6917	0.5107	0.9851	0.5872
5x5	Raw	0.0142	0.8796	0.8933	0.9919	0.8861
	Lbp	0.0117	0.8956	0.9174	0.9930	0.9061
	Moments	0.0998	0.2350	0.2732	0.9417	0.2521
	Fourier	0.0620	0.0000	0.0000	1.0000	0.0000
	Sobel	0.0620	0.0000	0.0000	1.0000	0.0000
	Hog	0.0485	0.6106	0.6041	0.9744	0.6066
7x7	Raw	0.0107	0.9027	0.9270	0.9934	0.9145
	Lbp	0.0112	0.8993	0.9224	0.9932	0.9104
	Moments	0.0998	0.2350	0.2732	0.9417	0.2521
	Fourier	0.0620	0.0000	0.0000	1.0000	0.0000
	Sobel	0.0620	0.0000	0.0000	1.0000	0.0000
	Hog	0.0689	0.4465	0.4562	0.9625	0.4501
9x9	Raw	0.0113	0.8994	0.9210	0.9932	0.9099
	Lbp	0.0108	0.9062	0.9211	0.9937	0.9132
	Moments	0.0998	0.2350	0.2732	0.9417	0.2521
	Fourier	0.0654	0.4650	0.3768	0.9714	0.4160
	Sobel	0.0620	0.0000	0.0000	1.0000	0.0000
	Hog	0.0898	0.2812	0.2886	0.9512	0.2841
11x11	Raw	0.0118	0.8916	0.9226	0.9926	0.9066
	Lbp	0.0108	0.9062	0.9211	0.9937	0.9132
	Moments	0.0998	0.2350	0.2732	0.9417	0.2521
	Fourier	0.0834	0.3427	0.3787	0.9521	0.3592
	Sobel	0.0620	0.0000	0.0000	1.0000	0.0000
	Hog	0.0975	0.2294	0.2426	0.9460	0.2349

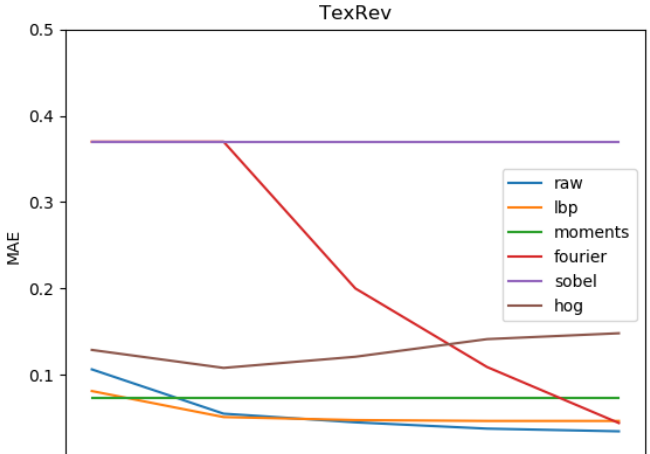
Table 4.1: *Results of CharS experiments*

feature returned an all black image, therefore, a worthless operator. As the output image contains more black pixels than white ones, this operator acquired a deceptive MAE value.

To enable further inspections on the operators learned in this dataset, the values of MAE, Precision, Recall, Specificity and F1-measure are displayed in Table 4.1.

TexRev

TexRev is also a binary dataset, but with patterns that were not easily discriminated by small regions. In this dataset, the main goal is to select only the text of an image from a magazine page. As in the results of the operators for the *CharS* dataset, Figure 4.6 dis-



plays a chart with the MAE value of the operators learned with different features.

Just like the *CharS* dataset, the operator learned with the Sobel feature for the *TexRev* dataset returned black images, and the Raw and LBP feature performed really well.

The interesting result in this dataset is the operator learned with the Fourier feature. The window size on this feature was crucial to its performance, improving almost 30% between the 5x5 window and the 11x11 window. The good performance of this feature can be explained by the difference of frequencies between an image and a character, since a letter of a text contains higher frequencies than an image in the *TexRev* dataset. Due to the Nyquist Frequency, in order to recognize a pattern on the frequency domain of a region, the size of W must be considerably large, what explains the improvement between 5x5 and 11x11 windows.

Table 4.2 displays all the results from the experiments with the *TexRev* dataset.

Window Size	Feature Extractor	MAE	Precision	Recall	Specificity	F1-value
3x3	Raw	0.1064	0.8538	0.8452	0.9189	0.8467
	Lbp	0.0813	0.9353	0.8378	0.9648	0.8818
	Moments	0.0747	0.9416	0.8559	0.9636	0.8954
	Fourier	0.3699	0.0000	0.0000	1.0000	0.0000
	Sobel	0.3699	0.0000	0.0000	1.0000	0.0000
	Hog	0.1289	0.9127	0.7111	0.9601	0.7976
5x5	Raw	0.0551	0.9523	0.8939	0.9729	0.9212
	Lbp	0.0512	0.9639	0.8962	0.9798	0.9280
	Moments	0.0747	0.9416	0.8559	0.9636	0.8954
	Fourier	0.3699	0.0000	0.0000	1.0000	0.0000
	Sobel	0.3699	0.0000	0.0000	1.0000	0.0000
	Hog	0.1081	0.9048	0.7798	0.9546	0.8366
7x7	Raw	0.0451	0.9663	0.9118	0.9808	0.9378
	Lbp	0.0477	0.9668	0.9033	0.9816	0.9334
	Moments	0.0747	0.9416	0.8559	0.9636	0.8954
	Fourier	0.2001	0.7294	0.6979	0.8499	0.7104
	Sobel	0.3699	0.0000	0.0000	1.0000	0.0000
	Hog	0.1210	0.8793	0.7689	0.9413	0.8193
9x9	Raw	0.0378	0.9714	0.9282	0.9827	0.9490
	Lbp	0.0467	0.9674	0.9057	0.9820	0.9349
	Moments	0.0747	0.9416	0.8559	0.9636	0.8954
	Fourier	0.1091	0.8949	0.7977	0.9442	0.8423
	Sobel	0.3699	0.0000	0.0000	1.0000	0.0000
	Hog	0.1414	0.8468	0.7370	0.9278	0.7865
11x11	Raw	0.0347	0.9736	0.9359	0.9829	0.9541
	Lbp	0.0467	0.9674	0.9057	0.9820	0.9349
	Moments	0.0747	0.9416	0.8559	0.9636	0.8954
	Fourier	0.0443	0.9670	0.9154	0.9781	0.9402
	Sobel	0.3699	0.0000	0.0000	1.0000	0.0000
	Hog	0.1482	0.8361	0.7251	0.9238	0.7746

Table 4.2: *Results of TexRev experiments*

DRIVE

DRIVE is a dataset where the goal is the segmentation of blood vessels in digital retinal images. Distinct from the previous datasets, the DRIVE dataset contains grayscale images and the images from DRIVE are also larger than the images from *CharS*. Thus, it is expected that learning the image operator on this dataset is a harder problem than in the previous ones.

The operators trained on the

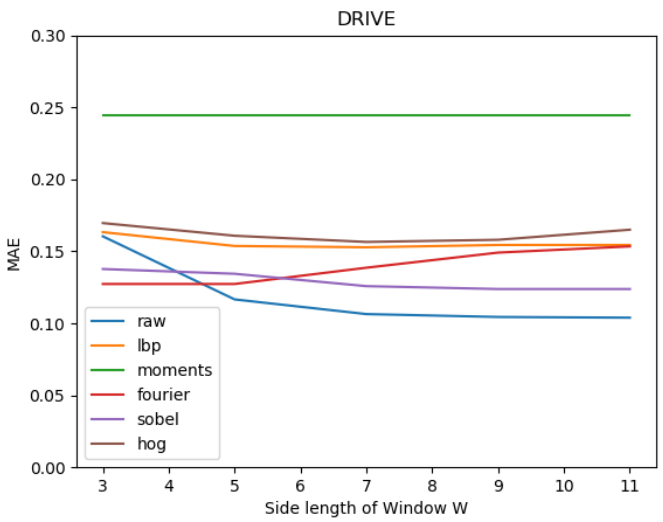


Figure 4.7: *Results of DRIVE*

DRIVE dataset presented the worst results amongst all the datasets. The task itself is a hard one, as many specialists do not agree on the segmentation of the vessels. One of the difficulties on DRIVE is the small contrast between the vessels and the background, which makes it difficult to separate both.

Figure 4.7 displays the results obtained from the experiments on this dataset. Apart from the Sobel feature, all the features had a significant decrease on the performance compared to the accuracy obtained on the binary datasets. Like the previous datasets, Raw had the best performance amongst all of the features, but the other features had diverse results. Sobel feature performed considerably better on the grasycale dataset, while the LBP feature, that previously had a performance similar to Raw, was considerably worse.

Similar to the previous shown results, table 4.3 displays the measures calculated from the experiments on this dataset.

DIBCO

The objective on the DIBCO dataset is the segmentation of text from a document image. Although the images on this dataset are larger than the other ones, the task on this dataset is considerably easier than DRIVE due to the large contrast between the segmentation target and the background.

As can be seen in Figure 4.8, Raw and Moments had similar MAE and they were the best performing features in this dataset. The other features also had similar results, around 0.13 of MAE on the 11x11 region.

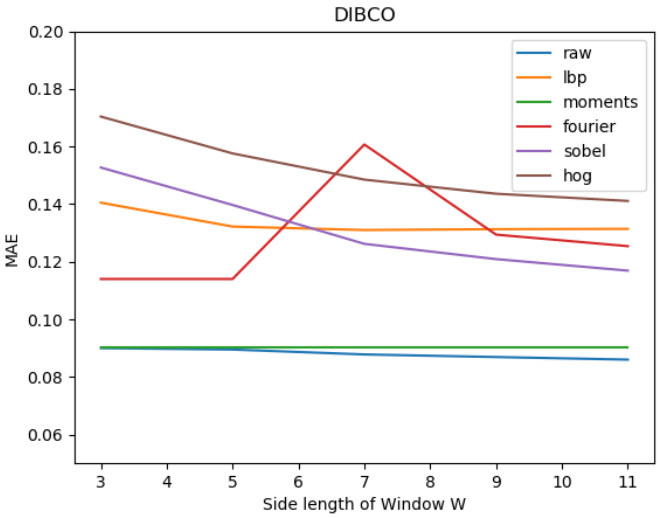


Figure 4.8: *Results of DIBCO*

Although this dataset had better results than the DRIVE dataset, the memory and time consumption on DIBCO was considerably larger and, at sometimes, the training of the operator almost became unfeasible. This happened due to the size of the images on this dataset, that were the largest of all the four datasets.

Table 4.4 displays the calculated measures of all features in all trained windows, in order to enable further inspections.

Window Size	Feature Extractor	MAE	Precision	Recall	Specificity	F1-value
3x3	Raw	0.1604	0.3966	0.4785	0.8923	0.4303
	Lbp	0.1633	0.3746	0.4240	0.8969	0.3969
	Moments	0.2449	0.1338	0.1675	0.8408	0.1480
	Fourier	0.1273	0.0000	0.0000	1.0000	0.0000
	Sobel	0.1377	0.4633	0.4843	0.9175	0.4707
	Hog	0.1731	0.3280	0.3431	0.8975	0.3341
5x5	Raw	0.1166	0.5419	0.6133	0.9230	0.5712
	Lbp	0.1537	0.4091	0.4677	0.9015	0.4352
	Moments	0.2449	0.1338	0.1675	0.8408	0.1480
	Fourier	0.1273	0.0000	0.0000	1.0000	0.0000
	Sobel	0.1344	0.4784	0.5714	0.9086	0.5182
	Hog	0.1663	0.3595	0.3919	0.8982	0.3737
7x7	Raw	0.1064	0.5775	0.6469	0.9298	0.6064
	Lbp	0.1528	0.4125	0.4744	0.9017	0.4401
	Moments	0.2449	0.1338	0.1675	0.8408	0.1480
	Fourier	0.1386	0.3651	0.1189	0.9697	0.1777
	Sobel	0.1258	0.5070	0.5971	0.9147	0.5457
	Hog	0.1612	0.3792	0.4178	0.9002	0.3961
9x9	Raw	0.1044	0.5843	0.6555	0.9308	0.6140
	Lbp	0.1544	0.4070	0.4693	0.9005	0.4348
	Moments	0.2449	0.1338	0.1675	0.8408	0.1480
	Fourier	0.1491	0.4269	0.4829	0.9046	0.4504
	Sobel	0.1238	0.5136	0.6017	0.9163	0.5517
	Hog	0.1633	0.3725	0.4133	0.8984	0.3905
11x11	Raw	0.1039	0.5865	0.6553	0.9314	0.6153
	Lbp	0.1544	0.4070	0.4693	0.9005	0.4348
	Moments	0.2449	0.1338	0.1675	0.8408	0.1480
	Fourier	0.1534	0.4115	0.4597	0.9030	0.4315
	Sobel	0.1238	0.5134	0.6018	0.9163	0.5517
	Hog	0.1695	0.3502	0.3874	0.8951	0.3666

Table 4.3: *Results of DRIVE experiments*

Window Size	Feature Extractor	MAE	Precision	Recall	Specificity	F1-value
3x3	Raw	0.0900	0.4622	0.7334	0.9288	0.5348
	Lbp	0.1405	0.2633	0.4477	0.8967	0.3108
	Moments	0.0906	0.4722	0.6591	0.9311	0.5201
	Fourier	0.1140	0.0026	0.0009	0.9677	0.0013
	Sobel	0.1527	0.2842	0.6130	0.8756	0.3541
	Hog	0.1704	0.2209	0.4545	0.8670	0.2736
5x5	Raw	0.0895	0.4638	0.7454	0.9291	0.5387
	Lbp	0.1322	0.2863	0.4687	0.9042	0.3337
	Moments	0.0906	0.4722	0.6591	0.9311	0.5201
	Fourier	0.1140	0.0026	0.0009	0.9677	0.0013
	Sobel	0.1397	0.3086	0.6331	0.8872	0.3790
	Hog	0.1576	0.2550	0.5249	0.8754	0.3177
7x7	Raw	0.0878	0.4681	0.7465	0.9306	0.5437
	Lbp	0.1310	0.2891	0.4671	0.9057	0.3350
	Moments	0.0906	0.4722	0.6591	0.9311	0.5201
	Fourier	0.1607	0.1950	0.3336	0.8869	0.2226
	Sobel	0.1262	0.3405	0.6546	0.8998	0.4116
	Hog	0.1485	0.2661	0.5371	0.8838	0.3320
9x9	Raw	0.0869	0.4720	0.7434	0.9315	0.5467
	Lbp	0.1313	0.2892	0.4664	0.9054	0.3349
	Moments	0.0906	0.4722	0.6591	0.9311	0.5201
	Fourier	0.1294	0.3234	0.6127	0.8961	0.3921
	Sobel	0.1209	0.3535	0.6668	0.9038	0.4261
	Hog	0.1436	0.2679	0.5180	0.8906	0.3316
11x11	Raw	0.0860	0.0288	0.2562	0.0680	0.0503
	Lbp	0.1314	0.2890	0.4664	0.9053	0.3347
	Moments	0.0906	0.4722	0.6591	0.9311	0.5201
	Fourier	0.1254	0.3241	0.5379	0.9072	0.3810
	Sobel	0.1169	0.3641	0.6715	0.9077	0.4381
	Hog	0.1411	0.2679	0.4923	0.8955	0.3266

Table 4.4: *Results of DIBCO experiments*

Chapter 5

Conclusion and Future Works

In this project we studied the use of different features and representations in the process of learning image operators. Although the framework of this task was studied and improved in many previous works, one problem still remains, and that is the curse of dimensionality in the features extracted from the training images. The use of different features with smaller sizes and similar accuracy could enable the framework to learn image operators from larger regions than before.

We studied and manually implemented many features, in order to be consistent with TRIOSlib, a library that was implemented for the task of image operator learning. For the experiments we searched and utilized public datasets so all of our experiments can be easily reproduced by any interested reader. The code for TRIOSlib and every feature implemented in this project are available at github. All test images can be seen in the project's page.

The experiments showed that, while the raw feature provide the best overall accuracy, different features provide similar results in specific datasets with smaller sizes of vectors. For instance, the Fourier feature is a quarter of the size of the Raw feature and it performed nearly the same on the TexRev dataset.

This diversity of results produced by the different features showed that, given a dataset, one particular feature can be used to minimize the dimensionality of the training data while still having a reasonable performance on the resulting operator. The study of the best feature to be used in a dataset and the automatic selection of the features are some of the problems that can be tackled in future works.

Future works that can be done in this project include the study of combination of the features extracted from the training images. Such combination can improve the accuracy on some datasets or even decrease the size of feature vectors. Previous works on this subject included the combination of different windows and it obtained a significantly improvement on previous results.

Project's page: https://linux.ime.usp.br/~augustocms/mac0499/.

Bibliography

D. Julca-Aguilar e Hirata(2017) Frank D. Julca-Aguilar e Nina Hirata. Image operator learning coupled with cnn classification and its application to staff line removal. Cited on pag. 4, 8

Dalal e Triggs(2005) Navneet Dalal e Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893. Cited on pag. 17

Dornelles e Hirata(2015) M. M. Dornelles e N. S. T. Hirata. Selection of windows for W-operator combination from entropy based ranking. In *28th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 64–71. doi: 10.1109/SIBGRAPI.2015.41. Cited on pag. 3

Gonzalez e Woods(2002) Rafael C. Gonzalez e Richard E. Woods. *Digital Image Processing (2nd Ed)*. Prentice Hall. Cited on pag. 15

Guyon e Elisseeff(2003) Isabelle Guyon e André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944968. Cited on pag. 11

Heijmans(1994) Henk JAM Heijmans. Morphological image operators. *Advances in Electronics and Electron Physics Suppl., Boston: Academic Press,| c1994*. Cited on pag. 4, 5

Hu(1962) Ming-Kuei Hu. Hu, m.k.: Visual patter recognition by moments invariants. ire transaction of information theory it-8. 8:179 – 187. Cited on pag. 14

Jung(1967) C.G. Jung. *Tipos psicológicos*. GUANABARA. ISBN 9788524504396. URL https://books.google.com.br/books?id=m1dQPQAACAAJ. Cited on pag. 20

Junior Barrera(1997) Nina Sumiko Tomita Junior Barrera, Edward R. Dougherty. Automatic programming of binary morphological machines by design of statistically optimal operators in the context of computational learning theory. *Journal of Electronic Imaging*, 6:6 – 6 – 14. doi: 10.1117/12.260010. URL http://dx.doi.org/10.1117/12.260010. Cited on pag. 1, 3, 5

McConnell(1986) R.K. McConnell. Method of and apparatus for pattern recognition, January 28 1986. URL http://www.google.co.uk/patents/US4567610. US Patent 4,567,610. Cited on pag. 17

Montagner *et al.*(2016a) I. S. Montagner, N. S. T. Hirata, R. Hirata e S. Canu. NILC: A two level learning algorithm with operator selection. In *IEEE International Conference on Image Processing (ICIP)*, pages 1873–1877. doi: 10.1109/ICIP.2016.7532683. Cited on pag. 3

Montagner *et al.***(2016b)** I. S. Montagner, Nina S. T. Hirata e R. Hirata Jr. Image operator learning and applications. In *29th Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*. Cited on pag. 3, 20, 21

Nina(2009) N. S. T. Nina. Multilevel training of binary morphological operators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):707–720. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.118. Cited on pag. 1, 4, 8

Ntirogiannis *et al.***(2014)** K. Ntirogiannis, B. Gatos e I. Pratikakis. ICFHR2014 Competition on Handwritten Document Image Binarization (H-DIBCO 2014). In *14th International Conference on Frontiers in Handwriting Recognition*, pages 809–813. doi: 10.1109/ICFHR.2014.141. Cited on pag. 23

Ojala *et al.***(1994)** Timo Ojala, Matti Pietikainen e David Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 582–585. IEEE. Cited on pag. 13

Pietikäinen *et al.***(2011)** M. Pietikäinen, A. Hadid, G. Zhao e T. Ahonen. *Computer Vision Using Local Binary Patterns*. Computational Imaging and Vision. Springer London. ISBN 9780857297471. URL https://books.google.com.br/books?id=3iu-NAEACAAJ. Cited on pag. 13

Pratikakis *et al.***(2016)** I. Pratikakis, K. Zagoris, G. Barlas e B. Gatos. ICFHR2016 Handwritten Document Image Binarization Contest (H-DIBCO 2016). In *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 619–623. doi: 10.1109/ICFHR.2016.0118. Cited on pag. 23

Proakis e Manolakis(2007) J.G. Proakis e D.G. Manolakis. *Digital Signal Processing*. Pearson Prentice Hall. ISBN 9780132287319. URL https://books.google.com.br/books?id=twtGPwAACAAJ. Cited on pag. 15

Soille(2002) Pierre Soille. *Morphological Image Analysis*. Springer. Cited on pag. 1

Staal *et al.***(2004)** J.J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever e B. van Ginneken. Ridge based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4):501–509. Cited on pag. 22

Wang e He(1990) Li Wang e Dong-Chen He. Texture classification using texture spectrum. *Pattern Recogn.*, 23(8):905–910. ISSN 0031-3203. doi: 10.1016/0031-3203(90)90135-8. URL http://dx.doi.org/10.1016/0031-3203(90)90135-8. Cited on pag. 13