UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Graph-based segmentation of letters in handwriting words with Lucas Kanade template warping

Dissertation presented by
Clara SANSALVADÓ

for obtaining the master's degree in
Electrical Engineering

Supervisor
Cristophe DE VLEESCHOUWER

Readers
Benoît MACQ, Marie VAN REYBROECK

Academic year 2017 - 2018

GRAPH-BASED SEGMENTATION OF LETTERS IN HANDWRITING WORDS WITH LUCAS KANADE TEMPLATE WARPING

ABSTRACT

Handwriting Character Recognition (HCR) is the task of detecting and recognizing characters or symbols in a handwritten material, captured by some physical device.

This thesis develops a new tool of HCR for a particular context in which the input text in the handwritten sources is known a priori since it results from a dictation task. We also assume in this work that the beginning and the end of the words have been identified. In contrast to previous related works, the objective of this method is to determine the segmentation of the words into letters instead of aiming to decode its content.

We work with handwriting data recorded by a pressure-sensitive touchscreen device. We reconstruct an image of the handwritten contents and compute its distance transform. Our major contribution is to combine template matching techniques with graph theory to identify the letters on the processed image. Specifically, we first warp reference letter templates using the Lucas-Kanade optimization algorithm. In a second step, we consider that each warped template can be translated locally, and build a graph that connects all possible displacements of consecutive letters. Costs of edges in the graph account for (i) the Normalized Cross Correlation between each translated template and the word to segment and (ii) the distance between the end of a template and the beginning of the next one; mainly given by the sequential order of the letters in the word. The segmented solution is obtained on the graph representation by means of the Dijkstra shortest-path algorithm. The shortest-path solution jointly optimizes the similarity of the handwritten letters with warped templates and optimizes the spatial coherence between the letters.

This method is designed to be robust in front of a high degree of variability in the handwriting style. Doing so, we want to make it applicable to segment handwritten words from children who are still acquiring handwriting skills; especially to make it applicable to handwritten words from children with dyslexia, whose writing performance at young ages is remarkably poorer.

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my advisor Christophe De Vleeschouwer. He has been an essential reference for me during these months and he has brought an immeasurable help for the development of this master thesis. He has been constantly giving me good answers whenever I had some doubts and some inspiring suggestions to escape form situations in which I was feeling stacked. He has offered me a constant supervision of the work, through regular meetings in which I have been able to discuss with him new progresses or ideas; what intrinsically means that he has rewarded me with a lot of his time and even a bigger amount of significant ideas and knowledge.

This thesis would definitively not have been possible without his constant support. But further from this work, without his trust in my person to participate in this research and his motivation to do it, my entire stay in Belgium would not have been possible either. Thus I can avoid being also grateful for the opportunity that he has given to me; an opportunity that I have truly enjoyed and converted into one the most enriching experiences until now.

In addition, I am grateful to Claire Gosse for introducing me to the context of this thesis and for her help during these months. She has familiarized me with the main framework of research and she has also been inspiring to me. I also would like to thank Simon Carbonnelle for making my job possible. He has given me the chance to participate in this topic and he has provided me all the necessary data to work with.

I am grateful to the jury members, Marie Van Reybroeck and Benoît Macq, for accepting to evaluate this work and show their interest towards the project. I believe that the quality of this thesis is improved because of their constructive comments.

I have developed this thesis within a mobility program. During my stay in Belgium, I have also received the support from my friends that have highly contributed to enrich this unforgettable experience. Furthermore, my time at UCL has been made incredibly pleasant thanks to them, through working and research hours combined with sport, board games, food, travelling and cultural exchange.

I also want to express my deep gratitude to my family, specially my parents, Marc and Gemma. They also have been a key factor to make this experience possible. During all these years, they have given me all their love and have supported me in my academic and professional development and also in all personal experiences. As it could not be better, they are offering me this time too, their warmest feelings and enthusiasm by travelling to be with me in one of the greatest steps that I am going through: The end of my academic years and the beginning of a new chapter of my life; of which I truthfully hope they will continue being such an essential part.

Finally, I have to immensely express my gratitude to Ignasi. With him being with me, this experience has been amazing; and so is the decision that he made of coming with me to be a part of it. I appreciate and thank that he left all behind to spend these months living with me in a different country and with me being involved in this research. Our adventures, trips, excursions, shared moments, his belief on me and his delicious recipes have been an incredibly powerful source of energy and have made me get the best of me for this thesis development. He has made me happier every day and I would not have been able to get this far without his support.

CONTENTS

NOTATION AND ABREVIATIONS

List of abbreviations

- AA ..... Analysis Area
- ANN ..... Artificial Neural Networks
- App ..... Application
- BF ..... Brute Force
- BFS ..... Breath-First Search
- BW ..... Binary Image (Black and White Image)
- CC ..... Cross Correlation
- CSV ..... Comma Separated Values (.csv)
- DFS ..... Depth First Search
- DM ..... Distance Map (also Distance Transform, DT)
- DOF ..... Degree(s) Of Freedom
- DT ..... Distance Transform (also Distance Map, DM)
- FCC ..... Fast Cross Correlation
- FNCC ..... Fast Normalized Cross Correlation
- GUI ..... Graphical User Interface
- HWR ..... Handwriting recognition
- I ..... Image (also Source Image)
- IP ..... Image Patch (also Image Window, IW)
- IW ..... Image Windows (also Image Patch, IP)
- LK ..... Lukas Kanade
- MAT ..... Matlab Variable
- NCC ..... Normalized Cross Correlation
- OCR ..... Optical Character Recognition
- SI ..... Source Image (also Image)
- SSD ..... Sum of Squared Differences
- SW ..... Search Window
- T ..... Template
- TIF ..... Tagged Image File (.tif)

Notation

- $\mathbf{x} = (x_1, x_2, \dots, x_n)$ ..Vector
- $\mathbf{x}^*$ .....Vector evaluated in a particular point $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_n^*)$
- $f(\mathbf{x})$ .....Function
- $f(\mathbf{x}^* + \boldsymbol{h})$ .....Function around a specific point $\mathbf{x}^*$
- $\Delta\mathbf{x}$ .....Increase of the vector parameters $\Delta\mathbf{x} = (\Delta x_1, \Delta x_2, \dots, \Delta x_n)$
- $\alpha \in [\underline{\alpha}, \overline{\alpha}]$ .....Range within the variable α exists
- $\overline{\alpha}$ .....Maximum value that variable α can take in its domain
- $\underline{\alpha}$ .....Minimum value that variable α can take in its domain
- $\hat{\alpha}$ .....Mean value of a variable α
- $\check{\alpha}$ .....Median statistical value of a variable α
- $\tilde{\alpha}$ .....Mean-subtracted variable α, which belongs to $\tilde{\alpha} \in [\underline{\alpha} - \hat{\alpha}, \overline{\alpha} - \hat{\alpha}]$
- *(x,y)* .....Fixed 2D coordinates of an image
- *(x',y')*.....Fixed 2D coordinates result of a transform applied to *(x,y)*
- *(u,v)* .....Translational variable on the 2D coordinates of an image
- $\{x, y\}_{R.F.}$ .....*(x,y)* coordinates in the reference frame RF.
- $\nabla f(\mathbf{x})$ .....Gradient of a function
- $\mathbf{H}(f(\mathbf{x}))$.....Hessian of a function
- $\mathbf{W}$ .....Warping
- $\boldsymbol{p}$ .....Warping parameters vectors $\mathbf{p} = (p_1, p_2, \dots, p_n)$
- $\boldsymbol{W}(i([x, y]; \boldsymbol{p}))$.....Warping of the Image *i* under the parameter vector $\mathbf{p}$
- θ .....Rotation angle
- T_1.....Horizontal component of the translation vector.
- T_2.....Vertical component of the translation vector.
- K_1 .....Horizontal component of the scaling vector.
- K_2 .....Vertical component of the scaling vector.
- S_1 .....Horizontal component of the shearing vector.
- S_2 .....Vertical component of the shearing vector.
- $i(x, y)$.....Image function.
  Note that $i(x, y) = i(\boldsymbol{x}) = i$
- $\hat{\imath}$.....Image mean
- $\hat{\imath}_{u,v}$ .....Image mean within a particular Image Window
- $\tilde{\imath}(x, y)$ .....Mean-subtracted Image function
- $\tilde{\imath}_{\boldsymbol{u},\boldsymbol{v}}$ .....Locally mean-subtracted image function within a particular IW.
- $t(x, y)$.....Template image function.
  Note that $t(x, y) = t(\boldsymbol{x}) = t$
- $\hat{t}$.....Template image mean
- $\tilde{t}(x, y)$.....Mean-subtracted Template image function
- $e(\boldsymbol{p})$.....Optimization function.
- $i_x(x, y)$ .....Horizontal gradient of the Image function $i_x(x, y) = \frac{\partial i(x,y)}{\partial x}$
  Note that $i_x(x, y) = i_x(\boldsymbol{x}) = i_x$

- $i_x(x,y)$ .....Vertical gradient of the Image function $i_y(x,y) = \frac{\partial i(x,y)}{\partial y}$
  Note that $i_y(x,y) = i_y(\boldsymbol{x}) = i_y$
- N.....Number of rows of a digital image
- M.....Number of columns of a digital image
- $\boldsymbol{\nabla} i$ .....Image gradient $\nabla i = \left[\frac{\partial i}{\partial x}\ \frac{\partial i}{\partial y}\right]$
- $\boldsymbol{J}$ .....Jacobian of the warping $J = \frac{\partial W}{\partial \vec{p}}$
- $P_{i,m}$.....Graph node *m* that belongs to Search Window of letter *i*
- $D_{i,m-j,n}$ .....Graph edge that connects nodes $P_{i,m}$ and $P_{j,n}$
- X.....Horizontal coordinate of a point form the text contour
- Y.....Vertical coordinate of a point form the text contour
- w_t .....Template width
- h_t.....Template height
- w_i .....Image width
- h_i .....Image height
- A_x .....Proportional magnitude of an image width
- A_y .....Proportional magnitude of an image height
- M_x.....Horizontal margin distance between two areas
- M_y .....Vertical margin distance between two areas
- x_0.....Initial value of a variable x
- $d\left(Q_{end,i}, Q_{start,i+1}\right)$Distance between *end* point of a letter *i* and *start* point of the next letter, *i+1*
- $X_{end,i}$.....Horizontal coordinate of the *end* point of letter *i*
- w_l.....Approximation of a word's letter width
- δ .....link distance ratio with respect to w_l

The general notation that we use in this manuscript is:

Italic: Denotes scalar variable or function
Bold: Denotes a vector ($\mathbf{x} = (x_1, x_2, \dots, x_n)$)
CAPITAL BOLD: Denotes operator

LIST OF TABLES

1 INTRODUCTION

This chapter introduces the handwriting context and presents our motivation to work in this domain. It also defines our research statement and summarizes our key contributions.

1.1 CONTEXT AND MOTIVATION: HANDWRITING AS MARKER OF COGNITIVE ABILITIES

Among the several processing levels involved in the writing process, most researchers have centred their studies on spelling and motor production. Firstly, most researchers ignored the interaction between these two processes and postulated parallel processing of the different components: central processing, related to word spelling, and peripheral processing, related to handwriting production [9]. Some authors even stated that there was no interaction between the spelling module and graphic production of words [11].

Nevertheless, recent research outcomes support the idea that written production is modulated by spelling processes [12], provide evidence for this functional interaction between spelling and motor production [14].

Recent research suggests that letter production does not only depend on its shape but is also influenced by orthographical encoding [10]. Orthographic regularity and other spelling characteristics have been proved to have an influence on word production [13].

The interaction between these two processes becomes more relevant during the writing skills acquisition because the motor gestures are not yet automatized and spelling turns out into a more challenging process.

a) Learning to write

To learn how to write, children must acquire detailed orthographic representations to know words' spelling and they also have to learn to execute the graphomotor gestures to produce handwriting. At first, the movements to produce letters are slow and cognitively very demanding but, with practice, they become faster and more automatic [4]. Research suggests that when this automaticity increases, writing starts to be regulated by orthographic knowledge; meaning that it is during writing skills acquisition that the interaction between spelling and motor processing builds up. This will make possible for children to process orthographic knowledge in parallel to movement production. This fact can be detected in the amount of time that children need before starting writing a word to prepare their movements and it also influences the writing duration and fluency.

Even the interaction between central and peripheral processing has been supported by many studies, only few of them analyse these two components simultaneously. Moreover, research about the graphomotor side of writing is not abundant. New research aims to understand graphic complexity of letters by analysing its graphic characteristics, oppositely to previous research which mainly considered handwriting global characteristics [2]. This enabled the determination of the influence of graphic characteristics on handwriting production.

The importance of the interaction between spelling and graphomotor abilities is then supported by research; nevertheless, it is still not clear to what extent it takes place. On the one hand, it has

been demonstrated that spelling determines the variability on graphic production. But, although the influence of children's graphomotor abilities on spelling has been shown, only few researchers have investigated it. Authors give now evidence that some particular aspects of graphic production could lead to difficulties for children during writing acquisition.

b) Dyslexia

It's not uncommon for children to have some kind of trouble with written expression. One of the most common causes is a learning disability called dyslexia, which is known as a reading disability but it also impacts writing ability. Children with dyslexia systematically struggle to form letters, to put ideas into words, to spell correctly or in reading.

There are many areas affected by dyslexia, among which dysgraphia is the responsible of making the physical act of writing difficult. In consequence, handwriting can be poor or illegible.

Studies have shown that students with dyslexia have trouble with the pre-writing stage: Because they're often poor spellers, writing becomes a real challenge for them. Their handwriting may be slow, and they may have a hard time getting their thoughts down on paper.

The fact that motor abilities of children with dyslexia have been found to be poorer than those of typically developing children, has been object of many investigations. Dyslexia can make it hard to visualize how words should be encoded in letters. This condition also affects fine motor skills used in writing, drawing and tracing. Some authors have demonstrated that dyslexic children often perform below their typically developing peers in terms of the quality and size of texts they produce, as well as the legibility of handwritten letters [7].

The reason to study the interaction between spelling and production is to gain information on writing difficulties to help children during their writing skills acquisition. This knowledge can also contribute to a better understanding of cases of children with dyslexia. By examining results on this domain, it will be possible to determine if such children are more impacted by graphic complexity due to a potential graphomotor deficit (sensory-motor theories of dyslexia) or if, oppositely, their slowness is due to spelling deficits [8].

Moreover, if graphical and spelling difficulties of words are known, teachers and parents will be able to provide helpful aide to children, specially, to children with dyslexia, for which the learning process can be much more challenging.

1.2 KNOWN LINKS BETWEEN SPELLING AND MOTOR PRODUCTION PROCESSES

The interaction between spelling and motor production processes is a key concept in the analysis of writing skills acquisition and performance. Even though research is scarce on this field, authors have already provided some results. Most of them have centred their studies on the consequences of spelling difficulties into graphic word production. Less are the authors who have focused their efforts into the inverse relation: the impact of graphic complexity of words and letters into spelling correctness.

Among the authors who have considered the second approach, some have noticed that more precise and significant results can be obtained if handwriting is analysed at letter-level and not only in word-level. For example, some authors [10] worked with letter duration instead of total word duration and found out that letter production does not entirely depend on its shape; the

timing of motor production is also influenced by the way that we encode a word orthographically. In consequence, impact of orthographic regularity was modulated by the position of the irregularity within the word. Similarly, it has been demonstrated that other spelling characteristics like letter doubling also regulate the dynamics of word production [13].

Also, other authors [5] worked on the analysis by letters and revealed that sublexical processing takes places during all writing process. Because lexical processing is still operating when writing begins, the duration of the initial letters is often longer for long and orthographically irregular words. With that they concluded not only that sublexical central processing cascades into peripheral processing, but also that the position of the irregularity modulates the cascade.

It can be noticed then, that letter position also affects movement duration and fluency; for this reason, working at a letter-level can provide new interesting results on this domain. To obtain data at this level, different authors have followed different methodologies: First of all, some of them asked the children to lift the pen between word letters [5]. As mentioned, following this methodology, they were able to gain understanding on how the activation spreads from the central processing of spelling to letter production. However, this technique has de drawback that children handwriting will not be as fluent and natural as it would be out of their experiments; so obtained data can be considered to be less realistic. Secondly, other authors generated algorithms to segment the words which children wrote with continuous movement. Here, Kandel and Perret [4] can be pointed out, who used geometric (peaks and curvature maxima in the trajectory) and kinematic (tangential velocity minima) criteria for word segmentation. They also confirmed that letter position affected movement duration and fluency.

Research by C. Gosse and S. Carbonnelle [15] proposes for the first time an algorithm to quantify the graphic characteristics of words based on the letters composing them according to pen stroke trajectory. Such algorithm requires as an input word data segmented into letters.

From all these recent researches it can be appreciated the importance of working at a level-letter. For that, current segmentations algorithms are not yet able to provide realistic data and precise results and the accuracy of word segmentation into letters directly affects the precision of the final results on the writing domain.

1.3 RESEARCH STATEMENT AND CONTRIBUTION

Known the importance of measuring writing characteristics letter-by-letter and not at word level, it is convenient to provide researchers a methodology able to provide accurate segmented data for their studies.

Our contribution is related to the development of this methodology. We design an application that, given recorded data of children's handwriting that corresponds to whole words data, transforms it into segmented data, each segment being associated to one of the letters compounding a word. This way, the researchers working on psychology, handwriting learning and dyslexia can use the results that this app will provide them to get further results on their domains. The newest feature in our method consists in using a graph representation for the multiple options to segment a word. This graph is built in such a way that the length of a path in the graph measures two complementary costs. The first one reflects the similarity between each segmented letter and its predefined reference template, while the second one measures the distance between the end of each letter and the beginning of the next one. The fact of using a

graph representation permits us to solve the segmentation problem by means of a shortest path algorithm.

1.4 WORKING DATA

The data that is used as an input for the segmentation algorithm have been recorded during an experiment designed and conducted by psychological researchers at UCL [2]. In particular, children were given a tablet and were simply asked to write down the words they heard, at their usual speed. They used a digital tablet on top of which some white A4 paper sheets were attached and a digital inking pen (Wacom Intuos Pro Medium and Wacom Pinking Pen), to make sure that testing conditions were close to how children usually write.

The tablets record multiple parameters of children's handwriting. Among them, X and Y coordinates of the pen with respect to the tablet are recorded as a function of time.

Moreover, the same words dictated to children were also written by the PhD developer, researcher on this field in the same testing conditions; as well as all the letters of the alphabet, separately. The aim of recording this data is to provide a reference template for the segmentation of each letter.

1.5 WORKING HYPOTHESIS

Given the context of the experiments developed by the team of Van Reybroeck and C. Gosse, some assumptions can be done with respect to recorded data. In first place, it is important to mention that words written by children are known a priori, since they result from a dictation task. This fact differentiates the current work from existing algorithms that aim to decode unknown words (plate readers, OCR algorithms, scanning images into text…). Second, in our work, we assume that the beginning and the end of the words have been identified, e.g. based on [15].

1.6 ALGORITHM OVERVIEW

The data recorded from digital tablets during the experiments is transformed into a set of successive points described by its 2-dimension coordinates (X and Y). These data samples are grouped into subsets according to the different handwritten words they belong to.

First of all, we use this data to generate a digital image of the handwritten word. A distance map is then computed from this image. The same procedure is used with the provided reference letters.

A distance map is a greyscale image associated to binary image, where the value of each pixel corresponds to the minimum distance to a non-zero pixel in the original image. This image has smoother properties than the original word images and this fact has many advantages in the matching procedure that supports our segmentation method.

Two steps are considered to match individual letter templates with the word. The first is based on the Lucas Kanade optimization algorithm to warp the letter template so that its affine transform matches a part of the word. The idea behind this first stage is to modify the templates so that they are more similar to the letters in the word being analysed. The second considers normalized cross-correlation as a metric to translate each letter so as to jointly minimize the distance between consecutive letter templates and maximize the similarity between the letter

template and the local appearance of the written word. In practice, this joint optimization is formulated as a shortest-path computation problem in a graph connecting all possible displacements of consecutive letters.

1.7 ORGANIZATION OF THE THESIS

To present our work, this thesis is structured as follows:

Chapter 2 introduces preliminary concepts. Those include mathematical concepts, template matching techniques, and graph theory notions.

After that, Chapter 3 presents related works and existing methods used for handwriting recognition and digitalization. It also reviews optimisation techniques for template matching implementation.

Chapter 4 presents an overview of the proposed algorithm for word segmentation. It explains all the different steps included in the method and justifies the design choices that we have made. Extended validation and discussion of the key image processing steps of the algorithm (Lucas-Kanade individual letter template warping and graph-based joint alignment of the entire set of templates with the word) are presented in Chapter 5.

The method is then validated in Chapter 6. The criteria used to validate the results are also included in the same chapter. This chapter also includes a discussion on the results and the method weak points.

Finally, Chapter 7 presents the conclusions to the developed work and gives a review of possibilities for further work based on the weaknesses of the method and some new options that we came up with during the work development.

2 PRELIMINARIES

This chapter reviews the background material needed to develop the proposed methodology in this thesis. The chapter is divided in four sections: Section 2.1 defines some mathematical concepts to work with scalar functions; Section 2.2 presents image processing framework; Section 2.3 gives an overview of template matching techniques and Section 2.5 includes notions from graph-theory.

2.1 MATHEMATICAL CONCEPTS

In this work we often represent images as scalar functions of coordinate vectors. For this reason, in this chapter we present some mathematical concepts that become relevant to handle the image functions in our context as well as the particular notation that we use in this work.

2.1.1 Function concepts

Given a function $f(\boldsymbol{x})$, where $\boldsymbol{x}$ is a vector $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$, some concepts can be defined:

Gradient

Let $f: X \subseteq R^n \to R$ be a differentiable function, then the gradient of f at $\boldsymbol{x} \in X$ is defined by:

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \dots, \frac{\partial f}{\partial x_n}(\mathbf{x}) \right)$$

Hessian

Let $f: X \subseteq R^n \to R$ be a twice differentiable function, then the Hessian of f at $\boldsymbol{x} \in X$ is defined by:

$$\boldsymbol{H}\big(f(\mathbf{x})\big) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(\mathbf{x}) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2^2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_n \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\mathbf{x}) \end{pmatrix}$$

2.1.2 Taylor series expansion

Taylor series consist in the representation of a function as an infinite sum of terms calculated from values of the function's derivatives at a single point. A finite number of terms of this infinite series can be used to obtain the Taylor polynomial of a series to represent a function. The amount of terms used for this approximation of a function defines the degree of the polynomial and, consequently, the precision. The Taylor theorem gives approximation of the error of such estimation.

In particular:

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

First order Taylor expansion:

The first order Taylor expansion for f at the point $(\boldsymbol{x}+\boldsymbol{h})$, which is neighbouring $\boldsymbol{x}_0$ is:

$$f(\mathbf{x}_0+\boldsymbol{h}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)\cdot \boldsymbol{h} + O(\|\boldsymbol{h}\|^2)$$

Second order Taylor expansion:

The first order Taylor expansion for f at the point $(\boldsymbol{x}+\boldsymbol{h})$, which is neighbouring $\boldsymbol{x}_0$ is:

$$f(\mathbf{x}_0+\boldsymbol{h}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)\cdot h + \frac{1}{2}\boldsymbol{h}^T\cdot \boldsymbol{H}\big(f(\mathbf{x}_0)\big)\cdot \boldsymbol{h} + O(\|\boldsymbol{h}\|^3)$$

2.1.3 Optimization concepts

An optimization problem is a mathematical problem in which an objective function that has to be minimized or maximized is defined. This function and its variables are subject to some equality and inequality constraints.

General formulation:

Minimize or maximize e$(\boldsymbol{x})$ Subject to $g(\boldsymbol{x}) = 0$ Subject to $h(\boldsymbol{x}) \leq 0$	$e(\boldsymbol{x}) = f(\boldsymbol{x});\quad f: \mathbb{R}^n \to \mathbb{R}$ $g(\boldsymbol{x}) = (g_1, \dots, g_m): \mathbb{R}^n \to \mathbb{R}^m$ $h(\boldsymbol{x}) = (h_1, \dots, h_m): \mathbb{R}^n \to \mathbb{R}^p$	Objective function Equality constraints Inequality constraints
The set $\{\boldsymbol{x} \in \mathbb{R}^n \mid g(\boldsymbol{x}) = 0 \; and \; h(\boldsymbol{x}) \leq 0\}$ is the feasible region		

The solutions to the optimization function correspond to stationary points of the function (where the derivative of this function is null). These points are candidates of being minima or maxima of the function, and they can be characterized as:

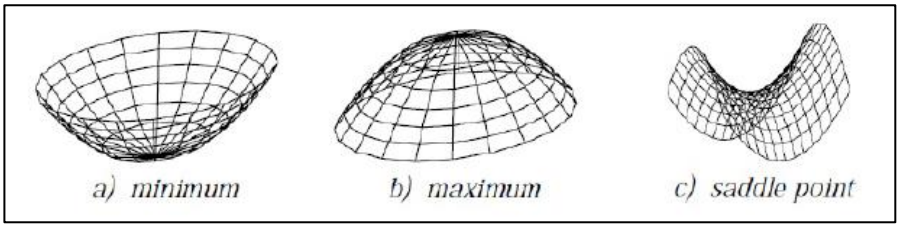


Figure 1: Graphical representation of different types of equilibrium points of a function

A subclass of the optimization problem is known as convex optimization problems. Its general formulation is:

Minimize $e(\boldsymbol{x})$ Subject to $g(\boldsymbol{x}) = A\boldsymbol{x} - b = 0$ Subject to $h(\boldsymbol{x}) \leq 0$	$e(\boldsymbol{x}) = f(\boldsymbol{x});\quad f: \mathbb{R}^n \to \mathbb{R}$ $g(\boldsymbol{x}) = (g_1, \dots, g_m): \mathbb{R}^n \to \mathbb{R}^m$ $h(\boldsymbol{x}) = (h_1, \dots, h_m): \mathbb{R}^n \to \mathbb{R}^p$	Objective function Equality constraints Inequality constraints
where f is a convex function, g is affine and h is convex.		

i) Gradient methods

The vector of variables that the optimization function depends on is optimized at each iteration. For that, there are multiple approaches, amongst which we find the gradient methods. In these approaches, the objective functions decreases (or increases) by varying the variables in the direction opposite to the function gradient at the current point.

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma \cdot \frac{\boldsymbol{\nabla} f(\boldsymbol{x}_i)}{\|\boldsymbol{\nabla} f(\boldsymbol{x})\|};$$

where γ is the step size.

The magnitude of the increase of the variables in the mentioned direction is given by the step size parameter. Multiple options are available to set this parameter:

a) Fixed step size

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma \cdot \frac{\boldsymbol{\nabla} f(\mathbf{x}_i)}{\|\boldsymbol{\nabla} f(\mathbf{x})\|};\ \gamma\ constant$$

b) Step proportional to the gradient norm

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma \cdot \boldsymbol{\nabla} f(\mathbf{x}_i);\ \gamma\ constant$$

c) Steepest descent with line search

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i \cdot \frac{\boldsymbol{\nabla} f(\mathbf{x}_i)}{\|\boldsymbol{\nabla} f(\mathbf{x})\|};\ \gamma_i = \underset{\gamma}{\operatorname{argmin}} f\left(\mathbf{x}_i - \gamma \cdot \frac{\boldsymbol{\nabla} f(\mathbf{x}_i)}{\|\boldsymbol{\nabla} f(\mathbf{x})\|}\right);$$

d) Steepest descent with quadratic approximation

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma \cdot \boldsymbol{\nabla} f(\mathbf{x}_i)\,; \quad \gamma = \frac{\boldsymbol{\nabla} f(\mathbf{x}_i)^T \cdot \boldsymbol{\nabla} f(\mathbf{x}_i)}{\boldsymbol{\nabla} f(\mathbf{x}_i)^T \cdot \mathbf{H}(f(\mathbf{x}_i)) \cdot \boldsymbol{\nabla} f(\mathbf{x}_i)}$$

If f is a quadratic function, this method coincides with the previous one.

Another well-known approach is the Conjugate gradient method.

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \gamma_i \cdot \tau_i$$

$$\tau_i = \begin{cases} -\boldsymbol{\nabla} f(\mathbf{x}_i)\ ; & if\ i = 1 \\ -\boldsymbol{\nabla} f(\mathbf{x}_i) + \beta_i \cdot \tau_{i-1}\ ; & if\ i > 1 \end{cases}$$

$$\beta_i = \frac{|\boldsymbol{\nabla} f(\mathbf{x}_i)|^2}{|\tau_i|^2}$$

or using quadratic approx.:

$$\begin{cases} \gamma_i = \underset{\gamma}{\operatorname{argmin}} f(\mathbf{x}_i + \gamma \cdot \tau_i); \\ \gamma_i = \dfrac{\boldsymbol{\nabla} f(\mathbf{x}_i)^T \cdot \boldsymbol{\nabla} f(\mathbf{x}_i)}{\boldsymbol{\nabla} f(\mathbf{x}_i)^T \cdot \boldsymbol{H}(f(\mathbf{x}_i)) \cdot \boldsymbol{\nabla} f(\mathbf{x}_i)} \end{cases}$$

ii) Newton methods

Newton's method (also called Newton-Raphson method), is also applied on optimization problems in which the objective function is twice-differentiable.

While the steepest descent methods use only first derivatives to select the suitable direction, Newton's method uses first and second derivatives and, in general, performs better. Given an initial point, it constructs a quadratic approximation of the objective function that matches the

first and second derivative values for that same point. This approximated quadratic function is optimized (minimized or maximized) instead of the original objective function. The vector of variables that results from this minimization phase is the starting point for the new iteration, and the procedure is repeated until convergence is reached.

This method uses second order Taylor series but it does not use the gradient directions. It is an exact method for quadratic functions.

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \boldsymbol{H}\big(f(\mathbf{x}_i)\big)^{-1} \cdot \boldsymbol{\nabla} f(\mathbf{x}_i)$$

2.2 DIGITAL IMAGE PROCESSING

2.2.1 Digital images notation

Digital images are represented by scalar functions of coordinate vectors, such as $f(\mathbf{x}) = f(x_1, x_2)$. Images being the core of this work, we simplify this notation for the specific case of images, thus working with the 2-D coordinates (x, y). For image functions, we adopt the nomenclature $f(\mathbf{x}) = f(\mathrm{x}, \mathrm{y})$.

Mainly, we work with the source image function, named $i(\mathrm{x}, \mathrm{y})$, and the template image function, named $\mathrm{t}(\mathrm{x}, \mathrm{y})$. In these particular cases, we usually simplify its notation in mathematical expressions by $i(\mathrm{x}, \mathrm{y}) = i$ and $t(\mathrm{x}, \mathrm{y}) = t$, respectively.

Similarly, for translation over the 2-D coordinates of an image (x, y), we also adopt a specific notation, concretely (u, v); where u corresponds to the horizontal translation along the x-direction $(u = T_1)$, and v corresponds to the vertical translation along the y-direction of the image $(v = T_2)$.

Finally, this notation simplification is involved in the image gradient expressions. In particular, we compute the horizontal and vertical gradients of an image $i(x, y)$as $\frac{\partial i(x,y)}{\partial x}$ and $\frac{\partial i(x,y)}{\partial x}$, respectively. In this work, we refer to these partial derivatives as $i_x(x, y) = i_x = \frac{\partial i(x,y)}{\partial x}$ and $i_y(x, y) = i_y = \frac{\partial i(x,y)}{\partial y}$.

Note that no other scalar values are named after this same notation to avoid confusion.

2.2.2 Mathematical morphology

Mathematical morphology is a theoretical framework for the analysis and processing of shapes in images using sets. Initially, it was applied to binary images although its extensions make it possible to work also with greyscale images. Techniques on this framework are mainly built on two basic image processing operations: Dilation and erosion. The first one enlarges objects in an image while erosion shrinks them. Mathematical morphology basic operations are non-linear in nature, what distinguishes these techniques from traditional linear image processing. Moreover, it preserves objects shape for the most part, what converts mathematical morphology in a tool where size and shape of objects play an important role.

As mentioned, the basic operations among the theory is built are dilation and erosion. To perform these actions it is necessary to define a structuring element, which determines the width of this increase or decrease in size of the objects. Using more complex structuring elements,

more complex operators can be constructed, which can be used for many purposes, such as image segmentation, noise filtering, feature detection….

2.2.3 Distance Transform

A Distance Transform, also known are Distance Map or Distance Field, is a derived representation of a digital image. This derived representation is a greyscale image associated to the initial binary image, with the same dimensions, in which the new value of each pixel of the background corresponds to the minimum distance to an obstacle pixel in the original image.

Different approaches to compute distance between two pixels can be used, as a result of considering images as discrete grids. In consequence, different results can be obtained by this technique. Take two pixels of an image, $\boldsymbol{q}_1 = (x_1, y_1)$ and $\boldsymbol{q}_2 = (x_2, y_2)$. Note that we use images nomenclature (Section 2.2.1).

Common metrics are:

- Euclidean distance: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
- City block distance or Manhattan distance $d = |x_1 - x_2| + |y_1 - y_2|$
- Chessboard distance $\mathrm{d} = \max\{|x_1 - x_2|, |y_1 - y_2|\}$

Even once the metric has been chosen, there are many ways of computing the distance transform of a binary image. One option is to perform multiple successive erosion operations until all the background pixels have been eroded. The value of each one of the pixels of this background is labelled with the number of erosions that perform to reach it. Selecting an appropriate structuring element to perform erosion is equivalent to selecting on of the above mentioned distance metrics.

Some different distance metrics have been used to compute the Distance map of the word image:

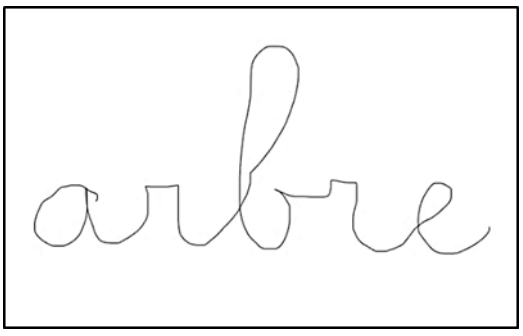


Figure 2: Digitalised image of the handwritten word "arbre"

Contour lines of similar grey-level values are plotted for better visualization of the result.

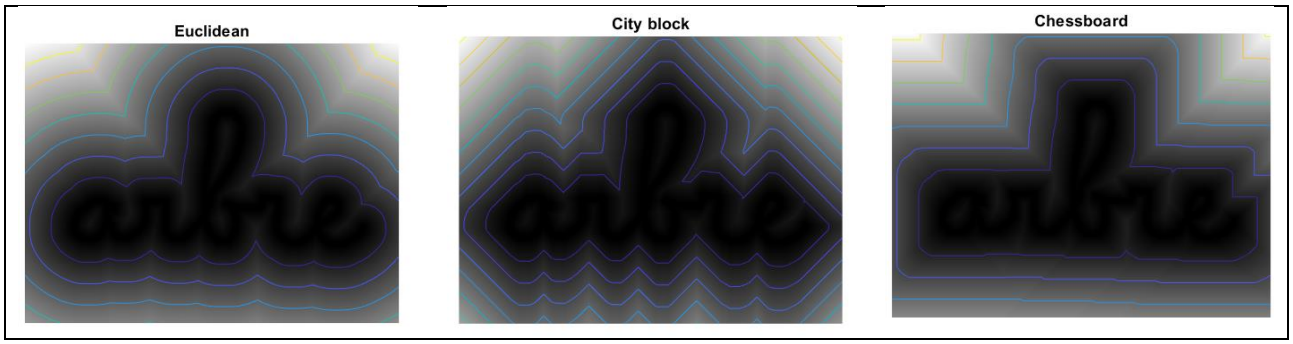


Figure 3: Comparison of the distance map of a word image created with different distance metrics

As a consequence of its construction, this technique converts the image function in a new and smoother function because the value of each image pixel is close to the value of the pixels in its neighbourhood. This fact has a relevant importance in this work, leading to smooth results in derivation and optimization too. More details of the DM building and justification are given in Chapter 5.

2.2.4 Affine transform in digital images

a) Homogeneous coordinates

Homogeneous coordinates are coordinates used in projective geometry. This is a technique widely used in computer image processing to perform geometric transformations in a simple way by multiplying matrices that represent transformations. It consists in the addition of one dimension to the working space to be able to operate in the mentioned way. Digital images are 2D elements on which different geometric transformations can be applied. Working with homogenous coordinates on digital images brings then to 3-dimension coordinates:

$$\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix} \Rightarrow \boldsymbol{x}_h = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Basic geometric transformations

Basic transformations operations in 2D images are:

- Translation: This operation moves a point into a new location and can be defined using homogeneous coordinates as: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_1 \\ 0 & 1 & T_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.
- Scaling: This operation changes the size of an object preserving proportionality. It can be defined as $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} K_1 & 0 & 0 \\ 0 & K_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.
- Rotation: This operation rotates the points in the images by a given angle θ with respect to the origin (The centre of rotation can be changed). It can be defined as: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.
- Shearing: This operation changes shape of objects. It can be applied to both of the images axis. Therefore, it can be defined:

- Shearing along x-axis: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & S_1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.
- Shearing along y-axis: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ S_2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.

Naturally, these two transformations can be combined and both axes can be deformed simultaneously.

b) General transformation matrix

The basic transformations can be combined to modify images. The matrices that describe the operations to be applied have to be multiplied. It is important to have in mind the order of the matrix multiplications, being the same order in which such operations affect the image. Combining some of the basic available transforms, some special cases are obtained:

- Rigid transformations: Involves only translation and rotation, thus it uses 3 parameters. It preserves angles and lengths.
- Similarity transformations: Involves rotations, translation and scaling, thus it uses 4 parameters. It preserves angles but not lengths.
- Affine transformations: Involves translation, rotations, scaling and shear, thus it uses 6 parameters. It preserves parallelism of lines but not length and angles.
- Projective transformations: Involves all basic operations and is defined by 8 parameters.

Modifying the geometry of an object in discrete space is more complex than in the continuous space. In the discrete domain, modifying the geometry consists in mapping values $\mathbb{Z}$ to $\mathbb{Z}$, or $\mathbb{N}$ to $\mathbb{N}$. When applying continuous transformations such as rotation, the obtained results may not belong to the discrete space. This fact is the reason why it is necessary to correct these results. Some examples of different approaches to face this are:

- Nearest Neighbour: Each resulting point is associated to the closest grid point. To do this, it is only necessary to round the resulting coordinates.
- Bilinear Transformation: Consists in a real bi dimensional interpolation.
- Triangular interpolation: Consists in an interpolation of three non-collinear points.
- Gauss's Pivot: It solves affine registration based in landmarks.

c) Affine transformation matrix

In this work, image transformations are limited to be affine transformations. The general affine transformation can be defined with only six parameters. Indeed, if we list the seven Degrees of Freedom (DOF) we have:

- θ: rotation angle.
- T_1: Horizontal component of the translation vector.
- T_2: Vertical component of the translation vector.
- K_1: Horizontal component of the scaling vector.
- K_2: Vertical component of the scaling vector.
- S_1: Horizontal component of the shearing vector.
- S_2: Vertical component of the shearing vector.

This description can be reduced to 6 independent parameters due the fact that the y component of shearing s_2 is a linear combination of the others.

The matrix that represents this transformation is:

$$\boldsymbol{T} = \begin{bmatrix} a_{11} & a_{12} & T_1 \\ a_{21} & a_{22} & T_2 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\begin{bmatrix} T_1 \\ T_2 \end{bmatrix}$ is the translation vector and $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ is the matrix describing the combination of rotation, scaling and shear.

The importance of affine transform for this work is linked to the use of template matching algorithms; concretely, in the fact that the object in the template may not be in the same scale, rotation and shear than the object in the image that wants to be identified. Therefore, it is convenient to transform it to make both images to be compared more alike and optimize template matching results.

2.3 TEMPLATE MATCHING TECHNIQUES

A general problem in image processing is to find the locality of an object in an image, when the template of this object is known but the scale and rotation of it in the image is unknown. The techniques in template matching are used in many fields, as face recognition, video compressing, patter recognition or signal processing.

Template matching is, therefore, a machine vision method used to identify components in a figure that match a given (set of) template(s). The general process of this method is to iteratively assess the matching between different patches of the image with the same size as the template and the template itself with the final objective to find Image window that is more similar to the template. The position of the Image Window that leads to the best matching assessment is assumed to be the position wherever the object (defined by the template) is located inside the image. To compare the Image patch and the template at each possible position, it is used a similarity measure method, such as SSD or NCC and the results are stored in a matrix. This results matrix determines how similar the template is to each particular area in the source image.

There are multiple techniques for template matching that are currently being used. Among these techniques, we must highlight Image Correlation Matching, which uses as similarity measure Cross-Correlation, Normalized Cross-Correlation, or Fast-Normalized Cross-Correlation among others. Another essential technique for template Matching is based on sum of squared differences. These techniques are presented in detail below.

2.3.1 Template matching Terminology

In template matching there are two main elements required:

Source Image (I): Picture in which an object has to be identified.

Template image (T): Image that contains the object that wants to be identified in the source image.

There is other vocabulary associated to template matching execution,

Image Window (IW): The source image and the template image do not have the same dimensions. Therefore, for each template position being assessed, the comparison is done

between the whole template image and a region of the source image below it (with the same dimensions than the template image), which is denominated Image Window or Image Patch.

Analysis Area (AA): If some a priori information is known about the position of the object of interest in the source image, the template matching methodology can be applied to a limited region of the source image instead of the whole source image to reduce computation cost. The region of the source image that is used for template matching is called Analysis Area.

Search Window (SW): The set of pixels contained in the Analysis Area where the reference pixel can be placed to assess similarity between the template and the current IW constitute the Search Window. The Search window is always smaller than the Analysis Area because it is created by subtracting from it the set of pixels where template can't be placed because it does not entirely fit. Figure 4 clarifies this distinction.

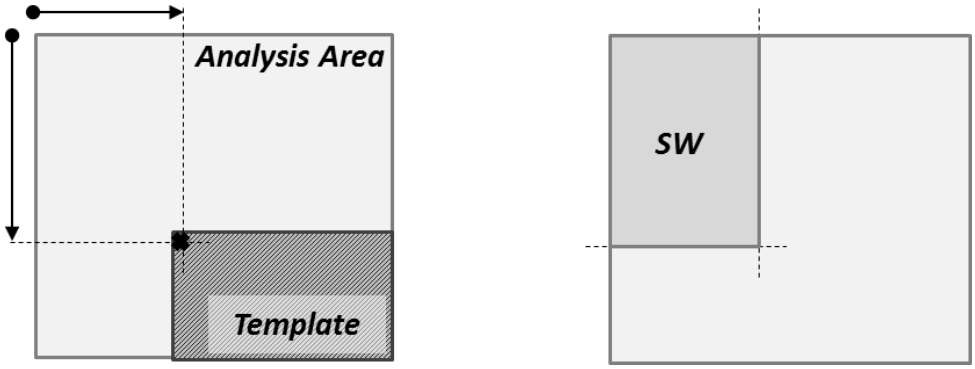


Figure 4: Analysis Area and Search Window graphical representation

2.3.2 Squared Sum of Differences (SSD)

The Sum of Squared Differences (SSD) is a mathematical expression used in many fields, among which we have template matching. SSD is a measure of discrepancy between two elements. It is calculated as the summation of the squares of the variation between the two series of values. SSD general formulation is:

$$\iint (f(\boldsymbol{x}) - g(\boldsymbol{x}))^2$$

where f and g correspond to two different functions.

This formula can be applied to the images domain, as a pixel by pixel intensity difference between the two compared images.

For example, if only translation is allowed between the template and the image, assuming that the object that has to be identified can be found in the same scale, rotation and shear in the image and in the template, SSD for template matching from previous expression becomes:

$$SSD(u,v) = \sum_{x=1}^{N}\sum_{y=1}^{M}[i(x,y) - t(x+u, y+v)]^2$$

where i and t correspond to the functions describing the image and the template, respectively; N and M define the height and width of the image patch, of the same dimensions of the template

(number of rows and number of columns); and (u, v) are the variables that define the shift component along the image x-direction and y-direction, respectively where the template is located on the image.

For a more general applicability, other degrees of freedom are allowed to find the template in the image. Therefore, warping of the image is included in the expression.

$$SSD(x, y; \boldsymbol{p}^*) = \sum_{x=1}^{N} \sum_{y=1}^{M} \left[i\left(\boldsymbol{W}(x, y; \boldsymbol{p}^*)\right) - t(x, y) \right]^2$$

The warped image is described by $i\left(\boldsymbol{W}(x, y; \boldsymbol{p}^*)\right)$, where $\boldsymbol{p}^*$ is the parameter vector evaluated at a particular point.

Note that in the previous expression, the two involved images (template and source image) are used in its raw description. This means that no mean-subtraction is indicated in the formula. However, this same expression can be applied under both, local or global mean-subtraction strategies. This is a practical issue on implementation, and the formula is slightly modified, by replacing the image notation $i(x, y)$ for $\tilde{\imath}_{\boldsymbol{u},\boldsymbol{v}}(x, y)$ if local mean-subtraction is used; or $\tilde{\imath}(x, y)$ if it is done globally; and also replacing the template notation $t(x, y)$ for $\tilde{t}(x, y)$, in both cases.

2.3.3 Correlation techniques

Another method to compute the similarity between the template and the image for each template position is correlation. This approach only allows translational degrees of freedom between the two images, being (u, v) the shifting of the template along the Search window to test all possible template positions on the Analysis Area (i.e., to place the template in all pixels of the Search Window).

a) Cross-correlation

Correlation assesses the degree of similarity between two elements. If these two elements are independent, it is called cross-correlation (CC).

Working with images, Cross-correlation is defined as:

$$CC(u, v) = \sum_{x,y} [i(x, y) \cdot t(x - u, y - v)]$$

where $i(x, y)$ is the image and the sum is over x, y, under the window containing the template t positioned at (u, v).

b) Normalized Cross Correlation

Cross correlation is sensitive to bright variations and other distortions. By normalizing it, some of its disadvantages are highly overcome. The expression of the normalized cross-correlation (NCC) between a function $i(x, y)$ that represents an image and the template $t(x, y)$ is:

$$NCC(u, v) = \frac{\sum_{x,y}[i(x, y) \cdot t(x - u, y - v)]}{\left(\sum_{x,y}[i(x, y)]^2 \sum_{x,y}[t(x - u, y - v)]^2\right)^{1/2}}$$

However, a usual practice to apply this method is to subtract the mean of the image functions. In consequence, the previous expression can be rewritten as:

$$NCC(u,v) = \frac{\sum_{x,y}\left[i(x,y) - \bar{\imath}_{u,v}\right]\left[t(x-u,y-v) - \bar{t}\right]}{\left(\sum_{x,y}\left[i(x,y) - \bar{\imath}_{u,v}\right]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2\right)^{1/2}}$$

where $\bar{\imath}_{u,v}$ is the mean of the value of the pixels of the area of the image $i(x,y)$ below the template, when the template is positioned at (u,v). ; and $\bar{t}$ is the mean of the template pixels value.

The mean of the source image and the template are subtracted to centre both images. The template image is constant for all possible Image windows. In consequence, it can be also defined the mean-subtracted template $\hat{t}(x,y)$:

$$\hat{t}(x,y) = t(x,y) - \bar{t}$$

This leads to the rewritten expression:

$$NCC(u,v) = \frac{\sum_{x,y}\left[i(x,y) - \bar{\imath}_{u,v}\right]\hat{t}(x,y)}{\left(\sum_{x,y}\left[i(x,y) - \bar{\imath}_{u,v}\right]^2 \sum_{x,y}\hat{t}(x,y)^2\right)^{1/2}}$$

For a more compact writing of the NCC expression where mean-subtraction is applied for the template and the image window, we use the term $\tilde{\imath}_{\boldsymbol{u,v}}$ to indicate local mean-subtraction of the image patch. Thus we finally obtain:

$$NCC(u,v) = \frac{\sum_{x,y}\tilde{\imath}_{\boldsymbol{u,v}}(x,y)\cdot\hat{t}(x,y)}{\left(\sum_{x,y}\tilde{\imath}_{\boldsymbol{u,v}}(x,y)^2 \cdot \sum_{x,y}\hat{t}(x,y)^2\right)^{1/2}}$$

c) <u>Fast Cross-Correlation</u>

To better understand the Fast Cross-Correlation (FCC) algorithm, it is necessary to briefly review the Fourier Transform and the Convolution theorem.

The Convolution theorem for 2-dimensional space states that the Fourier transform of the convolution of two functions is the product of their individual Fourier transforms.

$$g(x,y) * h(x,y) \leftrightarrow G(x',y') \cdot H(x',y')$$

The main advantage of this theorem is applicable to the image filtering domain. Given the fact that Cross Correlation between images is a filtering operation, this is a matter of interest of this work. To continue the explanation, it is relevant to notice the similitude between image convolution and correlation (where each image can be understood as a function of the coordinates vector):

$$Conv(g,h) = g * h = \int_{-\infty}^{\infty} g(\tau)h(t-\tau)d\tau = h * g$$

$$Corr(g,h) = \int_{-\infty}^{\infty} g(\tau+t)h(t)d\tau$$

These two operations are performed in a same way but in convolution the filter is reflected in the X and Y axes. This means that one can easily switch between both expressions by slightly changing the filter according to it.

From all this statement, it can be concluded that the Correlation (linear filter operation) between two images can be efficiently carried out by simple multiplications in the Fourier domain. For this, the necessary steps are:

1. Compute FT of both images (the source image and the template)
2. Multiply the two FT's
3. Compute the inverse FT of the result to bring the result back to the space domain

This methodology to carry out Cross Correlation is known as Fast Cross Correlation (FCC). Its main drawback is that it is suitable for Cross Correlation efficient computation but not applicable to its normalized version.

The same concept of Fourier Transforms applied to images to compute correlation is desirable for Normalized Cross Correlation, but its conversion is not as direct. Given the advantages that this technique could bring, some authors are currently working on the development of techniques to reduce computational cost of Fast Normalized Cross Correlation.

2.3.4 Lucas Kanade method

The method widely known as "Lucas-Kanade method" is a computer vision differential technique developed by Bruce D. Lucas and Takeo Kanade for optical flow estimation. It makes some assumptions with respect to the relative movement of the contents in two image frames of nearby time instants, such as the hypothesis that displacement of these contents is essentially constant and small in a local neighbourhood of each pixel under consideration. Then, it solves the optical flow equations by means of the least squares criterion.

Nevertheless, this method can be extrapolated from optical flow computation to image registration [35]. The registration problem is the problem in which we look for the spatial relationship between two images such that it minimizes some measure of difference between both. Different measures of discrepancy can be used for that, like the L1 norm or L2 norm.

These authors presented a new image registration technique that makes use of the spatial intensity gradient of the images to find a good match using a type of Newton-Raphson iteration. Furthermore, this registration technique can be generalized to handle rotation, scaling and shearing

The main contribution of this algorithm is that it provides tools to perform template matching avoiding an exhaustive search all over the image. The following function has to be minimized

$$e(\boldsymbol{p}) = \sum_{x,y} \left[i\big(\boldsymbol{W}(x,y;\boldsymbol{p})\big) - t(x,y) \right]^2$$

with respect to **p**. This quadratic function represents the dissimilarity, measured by means of square sum of differences, between the template and the warped source image, over the Image Window (below the template).

It can be observed from the previous formula that this methodology is not only useful to find the template position but also its appropriate warping (scaling, rotation, shearing).

Image warping $i\big(\boldsymbol{W}(x,y;\boldsymbol{p})\big)$ for a particular value of the warping parameters vector ($\boldsymbol{p}^*$) is approximated using Taylor expansion series, such that:

$$i\big(\boldsymbol{W}([x,y];\,\boldsymbol{p}^*+\Delta\boldsymbol{p})\big)\approx i\big(\boldsymbol{W}([x,y];\,\boldsymbol{p}^*)\big)+\left.\frac{\partial i}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p}^*}\Delta\boldsymbol{p}=$$

$$i\big(\boldsymbol{W}([x,y];\,{p_1}^*,{p_2}^*,{p_3}^*,\dots)\big)+\left[\frac{\partial i}{\partial x}\frac{\partial \boldsymbol{W}_1}{\partial p_1}+\frac{\partial i}{\partial y}\frac{\partial \boldsymbol{W}_2}{\partial p_1}\right]\Delta p_1+\left[\frac{\partial i}{\partial x}\frac{\partial \boldsymbol{W}_1}{\partial p_2}+\frac{\partial i}{\partial y}\frac{\partial \boldsymbol{W}_2}{\partial p_2}\right]\Delta p_2+\cdots$$

being $\boldsymbol{W}=(\boldsymbol{W}_1,\boldsymbol{W}_2)$.

Writing the expression in its matrix form:

$$i\big(\boldsymbol{W}([x,y];\,\boldsymbol{p}^*+\Delta\boldsymbol{p})\big)\approx i\big(\boldsymbol{W}([x,y];\,{p_1}^*,{p_2}^*,{p_3}^*,\dots)\big)+\begin{bmatrix}\frac{\partial i}{\partial x} & \frac{\partial i}{\partial y}\end{bmatrix}\begin{bmatrix}\frac{\partial \boldsymbol{W}_1}{\partial p_1} & \frac{\partial \boldsymbol{W}_1}{\partial p_2} & \cdots\\ \frac{\partial \boldsymbol{W}_2}{\partial p_1} & \frac{\partial \boldsymbol{W}_2}{\partial p_2} & \cdots\end{bmatrix}\begin{bmatrix}\Delta p_1\\ \Delta p_2\\ \vdots\\ \Delta p_n\end{bmatrix}=$$

$$=i\big(\boldsymbol{W}([x,y];\,\boldsymbol{p}^*)\big)+\nabla i\frac{\partial \boldsymbol{W}}{\partial \boldsymbol{p}}\Delta\boldsymbol{p}$$

Based on these steps, the quadratic optimization function can be rewritten as:

$$e(\boldsymbol{p})\approx\sum_{x,y}\left[i\big(\boldsymbol{W}([x,y];\,\boldsymbol{p}^*)\big)+\nabla i\frac{\partial \boldsymbol{W}}{\partial \boldsymbol{p}}\Delta\boldsymbol{p}-t(\mathbf{x})\right]^2$$

This function has to be derivate and forced to be equal to zero to find the equilibrium points, which are candidates of being optima. From the derivative of the function, the vector $\Delta\boldsymbol{p}$ has to be isolated and used to update the current parameter vector. Because this method is based in Taylor series approximation, the process has to be iteratively repeated until convergence for an accurate solution. Note that the original function is not being optimized, instead, a new approximation of it is optimized (Section 2.1.3).

The manipulation of the derivatives of the objective function until isolating $\Delta\boldsymbol{p}$ leads to the final expression to update the parameters:

$$\Delta\boldsymbol{p}=\left[\sum_{x,y}\left(\nabla i\frac{\partial W}{\partial \boldsymbol{p}}\right)^T\left(\nabla i\frac{\partial W}{\partial \boldsymbol{p}}\right)\right]^{-1}\left[\sum_{x,y}\left(\nabla i\frac{\partial W}{\partial \boldsymbol{p}}\right)^T\big(\mathrm{t(x,y)}-i\big(\boldsymbol{W}([x,y];\,\boldsymbol{p}^*)\big)\big)\right]$$

$$\Delta\boldsymbol{p}=\boldsymbol{H}^{-1}\left[\sum_{x,y}\left(\nabla i\frac{\partial W}{\partial \boldsymbol{p}}\right)^T\big(\mathrm{t(x,y)}-i\big(\boldsymbol{W}([x,y];\,\boldsymbol{p}^*)\big)\big)\right]$$

The obtained vector $\Delta\boldsymbol{p}$ is used to iteratively update $\boldsymbol{p}$ in the following way: $\boldsymbol{p}\leftarrow\boldsymbol{p}+\Delta\boldsymbol{p}$. These steps ($\Delta\boldsymbol{p}$ computation from previous $\boldsymbol{p}$, and $\boldsymbol{p}$ update) are repeated until estimates of the parameters $\boldsymbol{p}$ converge [36].

To sum up, the Lucas Kanade algorithm is made of the following steps, starting with a particular initialization of the parameters vector $\boldsymbol{p}$ after each iteration:

STEP	LK Algorithm Iterative Steps
1	Warp the Image $i(x,y)$ with ($\boldsymbol{W}([x,y];\ \boldsymbol{p})$ to compute $i\big((\boldsymbol{W}([x,y];\ \boldsymbol{p})\big)$
2	Compute the error image for the current Image warping as $e(\boldsymbol{p}) = t(x,y) - i\big((\boldsymbol{W}([x,y];\ \boldsymbol{p})\big)$
3	Warp the gradient ∇i with ($\boldsymbol{W}([x,y];\ \boldsymbol{p})$
4	Evaluate the Jacobian $\boldsymbol{J} = \frac{\partial W}{\partial \boldsymbol{p}}$ at current ($[x,y];\ \boldsymbol{p}$)
5	Compute steepest descent images $\nabla i \cdot \frac{\partial W}{\partial \boldsymbol{p}}(x,y)$
6	Compute the Hessian matrix as $\boldsymbol{H} = \sum_{x,y}\left[\nabla i \cdot \frac{\partial W}{\partial \boldsymbol{p}}(x,y)\right]^T \cdot \left[\nabla i \cdot \frac{\partial W}{\partial \boldsymbol{p}}(x,y)\right]$; and invert it to obtain $\boldsymbol{H}^{-1}$
7	Compute $\sum_{x,y}\left[\nabla i \cdot \frac{\partial \boldsymbol{W}}{\partial \boldsymbol{p}}(x,y)\right]^T \cdot e(x,y)$
8	Compute $\Delta \boldsymbol{p} = \boldsymbol{H}^{-1} \cdot \sum_{x,y}\left[\nabla i \cdot \frac{\partial W}{\partial \boldsymbol{p}}(x,y)\right]^T \cdot e(x,y)$
9	Update the parameters $\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p}$
10	Go to step 1 (iterate until convergence $\|\Delta \boldsymbol{p}\| \leq \epsilon$)

Table 1: LK optimization algorithm iterative steps summary

a) <u>Affine transformation application</u>

The affine warping can be represented by 6 parameters $\boldsymbol{p} = (p_1, p_2, p_3, p_4, p_5, p_6)$. The general concept of an affine transformation has been previously described in this work (Section 2.2.4). At this point, this kind of transform is applied in the 2-dimensional space, where images belong. The warping transform applied to images can be expressed using the 6 parameters as in a formula easy to manipulate:

$$\boldsymbol{W}([x,y];\ \boldsymbol{p}) = \begin{pmatrix} 1+p_3 & p_5 & p_1 \\ p_6 & 1+p_4 & p_2 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Among all the transformation parameters, (p_1, p_2) stand for the translational degrees of freedom (equivalent to $[T_1\ T_2]$); (p_3, p_4) stand for the scaling along both axis; and finally, the combination of the four parameters (p_3, p_4, p_5, p_6) is used to control shearing and rotation of the Image.

This can be better appreciated if the previous matrix warping expression is developed as a set of equations that express the new position of each pixel of the original image. The resulting equations are:

$$\begin{cases} W_1 = x + p_3 x + p_5 y + p_1 \\ W_2 = p_6 x + y + p_4 y + p_2 \end{cases}$$

The optimization function expressed in terms of the affine transformation becomes:

$$e(\boldsymbol{p}) = \sum_{x,y}\left[i\big(\boldsymbol{W}(x,y;\boldsymbol{p})\big) - t(x,y)\right]^2$$

The steps describe previously (Table 1) are applied for the general affine transformation. The terms that need to be computed to apply the 10-step algorithm and update the parameters vector in each iteration applied to the affine transform turn into:

i) Image gradient:

$$\nabla i = \left[\frac{\partial i}{\partial x}\ \frac{\partial i}{\partial y}\right] = [I_x\ I_y]$$

ii) Jacobian of the warping :

$$\boldsymbol{J} = \frac{\partial W}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial W_1}{\partial p_1} & \frac{\partial W_1}{\partial p_2} & \cdots & \frac{\partial W_1}{\partial p_6} \\ \frac{\partial W_2}{\partial p_1} & \frac{\partial W_2}{\partial p_2} & \cdots & \frac{\partial W_2}{\partial p_6} \end{bmatrix} = \frac{\partial \begin{bmatrix} x + p_3 x + p_5 y + p_1 \\ p_6 x + y + p_4 y + p_2 \end{bmatrix}}{\partial \boldsymbol{p}} = \begin{bmatrix} 1 & 0 & x & 0 & y & 0 \\ 0 & 1 & 0 & y & 0 & x \end{bmatrix}$$

iii) Hessian matrix:

$$\boldsymbol{H} = \left[\sum_{x,y} \left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}}\right)^T \left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}}\right)\right] =$$

$$= \sum_{x,y} \left([i_x\ i_y]\begin{bmatrix} 1 & 0 & x & 0 & y & 0 \\ 0 & 1 & 0 & y & 0 & x \end{bmatrix}\right)^T \left([i_x\ i_y]\begin{bmatrix} 1 & 0 & x & 0 & y & 0 \\ 0 & 1 & 0 & y & 0 & x \end{bmatrix}\right) =$$

$$= \sum_{x,y} [i_x \quad i_y \quad i_x x \quad i_y y \quad i_x y \quad i_y x]^T [i_x \quad i_y \quad i_x x \quad i_y y \quad i_x y \quad i_y x]$$

where $i_x(x,y) = \frac{\partial i}{\partial x}$, $i_y(x,y) = \frac{\partial i}{\partial y}$ are the image gradient in x and y directions.

The Hessian results in a 6x6 matrix. Then, it is inverted and H^{-1} is obtained.

iv) Right-Side term:

$$\sum_{x,y} \left[\nabla i \cdot \frac{\partial W}{\partial \boldsymbol{p}}(x,y)\right]^T \cdot e(\boldsymbol{p}) =$$

$$= \sum_{x,y} [i_x \quad i_y \quad i_x x \quad i_y y \quad i_x y \quad i_y x]^T \cdot \left[t(x,y) - i\big(\boldsymbol{W}([x,y];\ \boldsymbol{p})\big)\right]$$

$$= \sum_{x,y} [i_x \quad i_y \quad i_x x \quad i_y y \quad i_x y \quad i_y x]^T \cdot \left[t(x,y) - i\big(\boldsymbol{W}([x,y];\ \boldsymbol{p})\big)\right]$$

Finally, $\Delta \boldsymbol{p}$ is computed with these terms and the corresponding formula in step 8 (Table 1).

The algorithm is used to iterate until convergence in the parameters is detected. In each iteration, error is computed between the template and the current warped image, defined by $\boldsymbol{p}$.

2.4 GRAPHS AND SHORTEST PATH ALGORITHMS

This work uses graph theory to find the best segmentation solution per each word based on individual letter identifications. Therefore, some concepts that are used in the work are clarified to give an overview and unify nomenclature.

2.4.1 Elements of graph theory and terminology

A graph is a diagram created with a set of vertices and links between vertices. The lines are called edges and the vertices, nodes. Each vertex represents one place or state of the system while the edges represent possible connections between the nodes.

Graphical dimensions of graph representations have no meaning. Graphs are a tool to represent a set of points and how are they joined up, without any metrical properties involved. However, nodes usually have a physical interpretation and the edges can have an associated weight that stands for the cost of connecting two nodes. For example, if nodes represent cities, edges value can be used to represent the distance between cities, or the time spent to travel from one to the other, etc.; if nodes represent possible machine configurations of a manufacturing company, edges will be indicating the economic cost of buying and selling machines to change between configurations.

Depending on its edges, a graph can be directed or undirected. If the edges between nodes are one-way connections, they are called directed, and so is the graph that contains them. The direction in which the edge has to be followed is usually marked with arrows. Oppositely, if the edges connecting nodes can be used in both ways, they are called undirected; and so is the graph that contains them. This difference can be appreciated in Figure 5 and Figure 6.

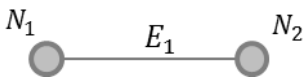


Figure 5: Undirected edge between nodes N1 and N2, with associated weight E1

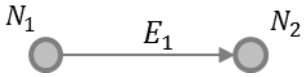


Figure 6: Directed edge from node N1 to node N2, and associated weight E1.

The image below shows an image of a graph example. This graph is constituted by 6 different nodes: $N = \{N_1, N_2, \dots, N_6\}$, and 9 edges $E = \{E_1, E_2, \dots, E_9\}$. In the presented example, all edges are undirected.

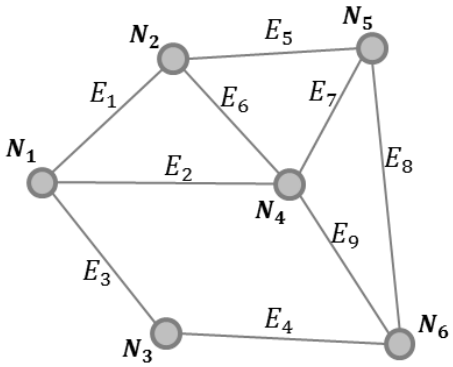


Figure 7: Example of an undirected graph constituted by 6 nodes and 9 edges

There are multiple properties and classifications for graphs. However, from all types of graphs, only trees are interesting for this work. A tree is a type of graph that is connected, does not contain any cycle and represents a hierarchical structure in a graphical form. To clarify this definition, it is convenient to define that a graph is said to be connected if there exists a path between every pair of nodes. Another remarkable concept is a graph cycle, which is a path in which a node is reachable from itself.

Some graph concepts have been introduced, such as nodes, edges, cycles, directed or undirected graphs… Nevertheless, complementary terminology of tree-types of graphs on this field is presented below:

2.4.2 Tree terminology

- Root**:** The top node in a tree is called Root node or Source node.
- Child**:** A node is a child of another node hierarchically superior, if it is directly connected to it when moving away from the Root.
- Parent**:** It is the converse notion of *child*. This means that a node is the parent to the nodes hierarchically inferior directly connected to it.
- Descendant**:** A descendant of a node is any reachable node by repeated proceeding from parent to child.
- Leaf: Nodes that have no children are called external nodes or leafs.
- Level**:** The level to which a node belongs is the number of connections between parent-child to reach it from the root node.
- Path**:** A path is the sequence of nodes and edges connecting a node with a descendant.

2.4.3 Shortest path algorithm

In graph theory, there are some graph-based algorithms that are commonly used in different fields of optimization. One of the well-known graph-based problems is the "Shortest path".

a) Shortest Path Problem

Given a connected graph (directed or undirected), the shortest path problem aims to identify the optimal, most effective path that connects a particular pair of nodes. The optimal path is selected as the one minimizing the sum of the associated weight of all edges that are included in the path sequence to connect the two nodes; so *optimal* can also mean *shortest, cheapest* or *fastest,* depending on the meaning of the edges associated weight.

Some different approaches have been developed to solve this problem. Some of these approaches are Brute Force -based. This means that they analyse all possible options to connect two nodes of a graph and select the global optimum. This strategy is computationally expensive and it can be not applicable in many cases.

There are many strategies to explore nodes' connectivity, such as Breath First Search algorithm (based on a nodes queue), Depth First Search algorithm (based on a nodes stack) or the Dijkstra algorithm, which we use in this work and is presented below. We also include an overview of the A-Star algorithm, being a generalization of the previous one.

Dijkstra

One of the most known greedy algorithms to solve this problem when edge weights are non-negative is the Dijkstra algorithm. Dijkstra algorithm solves the shortest path problem (P problem) in O(n2). It is an iterative algorithm that was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years. It has many variants but in its original formulation, it found the shortest path between two specified nodes of a graph [39].

The node at which we start is called the initial node. Because the current graph being analysed is a tree-shaped graph, this initial node corresponds to the root node. This root node is connected to its child nodes, which belong to the first level. Each one of these nodes is connected to some

nodes in the next level, which are its child nodes, etc. The edge that connects a node to each of its child nodes has a given weight or cost.

The algorithm also uses two variables to store the visited nodes, S, and the non-visited nodes, Q, which are updated after each iteration. Initially, only the root node belongs to the list of visited nodes, S, and all the other nodes of the tree belong to the que of nodes to be visited, Q. As the algorithm progresses, the set S will store those vertices to which the shortest path from the root node has already been found.

From these initial conditions, the Dijkstra algorithm works as follows:

1. Source vertex is selected. Label this node with cost 0 and insert it into the set S. Insert all other nodes in the set Q.
2. Label latest inserted node in S as current node. Consider each node in Q (not in S) connected to current node by an edge and compute the new distance to these nodes through current node. Use this data to label all nodes connected to current node with two data facts: The node from which they are reached (precedent node label) and the cost of reaching them by current analysed path (current distance label). This means that the cost label for the nodes connected to current node will be the distance to reach current node from the root node plus the weight of the edge from this current node new connected node.
   If one of these nodes is already labelled, compare the new computed distance to reach it with the stored one and maintain the smallest. The precedent node label will be the node that provided the smallest distance.
3. When all neighbours from current node have been analysed, current node is marked as visited and, therefore, stored in S. This means that it is also removed from the queue Q of non-visited nodes.
4. Select next node to be set as current node. For this, pick the node with minimum distance to be reached from the list of nodes Q. Repeat the above steps to check neighbours and mark the node as visited (steps 2 to 4).
5. If the destination node has been marked as visited or the smallest distance among the nodes in the list Q is infinity, algorithm is finished.

Once the algorithm is finished, S must contain all nodes in graph. Every node is marked with the precedent node from which it can be reached and the associated cost to reach it from root node.

The sink node can be taken and by backtracking precedent node labels, it is possible to build the shortest path from source node (root node) to objective node.

A-Star

A-star is a more generalized algorithm of the previously described Dijkstra algorithm.

In this case, every node connected to the current node being analysed, will not only be marked with the precedent node and the cost to reach it but also with a lower bound of the cost to reach the goal. Because the cost from these new nodes to the sink node is unknown, this label will be a lower bound based on some distance measure formula. For example, if nodes are cities and edges weights represent the distance to connect cities by car (using roads and highways), the lower bound could be the Euclidean distance between a city and the final destination city, drawing a straight line on the map, ignoring roads and highways. For sure, real distance by car will always be longer than this lower bound.

To select next node to be set as current node and proceed with its neighbour analysis, the whole distance will be taken into account: distance to reach a node plus distance to reach goal from that node.

As it was mention, the Dijkstra algorithm can be considered as a particular case of this A-star algorithm in which the lower bound that represents the distance form a node to the goal node is considered to be null.

2.5 CONCLUSION

In this chapter we have introduced some of the main concepts that we use in this algorithm development. The reason to include these concepts is to familiarize the reader with the specific background of out method and also to particularize these general concepts in the particular context of our work. Therefore, when applicable, the presented concepts have already been expressed in its version for images manipulation (discrete and digital framework) for a more direct interpretation.

3 RELATED WORK

This chapter presents the state of the art of handwriting digitalization techniques to review current background material that is useful for this thesis. It also introduces the state of the art in template matching and the available tools in Matlab to implement our method.

3.1 HANDWRITING DIGITALIZATION

3.1.1 Introduction

Handwriting recognition (HWR) is the computer process which receives and transforms handwritten inputs into digital information. These handwritten inputs can be images, photos, scanned documents, touch-screens and other. This is a domain of strong interest at the moment; actually, a recent study by Credence Research found that the handwriting recognition market is growing rapidly [21].

We can distinguish two main groups of digitalizing data techniques: On the one hand, there are engines that can be used to convert some information contained in paper sources into digital-shaped information. This can be easily achieved with cameras or scanners. However, after this step, the information is no easily accessible or searchable. For this reason, there's another set of digitalization techniques that also encodes the original information into machine symbols such as digital text. Thanks to this added features, it is possible to search the notes anytime, anywhere.

In this work, the second type of digitalization is analysed because our objective is to identify handwritten characters and match them with the correct alphabet letter that has been originally written. In consequence, during this work, handwriting recognition refers to two processes together: digitalizing and encoding paper information.

3.1.2 Online and offline handwriting recognition (HWR)

Handwriting recognition can take place in real time, while the user is writing information, by registering information of the pen being used and/or the writing on the paper. This is known as "on-line HWR" and it is currently one of the popular topics being developed. One example of on-line HWR is the digitalization of handmade notes that the user does on a touch-screen digital device such as a smartphone. This takes place simultaneously to the user-writing to save his notes as digital text instead of handwritten notes. On this first block, there are new technologies currently being developed; most of them are based on an optical pen, or grid-marked papers.

On the other hand, there is "off-line HWR", which is the application of HWR techniques for old-handwritten documents, which were created prior to the digitalization process. One example of application of HWR is for businesses to save years of handwritten records into electronic documents and records. If these documents were only scanned, they would be as images, difficult to modify or track. On this second block of HWR, there are many techniques available, which do not provide robust results yet; for example OCR algorithms (optical character recognition) or intelligent word recognition.

The main task included in Handwriting Recognition techniques is character recognition. Nevertheless, some extra functions are being added such as also handling format, lexical correction and grammatical adjustment.

a) Offline Handwriting Recognition

As mentioned above, the computer process to convert written or printed text into digital text is known as Offline Handwriting Recognition. Although this process is primarily focused on machine printed documents or sources (traffic signals, car plates, identity cards...), it can also be applied to recognise handwritten texts, which is the matter in this work.

If the input is a printing of a digital text, the results that can currently be achieved are close to perfect. The problem with this technique appears when less perfect inputs are treated, such as blurred images of printed words or handwritten texts, as different people have very different handwriting styles, especially in cursive handwriting. In this work, handwritten words are being analysed, and usually, bad quality- handwritten words. Therefore, these techniques are seen as bad candidates that are not able to provide robust results.

a.1) Optical Character recognition (OCR) methods

Optical Character Recognition Systems aim to transform images of documents into encoded and digital text. The input to these systems can be both, printed or handwriting text, but results robustness is currently only guaranteed for printed text while handwriting OCR techniques are still being developed for accurate results.

In the literature, the concepts of "Off-line Character Recognition" and "Optical Character Recognition" are commonly used indistinctly to refer to the computer process of encoding and digitalization of text in images, not taking into account if this text is handwritten or printed. However, some authors differentiate both concepts by considering that "Optical Character Recognition" is only a subclass of into OCR "Off-line Character Recognition" that faces the recognition of printed documents, parallel to the subclass of "Intelligent Character Recognition" which faces the off-line recognition of handwritten documents. In this work, the first nomenclature is used; what means that both, printed and handwritten character recognition processes, are referred under the same label of "Off-line Character Recognition". [17].

OCR algorithms involve different phases [16], [18], [19]:

I) Image acquisition

Although the input to this process are usually documents, they are provided as images of the original documents, such as scanned business documents, photos of traffic signals, recordings of digital cameras…

II) Pre-processing

A sequence of operation are applied to the input images with the objective of enhancing their quality and correct distortion to maximize probabilities of obtaining good results in text recognition. The main operations that are usually applied are:

- Noise removal: filtering or morphological operations
- Binarization
- Thinning
- Thresholding
- Edge detection
- Slant estimation and correction
- skew detection
- resizing

III) Segmentation

The segmentation phase is probably the main step in the process. It is done to separate text images into individual character images. As it can be intuitively deduced, segmentation of handwritten text is more complex than oriented text due to the increase of variability.

- External Segmentation: Based on general text layout, such as identifying paragraphs, sentences or words
- Internal Segmentation: Based on particular sentences or words segmentation.

IV) Feature extraction

Each of the elements obtained with the segmentation phase is analysed to retrieve the main information that it contains. This means that from each of these entities, a set of features is extracted such that the recognition rate can be maximized against the number of used elements. For this, each element is represented as a feature vector. Some methods applied for feature extraction are template matching, graph description, projection histograms, spline curve approximations, gradient features or Fourier descriptors.

V) Classification and Recognition

The classification stage is decisive with respect to handwritten text recognizing quality. This stage uses as an input the information of previous processing steps. Common classification approaches are:

- Statistical techniques: Methods concerned by statistical decision functions and a set of criteria that maximizes the likelihood of the observed patterns given a set of known models
    - Naïve Bayes theory- based techniques
    - Hidden Markov Models (HMM): Tries to predict the unknown sequence, hence it also tries to recognize the unknown character which is given as input
- Artificial Neural Networks (ANN): this technique has shown to be fast and reliable for classification aims. It is necessary to feed the net with a training dataset and the correct interpretations of the set. is able to find out the most probable character by
- generalization
- Template Matching: Recognizes characters by comparing image to images of the alphabet letters. It involves determining similarities between the templates and […]
- Multiple classifiers combination
    - Bayes classifier
    - Nearest neighbour classifier
    - Radial basis function
    - Support vector machines (use of Kernels)
    - Neural Networks
    - …
- Genetic algorithms: Motivated by the existence of multiple writing styles which, at the same time, are combined to generate new styles [20].

For word recognition it is common to use a lexicon. The set of character being recognized is attempted to be associated with an existing word according to choices in a lexicon.

a.2) Optical Character recognition (OCR) devices or software

There are nowadays many different OCR programs for digital printed document. Some examples are FreeOCR, Doxie Go (Portable scanner, only for printed documents), Tesseract, OCRAD, AnylineSDK….

b) Online Handwriting Recognition

Online Handwriting Recognition is the computer process of converting user handwriting notes into digitalized and encoded data in real time. Contrary to what happens in offline HWR, in this case input data are not complete images of text but a set of two dimensional coordinates of successive points as a function of time, that belong to the sampling in real time of the user's pen coordinate while writing in a surface. This writing surface can be a digital device or a specific paper notebook prepared for digitalizing objectives.

There are two main reasons for which these technologies are a current area of interest. First of all, handwriting is considered as an essential part of development, especially, for child's development. During the years, studies have revealed that handwriting helps children learn more quickly and retain information better and it also helps generating more ideas. This also applies in older people such as students during academic years in school or university, facing learning, memorizing and studying tasks, or even business people attending to conferences or meetings and trying to retain all important information that they receive. Because writing involves so many parts of brain, the mind is therefore more engaged in the process, leading to facilitated learning and greater working memory [27].

However, handwritten information is difficult to use, archive or search through. Therefore, the second reason of the growth of text digitalization techniques is to facilitate the management of handwritten notes. While it could be done by offline methods, results show that accuracy reached for handwritten texts is still low and that online text digitalization gives generally better results because of the characteristics of available data.

b.1) On-line character recognition methods

Because on-line handwriting recognition requires real time sampling of user's text, it is necessary to us some type of electronic device able to record this writing data. Usually, it involves the conversion of text at the same time that it is written in a special digitalizer support or PDA and the recording of the pen-tip movements and pen switching by some sensors usually integrated in the same pen. The recorded signal is converted into letter codes commonly through particular software integrated in digital device applications. By the moment, different strategies are currently being developed, and a summary of most popular techniques is presented below.

The common elements that techniques on these fields require are:

- a smart pen that the user will use to write
- a touch-sensitive surface, such as a tablet, smartphone or touch-screen laptop
- a software application that will convert recorded data by the two previous elements into digital text.

b.2) On-line character recognition devices and software

On-line character recognition techniques are currently being developed. In consequence, there is no uniformity over the methods strategies yet; software developers try new supports and approaches.

Smart Pens on special paper

Techniques centred in digitalizing text written on standard paper aim to maintain the natural gestures acquired in handwriting while providing the advantages of digital documents. In most of them, the paper has lots of embedded dots and marks that help recording and tracking the writing strokes of the user's pen.

Some examples in the state of the art on smart-pen based techniques are:

- Moleskine Smart Writing System: Dotted paper to track position of a Bluetooth-enabled pen. The digital pen can store about 1000 pages of notes in it portable memory. This data can be later transferred into a mobile app [24]

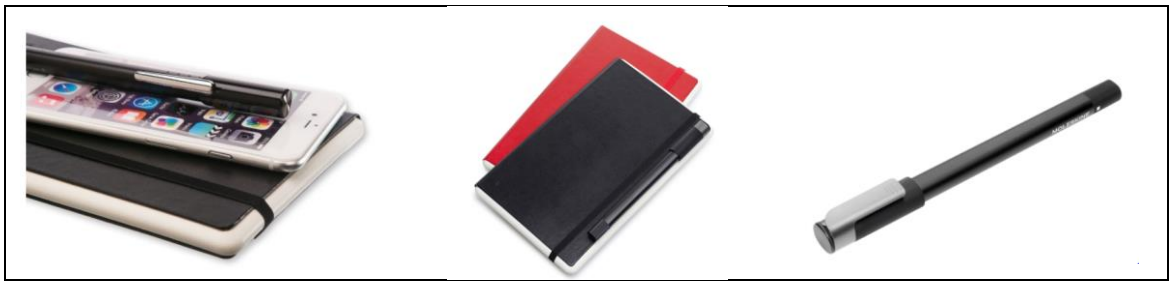


Figure 8: Moleskine Smart Writing System set

- Livescribe: Dotted paper and smart Bluetooth-enabled pen. The digital pen contains a set of elements, such as IR camera, ARM microprocessor, Bluetooth… Writing on a specific dotted paper, a mobile app is able to digitalize the handwritten notes. This pen has even a function to record audio notes through a built-in speaker. [23].

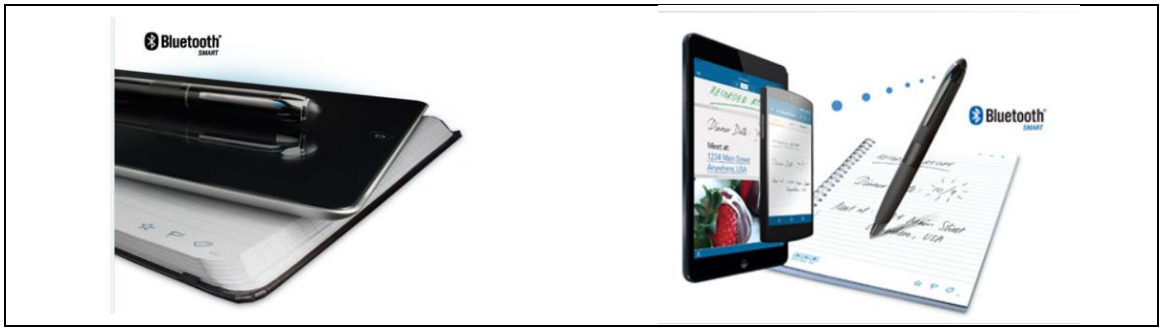


Figure 9: Livescribe Writing set

- LNeo SMartPen N2: Based on an optical pen and Ncode™ Technology. The optical digital pen contains a camera that captures more than 120 frames per second and calculates 256 steps of pressure. The pen sends camera recorded data to the mobile app, where it is converted into digital notes. Ncode™ is a combination of lines and symbols printed so small that they are difficult for the eye to perceive. They work as a pattern that serves to locate the pen tip with the mobile app. Software behind this set is based on MyScript's HWR technology [25].

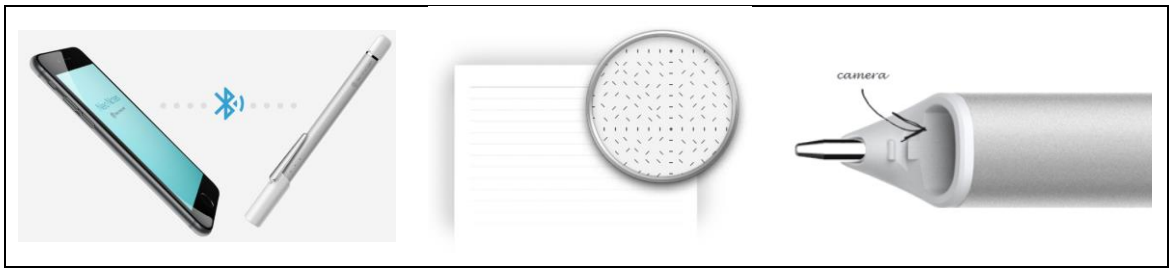


Figure 10: LNeo Smart Pen N2 writing set

- Wacom Bamboo Smart pads: Magnetic pen and pressure sensitive Pad. Wacom Smart Pads do not use dotted special papers or camera system, instead, a paper has to be clipped onto the pressure-sensitive screen and write normally with the pen, which hides a magnetic system

[26]. These particular SmartPads have been used to record the samples of the handwriting material that we analyse in this work.

Figure 11: Wacom Bamboo SmartPads: Folio solution (left) and Slate solution (right)

Touch-Screen devices

There is actually no need to buy a special set for digitalizing text purposes, although results may not be as accurate. Everybody that has a touch-screen digital device, such a smartphone, can have access to handwriting recognition and digitalization services by installing some applications.

- Smart note-taking: MyScript Smart Note

Another digitalizing tool based on MyScript's handwriting recognition technology. Recently launched, it is an advanced note-taking application that turns handwritten input into interactive, actionable digital text that.

- Evernote: Evernote Corporation digital application

Software designed to help the user to manage personal notes; being these notes texts, web pages, images or handwritten notes.

- OneNote: Microsoft's Free Software

Commercial Software designed to help the user in taking and managing notes, collecting information or sharing data with other users. The user can include handwritten notes among other information such as images, drawings, diagrams or even multimedia files. To improve performance on handwriting recognition and digitalization, it offers the option to be trained by the user, with its specific handwriting style (training ANN).

Other Software

- MyScript: MyScript offers technological solutions for multiple applications in handwriting recognition domain. Its basic concept is to represent data as a sequence of 2D points (x; y) ordered by time (t). This is denominated as digital ink, and it therefore refers to a dynamic process that takes into account where writing strokes start, where strokes end, and in which order they were drawn. Manufacturing companies can buy MyScript's technology to implement it in their digitalizing text products [27].

c) Contextualization of this work in the HWR state of the art

It is important to highlight that when operating in offline mode the input is the complete picture of characters that needs to be recognized in contrast to online character recognition, where data

is sampled while it is being written. Thus, on-line HWR, does not mean that data is processed simultaneously to the user's writing process; instead, it means that information of the writing process is available, besides than the final notes.

This fact introduces a HWR domain in which both strategies can be combined. Data samples from real-time handwriting can be used to digitally reconstruct handwriting material. Then, on-line and off-line HWR techniques can be applied to post process it. The first ones can be used to process the words in time-based mode, and the solution is complemented with application of the second group of techniques on the image word.

In the context of this work, data is collected with a pressure-sensitive digital screen (WACOM SmartPad); hence information of the writing process is available in data samples. However, we work most of the time on the reconstructed image of handwritten words (off-line HWR) but constraining the solution to the real-time information of the samples (on-line HWR); for example, by including writing fluency as a criterion for characters segmentation. On the other hand, our objective differs from standard HWR methods' objective. Off-line techniques for character identification and classification are not required because we have this information as an a priori assumption, though they can be used to complement and enforce the result.

3.2 TEMPLATE MATCHING OPTIMIZATION

Multiple template matching techniques have been presented in Chapter 2 as preliminaries to better understand the proposed methodology. These techniques are still being developed to gain efficiency, improve accuracy and reduce computational cost.

3.2.1 Correlation-based methods

The different options of applying correlation between images for template matching purposes have been also presented in the previous chapter. However, it is relevant to highlight that the implementation of these approaches is a current matter of work for researches.

Normalized Cross-Correlation has been proven to be efficient for template matching applications among others; but its considerable computational cost is seen as one of its main drawbacks. This has led some researchers to focus on the optimization of this algorithm.

Cross-correlation is commonly applied in the Fourier transform domain, achieving an increase of efficiency. Nevertheless, Normalized Cross-Correlation, which is preferred based on the obtained results, can only be computed in the spatial domain because it does not have a simple frequency domain expression. J.P. Lewis suggests a new method to efficiently implement Normalized Cross-Correlation by using precomputed tables containing the integral of the compared images over the search window [42].

3.3 MATLAB GRAPHICAL INTERFACE

The whole implementation of the segmentation methodology proposed in this work has been implemented using Matlab. The current version of the software that we use is Matlab R2017b student version, with its associate documentation.

This version of Matlab also offers an AppDesigner interface that has been used to develop the entire application user interface that embeds our method in a user-friendly interface.

To be able to use the Word Segmentation App, the user must be owner of a Matlab license and have had downloaded the software at the same or a later version of the one that has been used in this work. The distribution of our program is based on a Matlab App Package that the user can download and install.

3.4 CONCLUSION

In this chapter we have presented on overview of the state of the art in the main fields related to our method. With this research we aim to gain understanding on our working framework and to be able to use some developed techniques for similar purposes at the same time that we want to appreciate the differences on the methods that make other tools not applicable to our work.

With respect to the template matching techniques, we can notice that research is not scarce; reason why we propose to apply suitable implementations of the existing methods by particularizing them for our context. We also examine available frameworks that we may use for the method development; in particular, in Matlab.

We have mainly reviewed some of the current methods for similar objectives for handwriting digitalization and recognition. All these technologies have proven good results for printed texts and accurate handwriting, while they still present weaknesses when processing irregular handwritten texts. Moreover, the main function of these methods is to process the content of documents or notes to recognize the letters and words in it. Oppositely, the main assumption of this work is that the words being processed are known a priori, which is something completely new with respect to all presented techniques. Furthermore, the type of handwriting being analysed in this work is expected to be inaccurate and irregular because it belongs to children learning to write, being some of them children with dyslexia, whose handwriting performance is poorer.

These two facts added together make the presented techniques not suitable for the digitalization and segmentation tasks being faced in this project. It can be demonstrated that these techniques do not provide the necessary data for psychological research behind this work.

4 WORD SEGMENTATION METHOD OVERVIEW

This chapter aims to be an overview of our work's pipeline. In it, we present step-by-step all the phases that constitute the core of the method for word segmentation. We include a brief description of the different stages, each one with general information about the techniques that are applied, and the justification of why are they relevant for the method. Figure 12 presents a graphical scheme of the method's pipeline.

4.1 CONTEXT AND ASSUMPTIONS

4.1.1 Data

This method aims to determine the segmentation in letters of handwritten words; in particular for words written by children during the learning process of writing skills. Thus the input data is a digital representation of children's handwriting recorded with a pressure-sensitive screen of a digital Smart Pad.

Children data

Psychology Research team of M.V. Reybroeck and C.Gosse (*Faculty of Psychology and Educational Sciences, Université Catholique de Louvain, UCL*) [1] recorded data of children's handwriting through digital tablets. This data was stored in a .csv file. Data is basically a sequence of recorded points (also called packets by Wacom, the company that builds the SmartPad), each with the following information:

Feature	Description
index	Indexing of points starting at 0
PacketSerial	Indexing of points (=packets)
slice	Index of the stroke (i.e. sequences of consecutive points that are writing (or not writing)) to which the point belongs
writing	True or False depending if the pen touches the tablet or not
group	Index of the word to which the point belongs, starting at 0. (-1)index is used for strokes without writing between two groups
subject	Index of the subject that is writing. Index, PacketSerial, slice and group indexing restart (at 0 or 1) for each subject.
PacketTime	Time of arrival of the point/packet in milliseconds
X	Horizontal coordinate. 0 - 44703 points (1pt = 0.005mm).
Y	Vertical coordinate. 0 - 27939 points.
Z	Height coordinate. (-512) - (-1) points.
NormalPressure	Pressure of the pen on the tablet. If zero, the pen doesn't touch the tablet
Azimuth	Angle of the pen
Altitude	Angle of the pen

Table 2: Information associated per each WACOM sample of children's handwriting

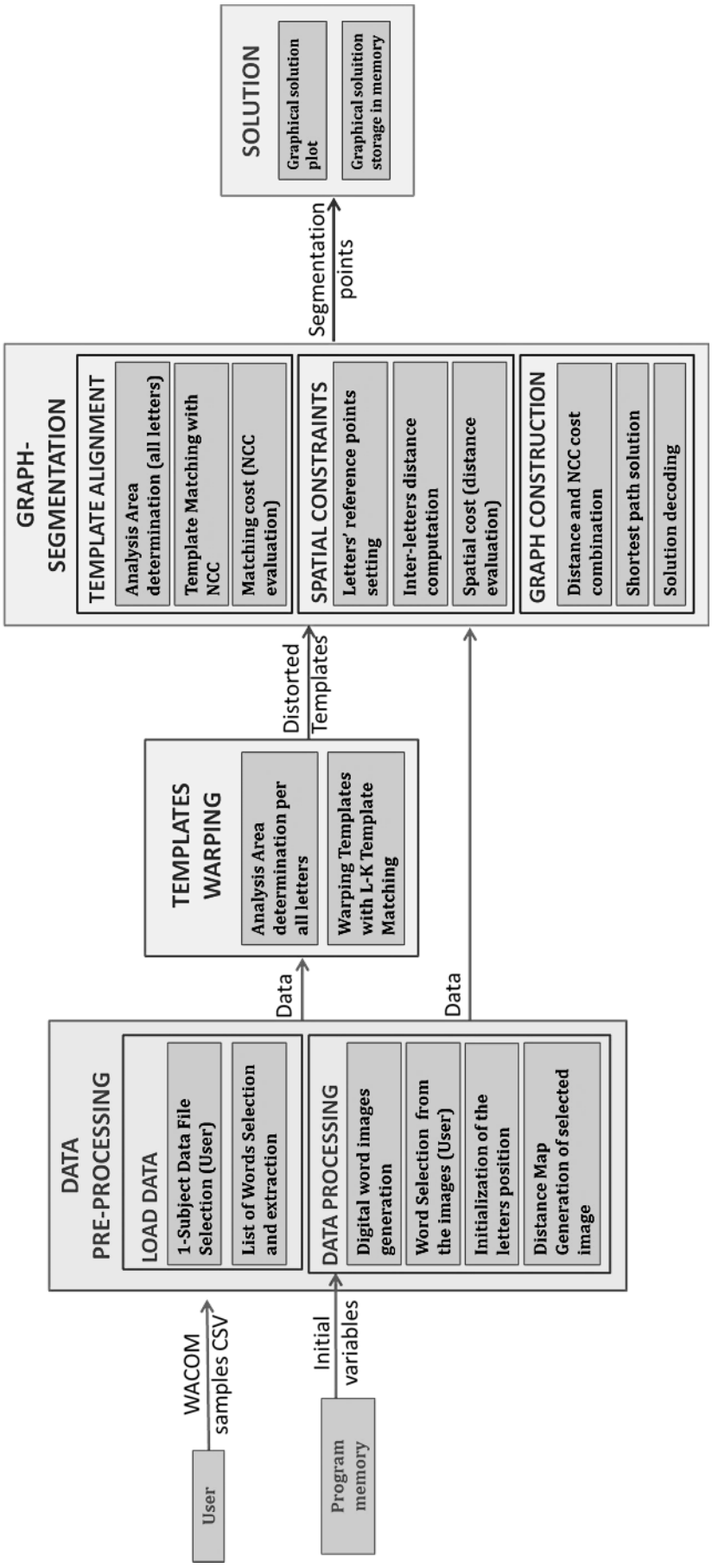


Figure 12: Scheme of the method's pipeline

Data samples from all subjects that participated in the experiment are stored in the same file. Using this data we can reconstruct the words that children wrote and associate to their movements some other information such as writing fluency or speed, among other options.

Together with this file that contains data of the handwriting of children, it is provided another (.csv) source file. This second file contains the list of words that were dictated to the children during the same recording experiment. It is essential to have these two files because it defines one of the basis on which this work is centred: The content of the words to be analysed is known from the beginning, it does not have to be decoded by the program as it happens with other handwriting character recognition techniques.

Figure 13 and Figure 14 show an abstract of the (.csv) type of data files that are used as input to the method.

	A
1	PacketSerial,subject,slice,writing,group,PacketTime,X,Y,Z,NormalPressure,Azimuth,Altitude
2	1,1101,0,False,-1,20508881,10469,17050,-1,0,2880,870
3	1,1104,0,False,-1,23959005,9594,20431,-1,0,2700,890
4	1,1106,0,False,-1,4845092,4496,18119,-1,0,2700,870
5	1,1107,0,False,-1,2852667,9127,27929,-1,0,310,840
6	1,1108,0,False,-1,22004932,8012,18468,-1,0,2250,870
7	1,1110,0,False,-1,5072419,27081,14497,-79,0,3550,370
8	1,1112,0,False,-1,4864277,14949,5201,-1,0,3460,860
9	1,1118,0,False,-1,1408046,7368,19374,-1,0,560,860
10	1,1119,0,False,-1,3125504,6827,22701,-1,0,1800,870
11	1,1120,0,False,-1,1420189,5095,18563,-1,0,1350,860
12	1,1122,0,False,-1,2904418,31128,16864,-1,0,2360,860

Figure 13: Abstract of the input data file of children's handwriting recorded by WACOM SmartPads (CSV)

	A	B	C
1	0	arbre	
2	1	avion	
3	2	avril	
4	3	bébé	
5	4	bouche	
6	5	boutique	
7	6	bouton	
8	7	branche	
9	8	cave	
10	9	chapitre	

Figure 14: Abstract of the List of words input file (CSV)

Reference data

Psychological researchers also recorded handwriting data to be used as a reference. For this, they wrote the same words that were dictated to the children. Moreover, they also recorded data of the handwritten letters of the alphabet, isolated. All these reference samples were recorded using the exact same system that was provided to the children for the experiment, so that reference data is significant for the development of this segmentation methodology. Thus, the data structure of these reference files is similar to the one described previously. The only difference is that in these two data files there is no information about "*subject*" writing, because all words are written only once and by the same subject (concretely, handwriting belongs to C. Gosse). Data encoded in the letters reference data file corresponds to the recording of the 26 letters of the alphabet + "é", "è", "â" written individually. This last file does not have its associated list file because the alphabet is standard and this information is provided.

4.1.2 Assumptions

Words known a priori

From the available data described above it can be observed that one of the main assumptions for the current work is that words being analysed are always known a priori. This gives an important characteristic to the project and differentiates it from letter-identification works such as plate reading or OCR methods. Given this condition, the necessary templates to be matched in each word are also known before proceeding to the analysis.

Non-regular writing

Handwriting words that have to be segmented come from children from 8 to 11 years old that are in the middle of the process of learning handwriting skills. During these years, they are taught about orthography and grapho-motor gestures to generate words. Because they are still learning, their gestures are not still automatized and their handwriting presents a high degree of variability, and even spelling mistakes.

Cursive writing

As mentioned, we work with words written by children learning how to write; particularly, in Belgium. These children are taught cursive handwriting firstly, and in the handwriting experiment, they were only allowed to use this kind of writing. Although intrinsic irregularity of the children's handwriting exists, this assumption simplifies the work because it is not necessary to consider multiple possible shapes for each letter such as: Cursive, "printed-style", "Upper case"…

4.2 INITIALIZATION SPECIFICS

There is a set of variables that are precomputed and provided with the program code. These variables do not depend on the particular analysis being carried out so we have decided to precompute and store them to reduce computational time during execution. The program can access to this data along the different method phases.

In this section, we give more details of how these variables have been generated so that a better understanding of the algorithm design is possible.

d) Alphabet

The first of the precomputed variables is a character of all the alphabet letters concatenated. The length of this variable is 29 characters because it contains the *a* to *z* letters with the addition of *é, è, â.* It is stored as a Matlab variable (.mat). It is used for comparison and letter search in multiple sections of the algorithm.

e) Letters' reference points

As mentioned, reference letters of the alphabet were written and recorded by C. Gosse (*UCL*) in the same way than the children's handwriting was recorded, through the SmartPads. This data is processed to generate the images that correspond to the letters of the alphabet. This process is very similar to the way children's word images are also extracted from initial data. Therefore, the methodology to generate the digital images from input data files is explained with more detail later in this chapter, for the case of children's handwritten words extraction (Section 4.4.3).

At first, letters are generated as square images, all of them being of the same dimensions, of 875x656pixels. After that, each image is adjusted to its letter contour but leaving some lines of background pixels to provide some margin for the distance transform to have an impact, as it is justified below. These final images are generated and stored as (.tif) images. An example of this process can be appreciated in Figure 15. Nonetheless, these images are not provided to the user with the segmentation program; they only serve during the program development to generate other associated data, in particular letter reference points and letter distance maps.

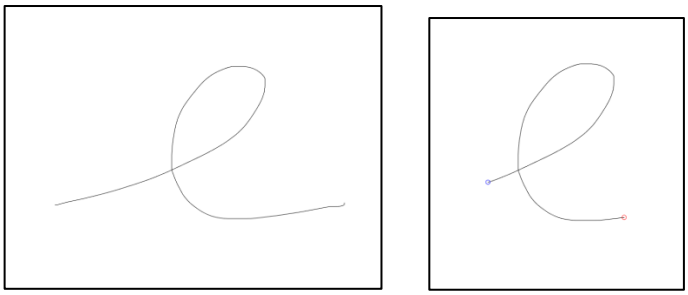


Figure 15: Original digital image of the letter "e" (left) and the adjusted image (right)

For each one of the letter templates, two reference points are marked: the "right-point" and the "left-point" (also called "end" and "start" reference points in this work, respectively). These points are important to locate letters within a word based on an initial and a final point. Some examples are shown below:

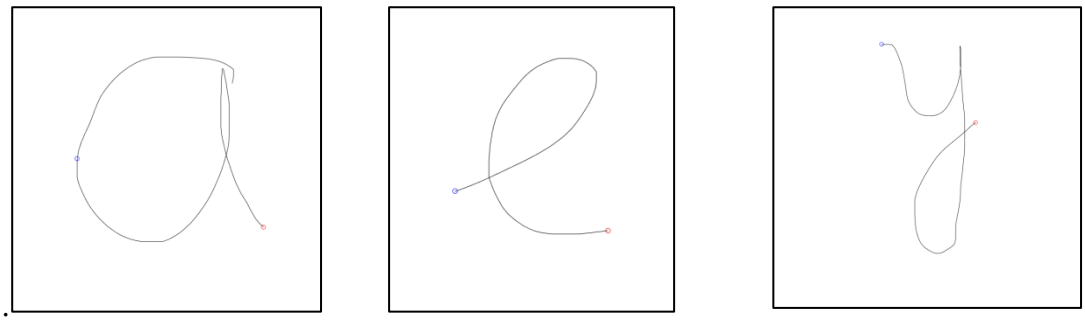


Figure 16: "Start" and "end" reference points on some letter images

The two reference points of all letters are saved in a Matlab variable (.mat) as the vertical and horizontal offset from the upper-left corner of the image to the respective points. This variable is available for the method application during execution.

f) Templates

The templates that we use in our method are the Distance Transforms of the previously described images of the isolated alphabet letters. Differently from the word's DT, which has to be computed during execution for the selected word, all raw templates DTs can be precomputed and stored. During word analysis, only the DT of the letters in the word under consideration are called and used by the algorithm.

Actually, the templates (distance maps) are stored as Matlab variables (.mat) containing the array with the greyscale levels of the associated images. These arrays have the same dimensions than the letter images used to generate the DTs, what means that size of the templates varies from one to the other.

To generate the pre-computed templates, the digital images of the letters are used. The methodology followed to generate the DTs of these images is similar to the methodology to generate the handwritten words' DTs. This is explained with more detail in next section of this chapter.

During execution, the algorithm calls the stored variables that contain the templates data needed in each situation. The template matching steps of this segmentation method are performed by comparing the Distance Transform of a particular handwritten word image (Source Images) and

the Distance Transform of the associated letter images (Templates). From now one, when we refer to the Templates, we are referring to these Distance Map Images of the letters, not to the previous binary images.

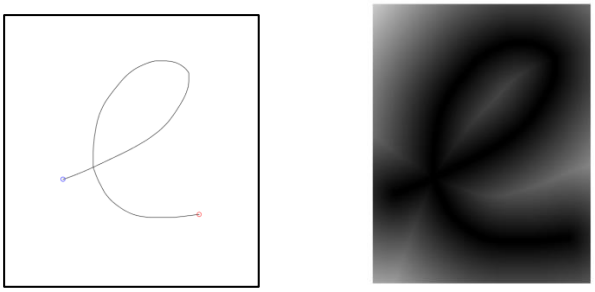


Figure 17: Example of the adjusted image of the letter "e" (left) and its Distance Transform visualization (right)

The template image associated to all the letters in the alphabet have different dimensions, depending on their graphical contour. For example, the Distance Map template in Figure 17 has final dimensions of 500x390pixels. The templates have to be resized before being used for comparison.

As it is remarked in the section that describes the words Distance Map generation, the distance maps of the letter images are post-processed and normalized, what results in a greyscale range that belongs to: $DM \in [0\ 255]$.

4.3 DATA PRE-PROCESSING SPECIFICS

A part from the pre-computed set of variables (previous section), the execution of the program requires two data files: the samples data files and the list of words, in the specific format in which they have been presented above.

It may be necessary to pre-process the samples data file if file size exceeds the program capacity. Next section gives further details on data processing.

4.3.1 Preliminary data rearrangement

The first step is loading the specific data files that contain the handwriting data that hast to be analysed. In this work, data is assumed to be provided in the specified format in Section 4.1.1. We use data from the experiment designed and conducted by M.V. Reybroeck and C. Gosse (*UCL*).

Children's handwriting is originally contained in a single huge (.csv) file. The user is asked to split this data into multiple subfiles that contain no more than 1 million samples per document.

For the development of this work, we have used a CSV splitter application. It is suggested to the user to use this same executable or another of his preference [28].

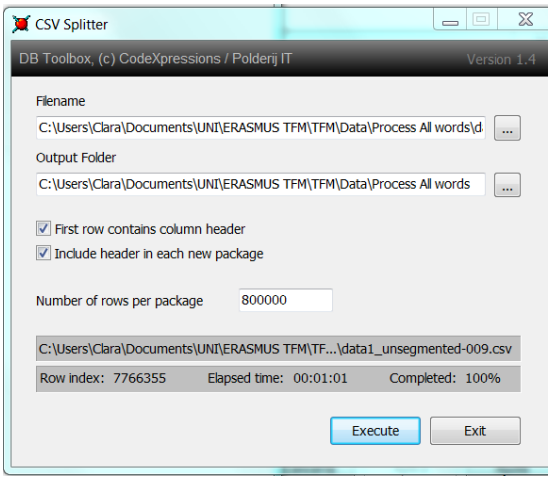


Figure 18: CSV-Splitter Graphical User Interface of the suggested software

In any case, this step has to be carried out outside the main program of our method; so that when the program is launched, CSV files with the appropriate size are available, (each one) with all data of different words written by different subjects.

4.3.2 Data file selection

The data file(s) contain data samples that belong to different words from different subjects. In contrast, we have designed this method to work with each subject's data separately. For this reason, we present two main options available to start with data-processing:

i) First time: With the "first time" option, the program reads the initial CSV file(s) and generates a new file per each subject, with all data samples associated to his handwriting. To do this, the program reads the content of the original files as a *table*, and uses the *Subject* variable as the criterion to assign each sample to a new file, depending on who does it belong to. Subjects are recognised with a four-digit code. Thus, the first time that the program finds a sample of a particular *subject* it creates a new CSV file named after the subject's code. The rest of the samples that are encountered and belong to the same subject, are added as new lines of the same document. So at the end, the program has created the same number of CSV files as the number of subject that participated in the recording experiment.

ii) Subject's Data File: If the initial data has already been rearranged based on the writing *subject* of the samples, the user has to select the specific CSV file of a specific subject. The words encoded in this file are processed in the next phases. So this selection serves the program to locate the source data file and to find out the 4-digit code of the *Subject* whose data will be analysed in the following steps.

To sum up, to apply this method we work on Wacom samples split per *subject* (identified with a 4-digit code), and we work on one-subject's handwriting at a time. We include these pre-processing steps in our development.

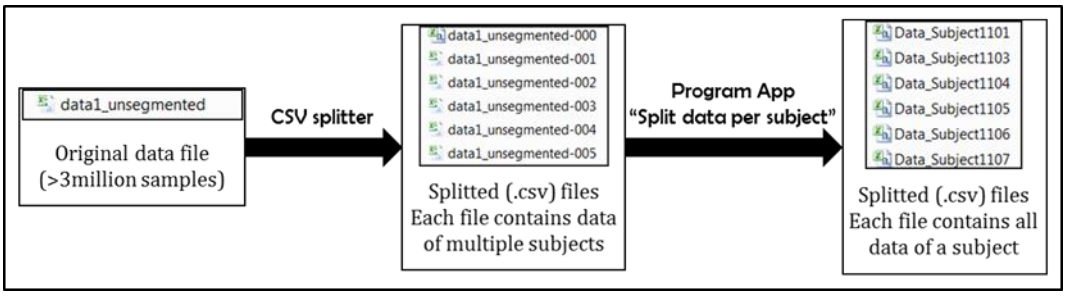


Figure 19: Scheme of data pre-processing applied to data recorded by WACOM SmartPads and stored in a CSV file

4.3.3 List of words' characteristics

Besides than then samples data file, it has to be loaded the CSV data file that contains the list of words that were dictated to the children during the handwriting experiment session.

This list is used for two main functions. First of all, it is used to generate an internal variable (.mat) that contains relevant information about all the words' characteristics. And secondly, it is used for the graphical user-interface's configuration.

This internal variable is an array, with one column per word in the list. The information associated to each word is included in the different rows of the corresponding column. The contents of the three rows consist in:

1) Digital encoding of the word: A character vector with all the characters that belong to the text of word.
2) Number of letters: The length of the character vector that represents the word is used to determine the length of the words in the list. This is sored as an integer variable in the array.
3) Word type: Word classification based on geometrical characteristics of the words based on a similar geometric classification of the letters.

List_words

4x155 cell

	1	2	3	4	5	6	7	
1	'arbre'	'avion'	'avril'	'bébé'	'bouche'	'boutique'	'bouton'	'b
2	5	5	5	4	6	8	6	7
3	'up'	'small'	'up'	'up'	'up'	'both'	'up'	'u

Figure 20: Abstract of the variable that contains word's characteristics. Each column belongs to one word of the list and the 3 rows define each word's text, the number of letters and type.

With respect to the *word type* feature, we have created a classification for the alphabet letters based on their graphical dimensions in cursive writing. The four possible labels are: "up", "down", "both" and "small". Letter classification possibilities can be seen in the Table below:

Type of letter	Letters that belong to this type
Small	a,c,e,i,k,m,n,o,r,s,u,v,w,x
Up	b,d,h,l,t,è,é,â
Down	g,j,p,q,y,z
Both	f

Table 3: Alphabet classification based on letter's graphical characteristics

The words are classified into four groups with the same labels depending on the letters that contain. This information is relevant to generate a spatial grid on which it is possible to approximately locate the different letters in a word, based on the expected dimensions that each letter may have. This is further developed in the *letter's initialization* (Section 4.4.2), in which this information is used.

4.3.4 Data extraction from files and Image generation

At this point, all necessary files required for the analysis are already loaded. The next step is to proceed with information extraction from samples of the selected subject's handwriting.

The content of the samples data file is now read from its directory and the content is loaded into the program memory as a *table*. Actually, instead of loading all the content from the CSV file, only the indispensable part is loaded; i.e., only the *table* columns with data associated to *X*, *Y*, *writing* and *group* features.

Group is used to associate each sample to a different word in the list. *Write* is used to distinguish from all samples of each word, those in which the pen was touching the paper, i.e. the child was writing, from those that correspond to air-movements of the pen. Finally, *X* and *Y* correspond to the 2D coordinates of the pen tip with respect to the tablet plane, and they are used to reproduce the words as digital images.

For each word, a new internal variable (.mat) is created. These variables are three-row arrays, which have as many columns as samples belong to that word. The first row contains X-coordinate, the second one contains Y-coordinates and the third row consists in a binary variable worth 0 or 1 depending to *writing* initial parameter. When this third variable is equal to 1, it means that the associated sample (i.e. associated coordinates) corresponds to an instant in which the pen was touching the screen, therefore, the child was actually drawing a letter-stroke; meanwhile if the variable is worth 0, the child was moving the pen close to the screen but without drawing any contour line.

3x1105 double

	1	2	3	4	5	6
1	13875	13875	13875	13877	13885	13894
2	5171	5164	5161	5159	5159	5159
3	1	1	1	1	1	1
4						
5						

Figure 21: Abstract of the variable that stores data from a word's samples. Each column belongs to one WACOM sample and the three rows define X, Y and Writing Condition, respectively

This information is essential to be able to reconstruct the digital word images in an accurate way with respect to children's graphical writing. To generate the digital images of the handwritten words, the samples have to be plotted in the same order in which they were recorded. Moreover, we have into account two main considerations for this step: To plot the word contour line, we only plot the 2D coordinates of samples whose associated variable "*write*" indicates that the pen was actually touching the screen (*Write* = True). And, secondly, non-writing samples in the middle of the same word indicate different strokes; hence when this is encountered, a non-continuous contour is plotted by leaving a gap between consecutive sample-points.

The images that correspond to all the selected subject's handwritten work are generated and stored in (.tif) format in a new folder named by the subject's code inside the Results folder (also created by the algorithm). The saved images have all the same dimensions of 656x875pixels.

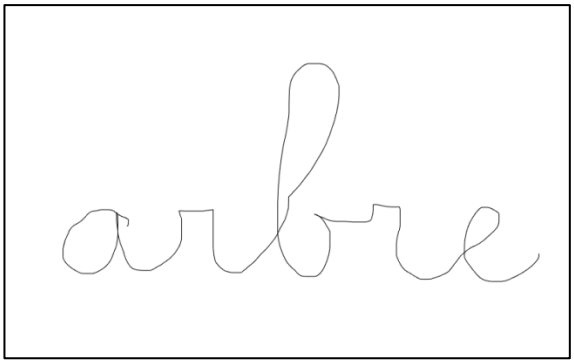


Figure 22: Example of the output of the image generation step applied to the handwritten word "arbre"

Also, data related to each word's construction are stored as Matlab variables (.mat). These variables are the connection between original data and word images on which the algorithm works from this point on.

4.4 IMAGE PROCESSING

To introduce image processing and manipulation section, we want to note that it is often necessary to reference relative position between images or to locate a particular pixel within an image. For this, a reference point is determined. During all this work description, we refer to images position or coordinates by always using their upper-left corner as reference. It is important to have this in mind during the entire lecture to find coherence between the different method stages.

4.4.1 Word selection

From this point on, the analysis steps are executed on a single word image, which is selected by the user. Hence, we process one word at a time as well. All words have an associate numbering. This number is read by the program and stored as a program variable. It serves to identify the word within the list and read its associate word characteristics from the variable that contains them (*List_of_words.mat*, in Section 4.3.3).

Per each selected word, all the necessary steps until complete word segmentation are applied. This transcript presents the program operation mode step-by-step, but an option to process all word in a black-box mode is available in the program GUI.

4.4.2 Estimates initialization

Thanks to the fact that the word text to be identified in the selected image is known, it is possible to compute an estimate size and position of the different letters that compound it. All the letters are associated the same width, by assuming a regular text distribution in the image. The height associated to each letter is different depending on the letter's graphical characteristics, following the classification presented before, in Table 3. So, the previous defined word-classification is used in this new step.

That classification is actually made based on coarse space distribution. The letters considered as “small”, which constitute the most dense and frequently used group, are taken as the reference for the body of the word. The word classification aims to define if other letters in the word, which are not “small”, are bigger than these ones in the upper direction or the lower direction, or both. Figure 23 shows an example which clarifies the explanation.

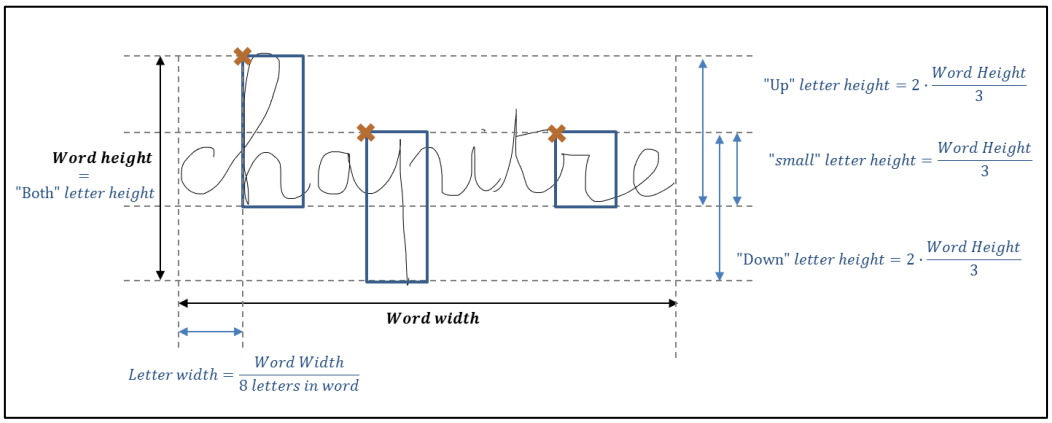


Figure 23: Example of the letters estimate position definition based on spatial characteristics

The whole word width is divided into the number of letters in the word to have an approximation of each letter’s width, under the initial assumption of all letters having a similar size. Each letter is associated an approximated height based on its graphical shape. This classification not only gives the estimate word dimensions as exposed, but also the relative position of the letters in the word images.

According to these dimensions, the estimated position of each letter is computed as a bounding box represented by 4 parameters, which are: 2 coordinates of its reference point, the box width and the box height. The reference point is the upper-left corner of the bounding box and it is stored with 2 parameters, the corresponding column and row of the array that represents the image.

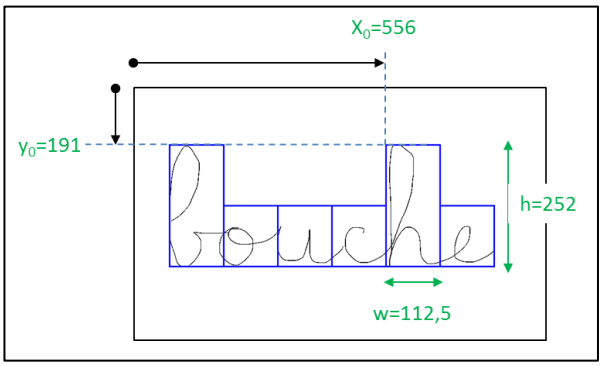


Figure 24: Example of the word “bouche” letter’s initialization and determination of the bounding box of the letter “h” through its upper-left corner coordinates and its width and height magnitudes.

This new information is computed for all letters in the selected word and added in the previous created variable “*List_words.mat”* that already contained word information (word characters, number of letters and word type). This variable contained three rows and one column per word in the list. So, estimate position information is stored in the fourth row of the column associated to the word under analysis; and it is stored as an array with one row per letter in word and 4

columns to store the 4 parameters to locate each letter. Figures below show an abstract of the variable with the words information.

List_words

4x155 cell

	1	2	3	4	5	6
1	'arbre'	'avion'	'avril'	'bébé'	'bouche'	'boutique'
2	5	5	5	4	6	8
3	'up'	'small'	'up'	'up'	'up'	'both'
4	5x4 double	5x4 double	5x4 double	4x4 double	6x4 double	8x4 double

Figure 25: Abstract of the variable with an added 4th row with the letter's initialization information of each word

	1	2	3	4
1	116	191	112.5000	252
2	228.5000	317	112.5000	126
3	341	317	112.5000	126
4	453.5000	317	112.5000	126
5	566	191	112.5000	252
6	678.5000	317	112.5000	126
7				

Figure 26: Abstract of 4th row of one word made of an array with one row per letter in the word and one column per initialization parameter X, Y, width and height respectively

This information is essential to avoid working with the whole image in further steps. Instead, for each letter's analysis, the region around its estimate position can be used and drastically reduce computational cost.

4.4.3 Distance map

The concept of Distance Map is covered in Chapter 2, with the preliminary concepts. However, it is important to remind that the DM is generated by labelling all background pixels with the value of the distance to their closest non-background pixel (i.e., word contour). This shows up how strongly linked is the result to the distance metrics used for computation. In this work, we use Euclidean distance. Also, from the DM construction steps it is demonstrated that the output image (Distance Transform) has the same dimensions than the input image (binary image); which is 656x875pixels.

To generate the DM, first of all, the image is converted to greyscale. The greyscale levels go from zero for the word contour, seen as black; to 255 for the background, seen as white. In second place, the image is inverted, thus the result is a white contour (greyscale level 255) on a black background (greyscale level 0). Mathematical morphology is applied to this image to make the word contour more robust. In particular, we perform dilation with a small circular structural element. The result is then binarized so that the image is expressed as an array of 1s (for the background) and 0s (for the word contour). The reason of this step is for the DM to be built by considering the word contour (binary 0s) as the obstacle pixels towards which label the distance. Finally, the Distance Transform is built on the binary image.

Figure 27: Mathematical morphology applied to the inverse of the word "arbre" digital image

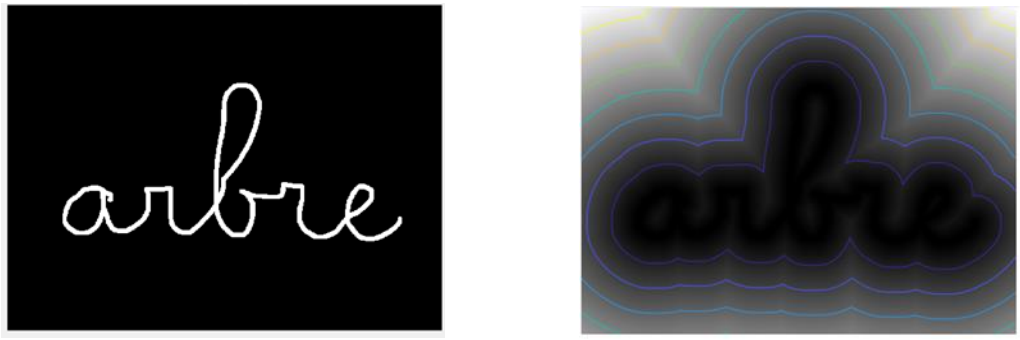


Figure 28: Standard distance map of a the "arbre" word image using Euclidean distance metrics and plotted contour lines that connect same grey-scale values to for visualization

The main raison to use a distance transform of the word image instead of the binary version is to work with smoother images, which favours convergence in further stages. To understand this, we point out that an image can be considered as a scalar function of the coordinates $f(x, y)$. If we use a DM to describe the word image, this image function also becomes smoother, what makes possible to compute image gradients that provide relevant data about the letter's contour and position. This gain in smoothness benefits convergence in optimization algorithms, what can be translated in most efficient and successful computation operations. With a distance map, abrupt changes in the image function are avoided as well.

In this work, we use first Euclidean distance to generate a preliminary distance map, as exposed. But then, this function is multiplied by itself, giving a quadratic result. The quadratic Distance Transform of the image is normalized to make it belong to the range of [0 255]. By using quadratic magnitude, the DM that we obtain is even smoother, and we have demonstrated its advantages on obtaining better results in the whole segmentation problem.

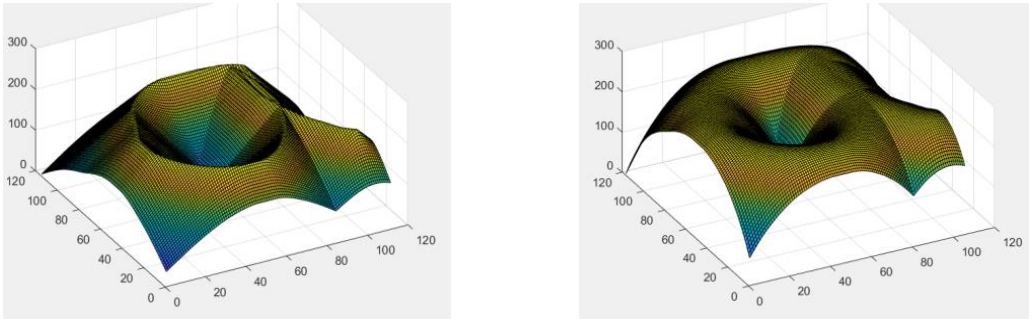


Figure 29: 3D visualization of the distance map of the letter "a". Left: Standard linear version saturated at 255. Right: Quadratic and normalization variant.

The letters in the handwritten words are always considerably different from the templates. Thus it is not advisable to try to match a very precise template contour; instead, it is better to try to match an approximate but very representative shape. From the images in Figure 29 it can be concluded that this can be achieved by using the quadratic version of the distance map as well. This modified DT has a larger region where a letter matching would be considered acceptable. This is observable through the letter contour being much less abrupt than in the linear DT. Thanks to this, the use of this image as a template does not penalize that much those letter contours identified close to the optimum position, but not exactly on it.

For the opposite situation, when a letter's contour is trying to be matched far from its real position, the quadratic template is also more advantageous given the fact that its pixel values far

from the letter contour increase faster than in the standard version. This clearly indicates that the matching is not being accurate.

The templates DMs, which are precomputed and provided to the user, are generated using the quadratic approach; and so is the DM associated to the working image during execution. More details that led to do this choice are included in Chapter 5.

The distance map associated to the selected word is stored as a program variable instead of an image for further utilization of it. This variable is an array which contains the greyscale level of each pixel.

4.5 TEMPLATE WARPING

It is important at this point to remember that the objective of this work is to provide an accurate segmentation of words into letters; not to assess the quality of them and neither to discover the encoded letters in an image (what is already known). For this reason, before proceeding to the template matching step, we are taking some measures to maximize segmentation solution accuracy.

The estimate initialization of the letter position mentioned above can be considered one of these tools. Further than that, the shape of the template is also slightly modified to better match the ones that children wrote in each case. The templates are adapted for the letter in each word.

Precomputed templates are used to generate the new set of warped template letters which are specific for each word. This fact is relevant for segmentation results when calligraphy in the handwritten word is not good. Proceeding this way, the letter limits found out within the word are more accurate than the limits that are found with the set of raw templates that does not almost match children's handwriting (Chapter 6).

We make use of the Lucas Kanade optimization algorithm for template matching to get this set of warped templates. We release 4 DOFs: 2 translational ones and 2 scaling ones; in both cases along the 2D axes of the image. We apply this iterative algorithm to compute the updated parameters vector $\boldsymbol{p} = [T_1 \quad T_2 \quad K_1 \quad K_2]$ that defines in each iteration the image warping that is being applied to optimize similarity between image and template. Although the objective is to warp the templates, the LK method is implemented in such a way that it is actually the image that is warped during the algorithm execution, so at the end of the phase invert the results.

To follow with this step description we must remark that an initialization of the parameters vector is required. In our method, this initialization is based on the first estimate position and size of the letters (Section 42). To improve performance and robustness, we use various initializations per each letter's analysis, so that the sampling is enlarged. The way we do this is by applying some variations of the initial estimation.

The achieved optimum solution after an initialization is stored in a program variable, as well as the associated value of the objective function. Then, the process is repeated for the next initialization vector of the list of variations. Once all initializations have been used for computation, we select the results with lower associated value of the optimization function, i.e., the minimum value of the SSD between the raw template and the distorted image. The scaling parameters that have led to that image distortion are inverted and used to warp the template. The warped template is finally stored and it becomes the output of this phase. After this, the whole

process is repeated for next letter, with same variations on its initialization as well; and successively for all letters in the word. The templates are distorted and saved as Matlab internal variables (.mat) for next steps.

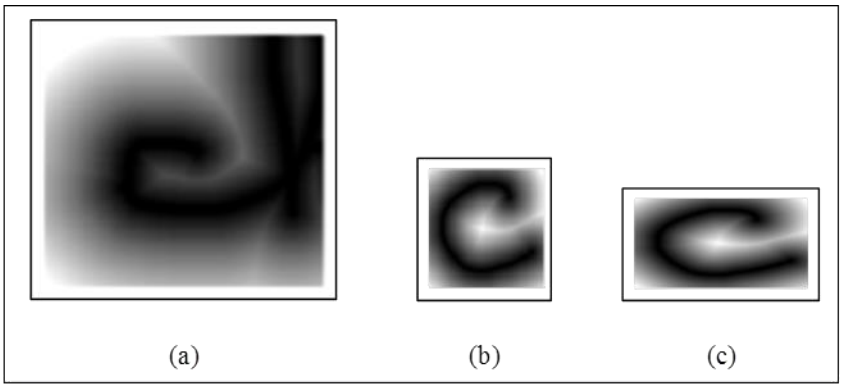


Figure 30: Example of template warping for the "c" in the word "cheval". (a) First letter's AA, (b) original template and (c) warped template (c) to better match the letter in the Analysis Area

Figure 30 shows an example of template warping on the letter "c" of the French word "cheval" written by one of the Subjects that participated in the experiment.

An important fact to mention is that LK algorithm is very sensitive to the initialization and differences in greyscale values, what justifies our choice of using multiple initializations per letter's processing. To help the results to be reasonable, we also limit the allowed warping during iterations in such a way that the output template can't be too small or too large; concretely, its size has to be bigger than half its original size and smaller than twice its original size. Moreover, translation is limited as well so that the obtained estimate position of the letter does not belong to the image borders. This second restriction is applied for two main reasons: (i) for accuracy, given the fact that the letters are never that close to the image border (images are generated with blank margin around the word contour), thus it helps the algorithm to converge in the appropriate direction; and for further steps, to be able design a new Analysis Area around the obtained estimate in which we can translate the warped template.

Notice also that we only select the best solution for a particular template warping among all the solutions that are obtained after the various initializations. Instead, we could also consider the multiple local minima resulting of the whole phase of a template's processing. This strategy would increase robustness of the final segmentation solution with the drawback of increasing computational cost of the next steps.

4.6 GRAPH-BASED WORD SEGMENTATION

The new templates are used together with previous data to implement the graph-based word segmentation in letters. This word segmentation is based on two main aspects: First, a fine alignment of the warped templates on the image, controlled by template matching with NCC; and second, the distance between end and beginning of consecutive letters.

We remark that this is one of the main points that differentiate our work from previous handwriting recognition algorithms. At this stage it is crucial the other main feature highlighted during all this work description: the fact that the content of the words being analysed is known a priori. We finally remark that the objective of this work is to find the optimal segmentation of the entire known word in letters other than to discover the content of the written word itself.

With this objective, we construct a graph that contains all possible connections between consecutive letters depending on where are they assumed to be located within the original handwritten word image. This graph aims to select the optimal position to place each template on their respective Analysis Areas based on the whole word segmentation optimization criteria. In other words, the graph aims to jointly optimize the spatial coherence of the word and the matching of its letters with the corresponding (warped) templates.

4.6.1 Graph-based word segmentation formalism

With the warping template's phase we have determined a set of templates that have to be matched on the word image. We then consider that each warped template can be translated locally; and its final placement is the result of the word segmentation problem.

To implement the segmentation problem in a graph-based representation, we associate the nodes to specific translations of the warped letter templates. The edges in the graph are directed and represent the nexus between consecutive letters depending on the involved templates translation. The cost associated to the graph edges has two components: first, the Normalized Cross Correlation results between the translated templates and the word letters; and second, the distance between the end of a template and the beginning of the next one, for each pair of consecutive letters. NCC outcome is obtained after a Template Matching analysis per each template being translated within the surrounding pixels of its estimated best position on the image (LK output). These surrounding pixels define the Search Windows for the TM phase.

To build this graph, we use a tree-structure and define an auxiliary node called *source* to be the root node. This root node is connected to all possible template positions (result of the warped template translation) for the first letter in the current word. This means that the source node is connected to as many numbers of nodes as the number of pixels in the first letter's Search Window. The link between nodes is always parent-to-child (Section 2.4.2).

At the same time, each one of these nodes is connected to all possible positions of the second letter's template within its respective Search Windows. Again, this means that each pixel in the second letter's Search window is a possible position for the second template

This same idea is applied for all letters in the word except for the last one. It can be thus appreciated that the graph is structured in levels, one per each letter. The final graph has a shape that differs slightly of a standard tree graph because all the nodes of the last level are also connected to another auxiliary single node called "sink". The nodes of the last level represent all the pixels of the last letter's Search Window, where the last letter's template can be placed.

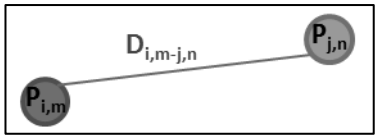


Figure 31: Connection between two nodes in the graph-representation of the segmentation problem

Take the connection between the two nodes, $P_{i,m}$ and $P_{j,n}$ in Figure 31. We have declared that nodes represent template positions on the letters' Search Windows; or, equivalently, nodes correspond to a specific translation of the warped letter templates.

So, the first sub-index indicates the letter to which the node is associated; and the second sub-index indicates the particular number of pixel in that letter's Search Window. Then, node $P_{i,m}$ represents the case in which the template of letter i is placed in the pixel number m of its SW; and similarly for the node $P_{j,n}$; where i and j are consecutive letters of a word.

The edge $D_{i,m-j,n}$ connects the two mentioned nodes, thus it represents the connection between the letters i and j if the template of the letter i is placed in the position m of its SW and the template of letter j is placed at the position n of its SW.

Following this nomenclature, Figure 32 presents the general structure of the graph that represents the word segmentation problem.

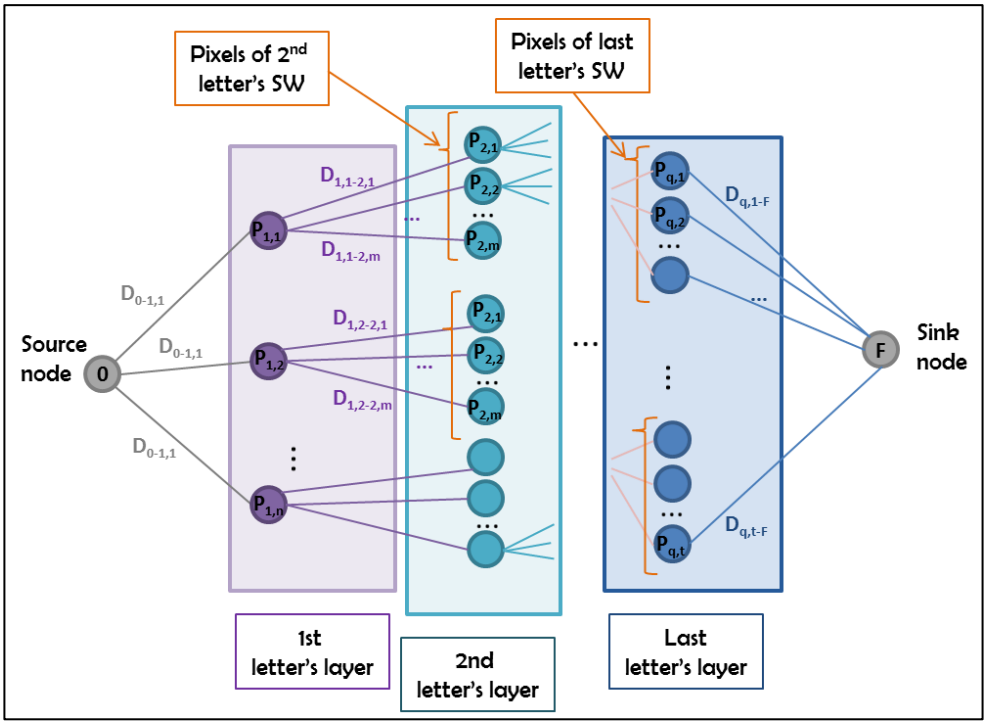


Figure 32: Scheme of the graph representation of the segmentation problem

From the graph's example (Figure 32), we can tell there are q letters in the word being segmented. We can also tell that the Search Window associated to the letters in the word have different size: being n pixels in the first letter's SW, m pixels in the second letter's SW and t pixels in the last letter's SW; or equivalently, the local translation possibilities for the templates, respectively.

The direct interpretation of the graph leads to the association of the NCC results to the nodes, as a result of the matching assessment between the warped template and the word for the translation referenced by the corresponding node. In order to work according to graph-theory, all costs must be included in the graph edges, and none in the nodes; hence NCC results are transferred to the edges as well. In consequence, the cost of the graph edge that connects two given nodes accounts for the NCC outcome associated to the parent node and the distance associated to the parent-child connection.

The segmented solution is obtained on the graph representation by means of the Dijkstra shortest-path algorithm. The shortest-path solution minimizes jointly optimizes the dissimilarity

of the handwritten letters in the word with respect to the modified wrapped templates (based on NCC results) and optimizes the spatial coherence between the letters (based on the distance between consecutive letters).

4.6.2 Fine Alignment of Warped Templates: NCC Template Matching

In this work, we find two phases in which template matching is applied: The first one, where the distortion of the templates is determined; and the second one, where we only allow local translation of the warped templates. Hence, the warping selected after the Lucas Kanade method application is no longer modified; the warped templates are compared with the image in different positions. In this section, we present the second of these phases.

It is of great interest to store the similarity between the image and the warped templates for the set of considered template translations. Hence, instead of using greedy approaches, we rather compute the matching assessment for all the possibilities of relative position between the image and the templates. At the end of the method, it may be preferable to select a non-optimal template position for a particular letter in a word which may maximize the global word segmentation result.

We compute this matching assessment by means of Normalized Cross Correlation. The selection of this technique is discussed with detail in Chapter 5.

We retrieve the warped templates and the estimated best position from the previous phase and compute template matching with NCC between the image and the template shifted around the estimated best position (locally). Then, we evaluate these results to include them into the graph-based formulation of the problem in a more representative way. The evaluation of the results is done with two main purposes.

In first place, the NCC assessments have to be inverted to make them suitable as a *cost* to be associated to the graph edges. A priori, NCC results have high values (1) for good matchings and low values (-1) for poor matchings. To use this data in *cost-form*, we force low matching assessments to have an associated high cost and vice versa.

In second place, we want to associate this cost in a non-linear way, such that we do not penalize much the best-matching positions, while we want to penalize in increase the template positions whose NCC result indicates a poor matching. We want to guarantee that various translation options are considered as candidates and the graph can constraint the solution to the spatial constraints.

Figure 33 shows the graphic representation of the evaluation function that we have design for the mentioned purposes.

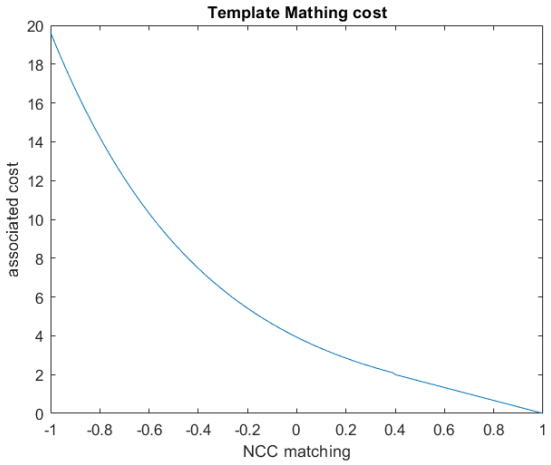


Figure 33: Graphical representation of the NCC template matching results' evaluation function

4.6.3 Inter-letter distance

For word segmentation, not only similarity between the warped templates and the word to be segmented is taken into account; spatial coherence of the result is considered to be another important feature. Basically, the inter-letter distance is used as a measure of the spatial coherence to constraint the word segmentation, together with NCC results. The term "inter-letter distance" denotes the distance between the end of a warped letter template and the beginning of the next warped letter template.

To clarify this, we show an example. Take the French word "arbre", for example. The algorithm is going to proceed to detect that the templates to be used are the ones that correspond to the letters "a", "r", "b", "r", "e", in this exact order. When the first letter's template is placed at a particular position (from its corresponding Search Window), a constraint is raised which imposes that the beginning of the second letter, "r", must be placed more at the right on the source image than the ending of the previous "a" recognised letter. And this same happens for each pair of letters in the word.

To manage this "start" and "end" points of each identified letter, the reference points that have been presented at the beginning of the chapter are used (Section 4.2). The distance between letters is computed based on these two points per each one, which are also the reference of their total width. Hence, the distance between letters is computed as the Euclidean distance between the *right* (end) reference point of one letter and the *left* (beginning) reference point of the following one. This magnitude is computed for the relative position between consecutive letters for all possible translations of both letters' templates; or equivalently, it is computed per each pair of consecutive letters for their templates being placed in all possible pixels within their respective SWs.

In this case too, we adapt the distance results to a *cost-significance* version that we can include in the graph-based representation of the problem. A priori, distances can already be interpreted as a cost, being larger magnitudes associates to higher costs. However, this is not completely suitable in our method. First of all, we consider working with distance ratios instead of absolute magnitudes to make this step robust in front of different dimensions of the handwritten material. Thus we divide all distance values by the approximate letter width of the word being analysed; this is, by the total word width divided by its number of letters (Section 4.4.2).

$$\delta = \frac{link\ distance}{letter\ width}$$

In second place, we evaluate the distance ratios based on common distance ratio between letters. We have measured by hand the inter-letter distance for multiple handwritten words from the reference data set; computed the link ratio and observed its most common values. Here, the term "link" denotes the stroke that connects consecutive letters in cursive handwriting. We penalize the template translations that give raise to certain distance ratio values according to statistical results. In summary, we assign low cost values for ratio distances below the mean ratio ($\delta = 0.34$) and we gradually increase the associated cost when the ratio is above this mean value but still below a defined threshold of $\delta = 0.67$. Finally, we associate a rapidly increasing cost to ratio values above this threshold.

Besides, we face the case of negative distance values. Because the distance is computed between the end of a letter and the start of the following one, we assume that negative distances are always a consequence of a mismatch, and it would represent an overlapping between consecutive images. In consequence, we always assign a high cost to letter connections that cause negative values of inter-letter distance.

To sum up all these information, Figure 34 shows the graphical representation of the inter-letter distance ratios evaluation function.

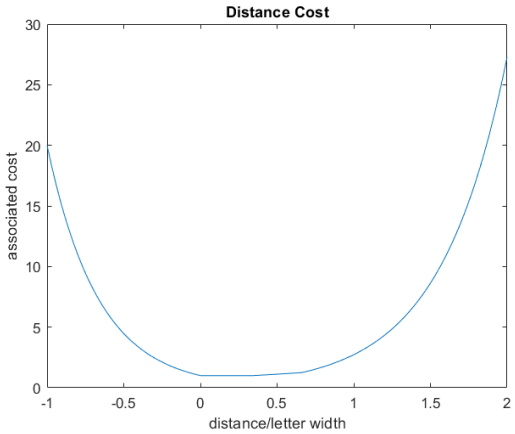


Figure 34: Graphical representation of the inter-letters distance ratios' evaluation function

4.6.4 Cost matrix and shortest-path solution

Up to this point, we have the evaluated the results of the warped templates alignment, controlled by NCC and the results of the connection between letters; both for the same set of template shifting possibilities. Notice that the output domains of both evaluation functions are designed to be coherent and in the same order of magnitude. This is crucial to combine the results of both constituents in a representative manner.

These results are stored in matrixes; thus to combine the results, we work with matrix manipulation. The final implementation of the graph-based representation in our method is done through matrixes as well. The complete matrix of the whole graph representation is a block diagonal matrix. Each one of the matrix blocks contains the cost of the connection between each pair of consecutive letters in the word. All the matrix spaces that do not belong to the block diagonal are assigned a value of infinity, meaning that no connection is allowed. As a result of

being a directed graph, this last fact applies to the inverse connections of consecutive letters and to non-consecutive letters as well.

This matrix structure brings with it the restriction of using one pixel of each letter's SW to place the corresponding template and find a path from the source to the sink nodes. This path is the solution of the word segmentation problem and it is determined by means of the Dijkstra's shortest-path algorithm (Section 2.4.3).

Only the nodes that represent translation of the warped templates on the image are included in the graph. In consequence, the shortest-path solution is the local translation that has to be applied to each letter's template to optimize the word segmentation according to our optimization criteria (similarity between the templates and the word, and spatial coherence).

4.7 SEGMENTATION COORDINATES AND GRAPHICAL SOLUTION

The shortest-path solution determines the most convenient translation for each warped template; or, in other words, the most appropriate pixel of each letter's SW where its template must be placed to optimize letter identification. Thus we convert the local coordinates within each SW to global coordinates with respect to the global image reference frame.

The reference points are used one more time to determine the segmentation points. First of all, we find the updated *start* and *end* points of the distorted templates by applying the same warping transform to its coordinates. Then, we place the warped templates in their respective positions and match their reference points to the word image. These reference points usually do not belong to the word contour as a consequence of the different shape of the letter templates and the letters in the word. To solve this, we associate these points to their closest point on the image that belongs to the word contour. These new points are the final solution of the segmented word.

Given the fact that in cursive writing there are links between letters but the isolated letters do not have these connection strokes, it often exists a gap between the end and the beginning of consecutive letters.

Our method provides a graphical representation of the segmentation solution that has been obtained. With this solution, psychological researchers will be able to assess separately each letter, depending on its graphical characteristics and the position within the whole word.

The *start* and *end* points of each letter are plotted on the original image of the handwritten word. To help in visualization, as a default configuration, vertical straight lines are plotted on each point so that the limits can be easily observed. The way to correctly observe the result is by looking at the intersection point between the vertical lines and the word contour; in particular, in the links between letters of the word contour. However we conceive that it may be preferable to keep the segmentation points as dots instead of vertical lines (Figure 35). The graphical solution can be easily adapted to personal preferences if this is the case. In any case, the limits associated to each letter are plotted in a different colour to facilitate the results understanding.

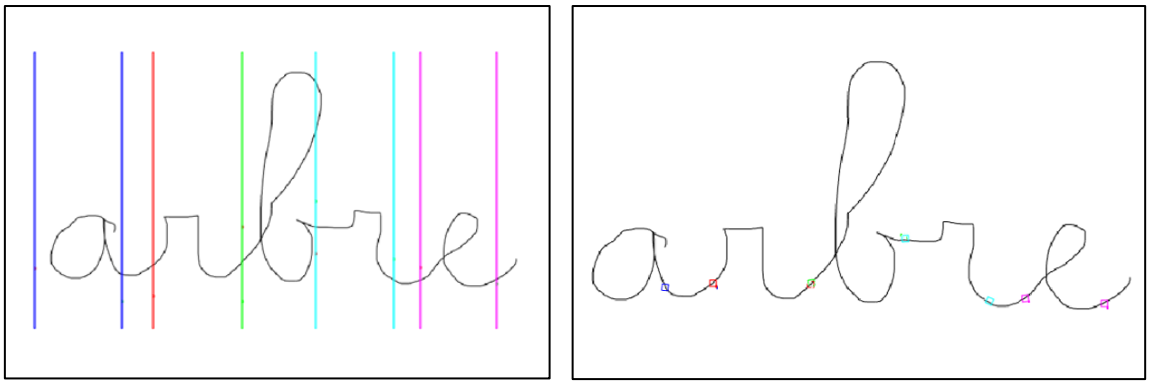


Figure 35: Graphical visualization of the segmentation result for the word "arbre" represented by vertical lines (left) or dots (right) over the solution points on the word contour.

The image that contains the solution plot is automatically stored in (.tif) format.

If desired, the user can reset the executable and start a new word segmentation process. If this is the case, the application brings the user back to step in Section 4.3.2 for word selection. To start processing handwriting of a new *subject*, the user must load another data file with new samples.

4.8 GUI

All the steps described along this chapter have been implemented in different Matlab functions. The main program that connects all the methodology phases and results is designed through an application in order to embed the code and make it user-friendly. The reason for this it to create a tool more accessible to researchers from different fields that may be interested in the results.

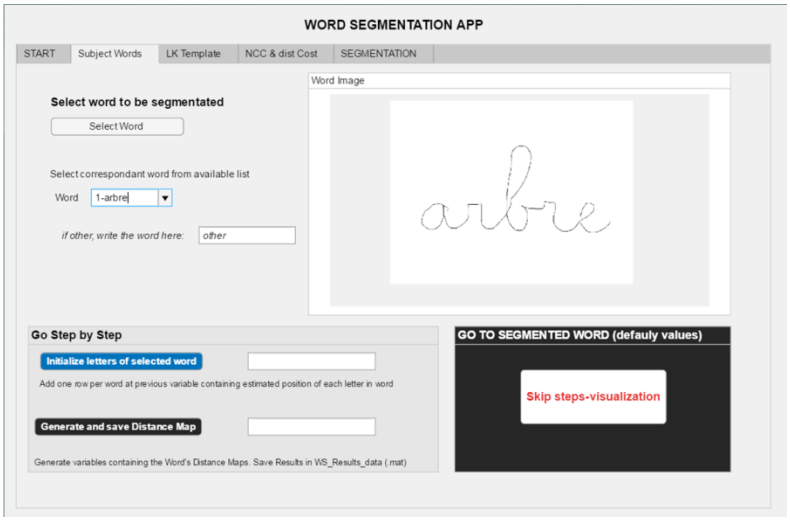


Figure 36: Abstract of the GUI of the program for Word Segmentation

The reason to choose Matlab support for this method development remains on the wide set of predefined functions that suit our work and the vast amount of available toolboxes for specific details. However, compared to other options, it has also been crucial for this choice our accessibility to this software (Student Version) and our familiarization with it. We consider moving the algorithm to other programming languages if it is convenient at some point.

A user-guide of the program application is included in Appendix F. In it, we relate each button and control in the GUI to the different steps of the algorithm described in this chapter with two objectives: to serve as instructions of use for researchers interested in using our program and also to facilitate the understanding of the implementation for further development of this work.

4.9 CONCLUSION

In this chapter we have presented the pipeline of the whole method. In our method, we start with a binary image of a handwritten word, reconstructed from samples recorded with digital SmartPads during an experiment conducted to children who are learning to write, and we compute its distance transform.

Our main contribution is to combine template matching techniques with graph theory to identify the different letters in the word.

First, we warp reference letter templates using the Lucas-Kanade optimization algorithm to make them more suitable to the word's handwriting style. We determine the set of possible locations of the word letters by considering local translation of the warped templates on the image. In a second step, we build a graph that connects consecutive letters based on these possible letter identifications. The cost of edges in the graph account for Normalized Cross Correlation matching between each translated template and the word and the distance between the end and the beginning of consecutive letters.

We finally obtain the segmentation solution on the graph representation by means of the Dijkstra shortest-path algorithm. This solution optimizes the two involved costs in the graph edges: similarity of the handwritten word with warped templates and spatial coherence of the word.

In the previous sections, we have reviewed the main steps required to implement this methodology, from the initial binary images to the visualization of the graph-segmentation solution.

5 ALGORITHM KEY ELEMENTS

In this chapter we develop with more detail the discussion of the main phases of our method: Distance Map generation, Template Warping with LK optimization algorithm and, finally, the graph-based formulation for the segmentation problem.

5.1 DISTANCE MAP

All the algorithm key steps of this work are implemented on DM of binary images; thus its construction has a big impact on the final result. For this reason we include on overview of the procedure that we have followed to build the images DM in our work.

5.1.1 Mathematical morphology

The construction of the distance map is heavily linked to the input image being used. This means that, in order to obtain good results of the distance transform it is indispensable to work on the image pre-processing and observe the effects of the possible modifications on the final result.

For this reason, multiple strategies have been tested as part of the image processing step previous to the distance map building.

The original image is binary (only contains 0 or 1 values) and is created in such a way that the word contour is represented in black (binary 0 element) and the background is white (binary 1 element). But in order to construct the distance map, is preferable that the contour is built by the zero-element so that it can be considered an obstacle, and all non-zero elements represent the distance to these obstacle pixels. Consequently, from this starting situation, multiple approaches can be followed. It is possible to first apply mathematical morphology to improve the image aspect and then invert the range to draw the contour in white; or the opposite strategy can be followed by first inverting the image values and then working on image enhancement.

Moreover, in the image morphology stage, there are many options that can be tested too; such as different structural elements to perform the operations or the operations to apply as well. With respect to the structural element, it is possible to use different geometrical shapes such as disks, circles, squares… and different sizes of each option are also valid. With respect to the operators, mainly different sequences of dilation, erosion, (opening and closing) can be combined. Of course, the selected sequence is strongly related to the kind of image used.

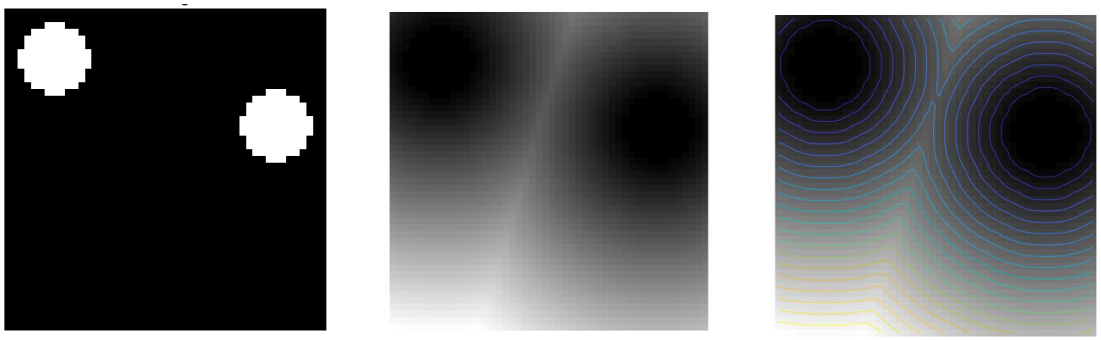
The distance map is built by using a defined Matlab function (*bwdist)*. However, this function offers free parameters to tune it. Besides, the results that this function provides have been tested by creating manually the DT of some images and comparing the results.

a) <u>Simple case</u>

To gain control over the result of the DT, we use a simple image for the first trials. In this case, we are using a binary image composed by two separated circles on a square background. We draw in white the circles so that we can consider them as obstacles to build the DM.

To do this, we use by the moment Euclidean distance as the selected metrics. With this image, we would expect the distance map to show concentric circles around the white circles, and distorted lines in the middle and lower part of the image, where the circles generated by both

circles coincide. It has to be also easily appreciated the region constituted by the set of pixels that are equidistant to both circles.



(a) (b) (c)

Figure 37: Distance Map generation on a simple binary image with Euclidean metrics. (a) Binary image, (b) Distance Map visualization, (c) Distance Map with plotted Contour lines that connect same-greyscale levels

Contour lines are added to the image to improve visualization of common greyscale levels among pixels. Notice also that the DM labels the distance to the denominated "obstacle pixels"; and this test image has relative small dimensions; therefore, the maximum distances within any pixel in the background and the circles is also low. As a result, the greyscale levels of all pixels are low and it is difficult to appreciate the distance transform. To solve this, the image is plotted with adjusted greyscale levels to cover the whole range [0 255] (normalization). However, this is only used to improve visualization and understanding of the image; it does not change the real greyscale levels of the stored variable.

b) Application to words

The next step is to validate the construction of the DM for the images being used in this work: handwritten letters and words. Moreover, for this analysis, variations on mathematical morphology are tested as well.

i) Mathematical morphology

The first strategy presented is the one in which the original image is firstly inverted and then its appearance is improved by different operators. The second applicable strategy is to firstly work on the image appearance and then invert it.

To complete the evaluation of the best way to generate the input image for the DT, it is also convenient to change the structural element and see its effects. Multiple tests have been performed varying all possible parameters. For this work, we decide to first invert the image and then improve its aspect, so that it is more direct to visualize the effects of the changes into the Distance Map construction. By observing the results of this approach, it can be appreciated that the dilation operator offers the most robust word contour. Moreover, being a basic operation, leaves more freedom to adjust the final desired shape, while the closing operation, being a compound operation (dilation followed by erosion), makes it more complex. It has been tested that by changing the structural elements, we obtain different image results.

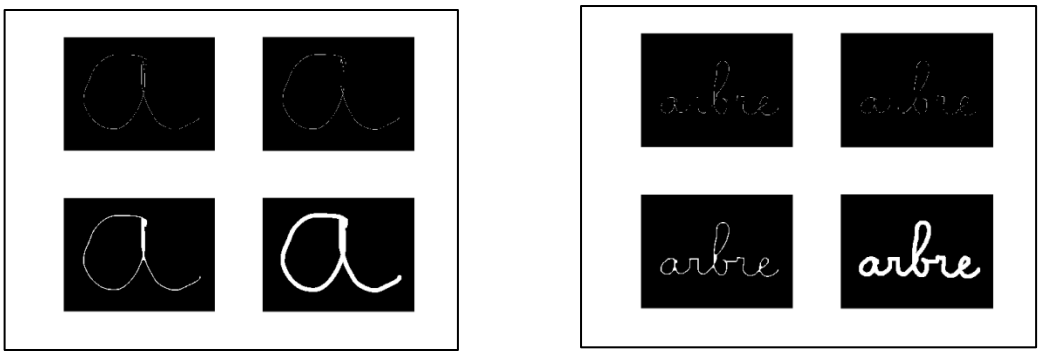


Figure 38: Result of multiple strategies of mathematical morphology and inversion applied to the original binary image of the letter "a" (left) and the word "arbre" (right)

ii) Distance metrics

The previous steps have been done in order to generate the optimal image for the DT. Nevertheless, some other parameters can be selected in this next phase.

An important fact to mention is that, because the input image is constituted by white contour (binary value 1) and black background (binary value 0); the distance transform presents an inverted aspect, by setting the contour to black (0 value, meaning no distance to the original white pixels) and the background to different grey levels, each one representing the distance to the original white pixels. The output image is no longer binary; its range starts at zero value (excluding the null value, which is reserved to the contour) and reaches a highest value that depends on the image size. However, it is possible to adjust the output range, as it is discussed later in this section. The main parameter to select to create the distance map is the distance metrics used.

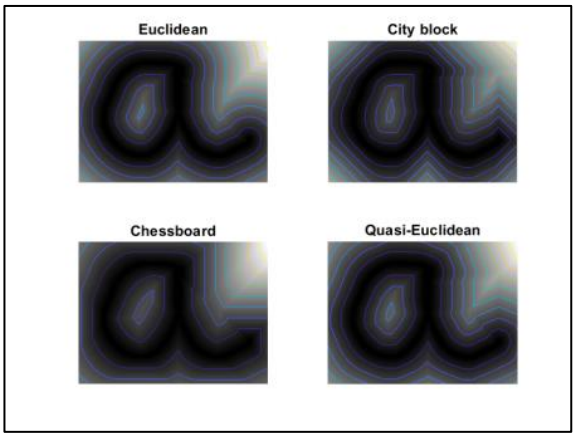


Figure 39: Comparison of the Distance Map of the letter "a" image constructed by different distance metrics. Contour lines are plotted on the distance map to connect same grey-scale levels

As before, contour lines are plotted on the Distance Map images to improve results visualization. Each one of these lines connects the pixels with the same greylevel. The smoother output function is given by the Euclidean distance. For this reason, it is selected as the option to use in this work.

The selected choices have been tested in new images to validate the results obtained through these steps. Smoothness can be appreciated by representing the DM as 3D graphs, where the height of each pixel is given by its greyscale level, which also varies in colour scale.

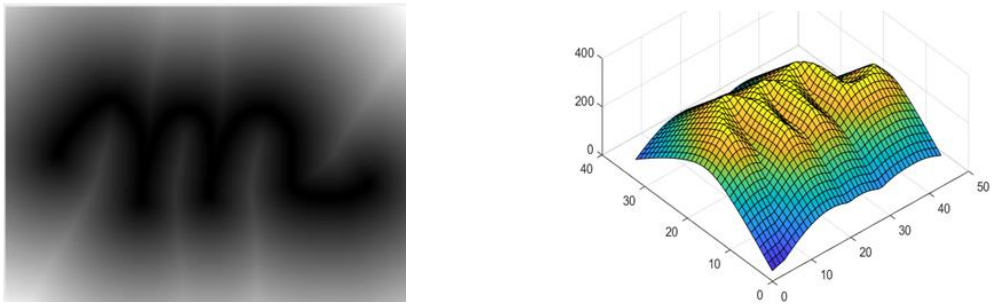


Figure 40: Distance Transform of the letter "m". Visualization of the greyscale results in 2D (left) and 3D (right)

iii) Distance Map tuning

Once the distance map is built, there still remain some more open options, such as modifying the greyscale range, inverting the results, increase the difference between consecutive pixels, applying a weighting function… The choice of these does not longer remain on appearance criteria but on performance in the next stages of the methodology, such as Template Matching. Although these options are included in this section, the choice has been done based on multiple analyses along the method. We have tested the effect of the different options of tuning the distance map on the whole algorithm, especially on the two different template matching stages. The different results obtained by different distance map configurations have been compared to conclude the best selection for the proposed methodology.

Some of the variants of the basic Distance Transform that have been considered are:

i) Raw Distance Map Result: Range from zero (excluded) to any possible value.
ii) Modified range: Between [0, 255]. This can be achieved by two different approaches:
    a. Take the raw result and saturate the greylevel at 255.
    b. Normalize all vales in the desired range
iii) Modified range: Between [0, 1]. By normalization.
iv) Evaluated Distance map: By construction of a weighting function, that gives more importance to the pixels close to the contour. The construction of the filter is done using the original word/letter image.
v) Inverted distance map: By inverting the values of all the pixels within the same range.
vi) Quadratic Distance Map: By using the square value of the pixel's label. Then, normalize the image within the range [0, 255].

We highlight that whatever the tuning option being tested is, it is always applied to all Distance Transforms, i.e., to the word images and the letter images.

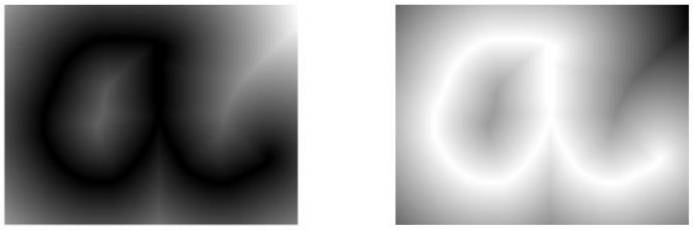


Figure 41: Examples of the Distance Map construction of the letter "a". Left: Standard linear version. Right: Inverted version.

After multiple tests with different configurations of free-parameters, it is concluded that the distance map that leads to better matching results is option (vi); in which the quadratic version of the distance map is used. As explained in previous chapter, this variation over the standard DT leads to a smoother penalization for matching close to the optimal position; while it strongly penalizes matching attempts of very low accuracy.

This decision is also validated in Chapter 6, where we present numerical quantification of the increase in the results quality when the quadratic version of the distance Map is used instead of the linear standard version (option (i)). More specifically, accuracy on word segmentations improves a 14,12 % with this modification. More details about the justification of this choice can be found in Section 6.2.

5.2 TEMPLATE WARPING THROUGH LUCAS KANADE OPTIMIZATION ALGORITHM

The initial templates that we use for letter identification in this work come from the images of the alphabet letters written in a very regular way. We have observed that the handwriting style from the templates is very different form the children's handwriting that we find in the word images. For this reason, we consider appropriate to warp the templates to make them suitable to the children's letters within the word being analysed. This justifies that the desired output of this phase is the set of warped templates. These templates are later used for a new template matching phase with local translation and NCC.

As mentioned in Chapter 2, Lucas Kanade method is an optimization algorithm that can not only be used for optical flow computations but also in template matching due the similarity of the working conditions of these two fields: In both cases it is necessary to identify corresponding objects between two images. In the first application, the two images are consecutive frames and this identification is done with the objective to compute the relative movement between them; while in the template matching application one of the images works as a template of an object instead of a consecutive image in time, and that object has to be identified to know its exact location in the whole image.

To apply the Lucas-Kanade algorithm for template matching, it is necessary to provide an initialization of the estimated location of the object accurate enough so that it converges to the optimal position. If it is an inconvenient to have an accurate initialization given the nature of the problem being analysed, it is also possible to provide multiple initializations to the algorithm to maximize probabilities of finding the global optima. The two alternatives are combined in this work: we use the estimate initialization based on the word length and the number of letter that has been described previously; but moreover, some variations of this estimate are also generated and used.

We recall that the Lucas Kanade method for template matching minimizes a sum of squared error function (SSD) in a Gauss-Newton gradient descent non-linear optimization process. The sum of squared error (SSD) function is defined between the template and the Image Window of the warped source image below the template, for each iteration.

It is applied as an iterative algorithm because it uses first order Taylor approximation of the objective function; thus the way to reach the optima is by iterating until convergence of the

parameters. In other words, in each iteration the algorithm computes the necessary image warping to minimize the dissimilarity between the template and the current image window, based on the previous iteration information to update the warping parameters vector.

As the algorithm is presented, it is the source image that is warped in each iteration until convergence. But it is the set of templates that has to be modified to increase its similitude with children's handwriting. To solve this, at the end of the Lucas Kanade method execution, the warping parameter vectors are inverted; and the inverse transform is applied to the templates. The new templates are stored in the program memory, still as Matlab variables (.mat) containing the greyscale level of all pixels.

5.2.1 Basic Template Matching implementation

Template matching techniques are used with Lucas Kanade algorithm to get a set of distorted templates in such a way that the letter contour behind these new images is more similar to the children's handwriting in the word under consideration. In this phase, template matching is linked to SSD similarity measure, and the allowed the degrees of freedom for the image warping are translation and scaling. More details of this approach are discussed below.

This first phase is considered to be one of the key elements of the method. But moreover, template matching is also used in this work in a later phase to assess the similarity between these new templates and the word image for a finer alignment. In this second phase, it is implemented based on correlation approaches and only translation is allowed.

It can be concluded then that template matching as a whole is a core part of this work. Its implementation and validity have been deeply analysed for the SSD LK variant; but most of the reached conclusions are also applicable to NCC template matching. This section firstly presents the discussion that has led us to define these two different template matching phases and to understand its common implementation issues. And secondly gives a more detailed clarification of the Lucas Kanade method implementation, which has implicated complex analyses and validation phases to reach its final implementation.

a) <u>Basic implementation of the template matching</u>

Multiple template matching techniques are being used nowadays. To better understand the performance of the different approaches we have carried out a comparison between them. With this analysis we also aim to validate the state of the art methods applicability to our work.

In first place, we compare the two main branches of the Template Matching techniques: SSD-base and Correlation-based, concretely under NCC. For this, we design a pair of images to be used as Source image and template, respectively, and solve the registration problem with the two method implementations and translational DOFs.

The complete test can be found in the Appendix A, including the images that we have designed, the implementation details and the results that we have obtained.

After the analysis, we can conclude that the two strategies being compared are able to solve the registration problem: The optimal solution is obtained under SSD and NCC. However, in the SSD approach, the results manifold is remarkably smooth with a clear peak for the optimal solution. On the other hand, the results manifold under the NCC approach is more abrupt; with a

bigger slope between consecutive positions, giving place to a larger region where the results indicate a null matching.

This difference is taken into account for the final choice; but at this point what is interesting is that we can validate the applicability of these techniques in the current work, and more specifically, we can validate our implementation. In next sections, the specific techniques to be implemented in our method are further discussed.

b) General considerations for template matching implementation

After general template matching implementation is validated, it is applied to the particular context of this work: images of handwritten words and letters. The first attempts to apply the previous algorithm with the word images were not successful and the results were not looking reliable at all: They were sometimes very abrupt, provided very inaccurate solutions or did not converge. However the previous simple test had proven that the algorithm was applicable and that its implementation on the program was right.

We have analysed the main points that may be a significant difference on the algorithm characteristics between the two sets of images, (i) simple test images and (ii) handwriting distance maps; and we have found out the most substantial differences are the image dimensions and the greyscale range in the template and image window being compared at every step. This has led us to the decision of deeply analyse multiple options of image *normalization* and *mean-subtraction*. The tests are applied to the same pair of source image and template under three template matching approaches: CC, NCC and SSD. The selected pair of images belongs to the reference data of this work, hence handwriting images DM. The complete list of strategies that we have compared and the detailed analysis, with the corresponding results, can be found in Appendix B.

The conclusion of this analysis is, first of all, that it is necessary to be aware of the details in the algorithm implementation because its impact on the solution is significant. And more particularly, that the best strategy depends on the similarity expression used for template matching.

In a more specific level, we conclude that performance on Template Matching improves drastically if the mean-subtraction of the image function is executed locally, only for the IW being compared with the template. This means that this step has to be applied within the iterative part of the algorithm the template and the Image window. Under this strategy, normalization of the output values does not have a significant impact; however, this depends on the specific technique being used.

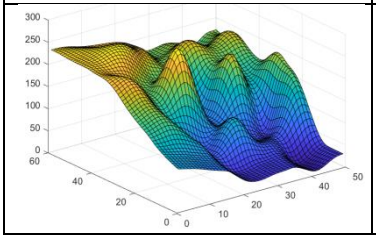
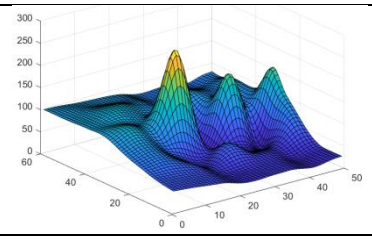
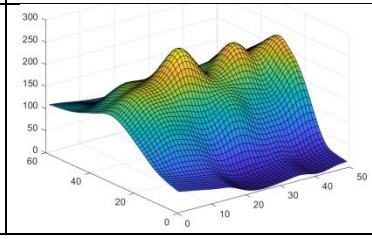
CC	NCC	SSD
		

Table 4: Results manifold with local mean-subtraction of the images with three different approaches: CC, NCC and SSD

The conclusion of this particular analysis has been essential for the method development, particularly, for the LK algorithm implementation. The impact of choosing a correct support to estimate the sum during LK optimization is decisive on the global algorithm's performance. Besides, it has made us gain control on template matching techniques.

The selection of the most appropriate technique to proceed is done according to further analysis to compare the three matching assessment approaches. However, this same analysis also gives us a clue for the assessment method selection, because it proves that the available options respond differently to image variations; being NCC more robust to greyscale changes than CC or SSD.

5.2.2 Template matching techniques comparison: SSD, CC and NCC

Using the conclusion of the previous chapter, it is possible to now proceed to the deep comparison between the multiple template matching approaches to select the most appropriate to use in this work. In consequence, in the following algorithm test, the selected normalization and centring strategy is applied: mean-subtraction of the Image Window and non-normalization.

A priori, we think of an optimization version of template matching for template's warping; in particular, with LK algorithm. In consequence, we consider applying template matching with SSD for this first purpose. However, the results of this analysis are necessary to determine the viability of using this approach in our context (SSD on DM images) and to validate if it is actually the most favourable way to proceed.

The same template matching problem has been solved under three different approaches: CC, NCC and SSD. As said, the reason why we do this analysis is to gain understanding of the different techniques, select the most appropriate for our work and validate its implementation. The complete analysis and the results are in Appendix C.

Some clear advantages have been observed of using NCC approach instead of CC or SSD. On the one hand, the output manifold is smoother; hence convergence under gradient descent techniques is improved. Moreover, the value for good matchings is strongly favoured; while it also penalizes bad matching. This means that this approach clearly reflects the matching accuracy of the images being compared.

It is also necessary to point out the smoothness of the output surface under SSD, even more remarkable than smoothness for the NCC case. This may be a feature of interest if the single-best position is not the main objective but to obtain an accurate enough result, with a given threshold, and making convergence the easiest possible.

In particular in this work, the SSD approach seems appropriate for a preliminary step in which it is important to guarantee an approximate matching between the template and a region of the image, accurate enough. In this case, sub-optimal solutions must not be strongly penalized. After this preliminary phase, the template alignment is refined in a more complete template matching phase where the value of the matching assessment is relevant to make the final template placement choice. This second objective can be achieved by applying template matching under the NCC. Moreover, not using an optimization method, NCC is useful to store all results within a search area.

5.2.3 Template Matching with Lucas Kanade optimization algorithm: general implementation

The previous analyses are crucial to demonstrate that template matching techniques can be applied in our method; and besides, that the results that they provide in this context are valid and reliable. Finally, we can proceed with the key step of our method in which we apply Template Matching for template Warping. For this purpose, we use the Lucas-Kanade algorithm, which is an optimization method for template matching under SSD, and has already been introduced (5.2).

The generalized objective function being minimized is based on the SSD between two elements. Ehen the algorithm is applied to template matching, its expression becomes:

$$e(\boldsymbol{x}) = \sum_{x,y} \left[i\big(\boldsymbol{W}(x,y;\boldsymbol{p})\big) - t(x,y) \right]^2$$

where $\boldsymbol{p}$ is the parameter vector of the image transform that is applied.

It has been also demonstrated in chapter 2 that the final expression to update the parameters at each iteration is:

$$\Delta\boldsymbol{p} = \boldsymbol{H}^{-1} \left[\sum_{x,y} \left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}} \right)^T \big(\mathrm{t(x,y)} - i\big(\boldsymbol{W}([x,y];\, \boldsymbol{p}^*)\big) \big) \right]$$

This method has been used for many authors in the template matching field. However, given the particular characteristics of the working context in this project, we considered necessary to test the method and verify that it is applicable for the current purposes. This algorithm validation is developed starting from the simplest cases and, progressively complexity of the images and released DOFs has been increased until complete validation of our method for the template warping phase. We have finally applied the method to the real context, in which the letter templates and the images of handwritten words are used.

Notice that in Section 5.2.1, we have determined that the best practice is to work with locally mean-subtracted images and non-normalized. However, we consider this as an implementation issue (covered in the Appendix B) and use the generalized expressions for the template matching techniques.

The complete procedure of validation of the method can be found in Appendix D. Mathematical manipulation to obtain the expression for the algorithm implementation is also included in the same Appendix.

As a conclusion of this analysis we want to mention the remarkable impact of two main factors in the final algorithm performance: initialization of the warping parameters vector $\boldsymbol{p}$ and specific characteristics for the images, such as grey-scale smoothness or working values. These two features are reflected in the image gradients. In consequence, we appreciate the importance of an appropriate image pre-processing and we also repeat the importance of selecting the appropriate support to apply the iterative steps of the algorithm, i.e., within the IW instead of the whole image.

5.2.3.1 LK algorithm implementation for template warping

After we have validated all intermediate tests, we proceed to apply Lucas Kanade algorithm with the 4 selected DOFS $\{T_1, T_2, K_1, K_2\}$ to the word analyis, in the same way that it is applied in our method for word segmentation.

a) First analysis area determination

The LK optimization algorithm is used to warp the templates, thus this phase is repeated as many times as the number of letters in the word. For each execution, two main elements are required: The image serving as source image and the template, although it is also decisive to have a proper initialization of the warping parameter vector. The image to be used as source image is not the complete DM of the word, only a smaller region where the letter is assumed to belong. In this work, we denominate this region as Analysis Area (Section 2.3.1).

We use a specific Analysis Area per each letter in the word. The set of Analysis Areas is determined based on the estimate position of the letters that has already been detailed, which is based on a graphical grid that depends on the number and shape of letters that constitute the word being segmented (Section 4.4.2). The Analysis Areas of consecutive letters are always overlapped, so that any part of the centre of the source image is left unexplored.

To select the AA of all letters in a word, a fixed magnitude is added along the associated template dimensions. This amount of longitude is set proportional to the original initialization's letter width. The letter width estimation is used as a reference because it is a common magnitude for all letters in a word, quite adapted to each case dimensions, to avoid the inconvenient of different words having different number of letters, and, consequently, different global dimensions. This extra longitude that is added along each side of the templates is distributed in such a way that the estimated position is centred in it, if possible. If the global limits of the image do not allow this, it is adapted.

It is also advantageous to make the AAs depend on the templates dimensions to guarantee that the analysis region is bigger than the template, so that there are degrees of freedom between both images.

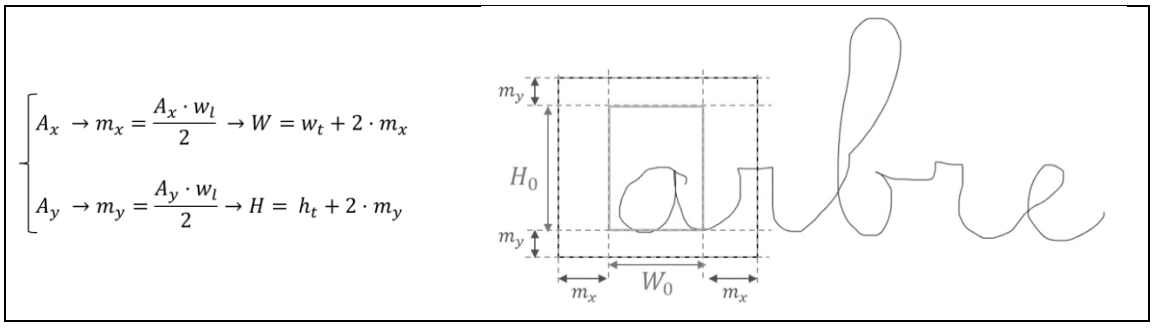


Figure 42: Example of the determination of the Analysis Area for the letter "a" in "arbre" through its mathematical expression (left) and the graphical representation (right)

The Search Windows is the set of pixels where the template can be positioned to be compared with a region of the image (Section 2.3.1). From this definition it is demonstrated that the SW is a subset of the Analysis Area, which has been previously defined too. The Analysis Area dimensions vary for each letter in each word because they depend on the template dimensions, as mentioned. However, by adding a fix magnitude along each axis to generate the different Analysis Areas in a word, it is possible to generate SWs that do not depend on the template

sizes. This is an advantage that permits us to work with the same SW dimensions for all letters in a word.

In the method that we have developed, the definition of the Analysis Area is a free parameter for the user (through two slides to determine the parameters A_x and A_y respectively). Default values are given as a general value that works for most cases but it is left editable in case that the obtained results want to be improved. This is thought to be convenient being this an initial phase in the letters recognition, in which the given initialization may not be accurate.

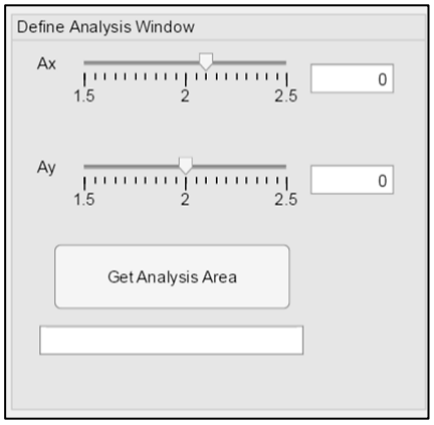


Figure 43: Image from our segmentation program GUI where the user can modify the parameters Ax and Ay for AA determination

b) <u>Lucas Kanade algorithm implementation</u>

Other than the AA and the raw templates, we need to have a proper initialization. This initialization is based on the initial letters estimate position; which has also been used to define the Analysis Areas.

At first, the initialization of the letter's position is described by the position of the upper-left corner of the box that encloses the letter with respect to the reference point of the source image (also upper-left corner); as well as its width and height. But it is also important to remember that only a cropped region of the Source Image, the Analysis Area, is used for template matching. This leads to the fact that the first essential step is to move the initialization coordinates from the global reference frame ($\{u_{0,in}, v_{0,in}\}_{S.I.}$ to the AA reference frame $\{u_{0,in}, v_{0,in}\}_{A.A.}$, by subtracting the relative position of the AA with respect to the Source image $\{x_{0,AA}, y_{0,AA}\}_{S.I.}$.

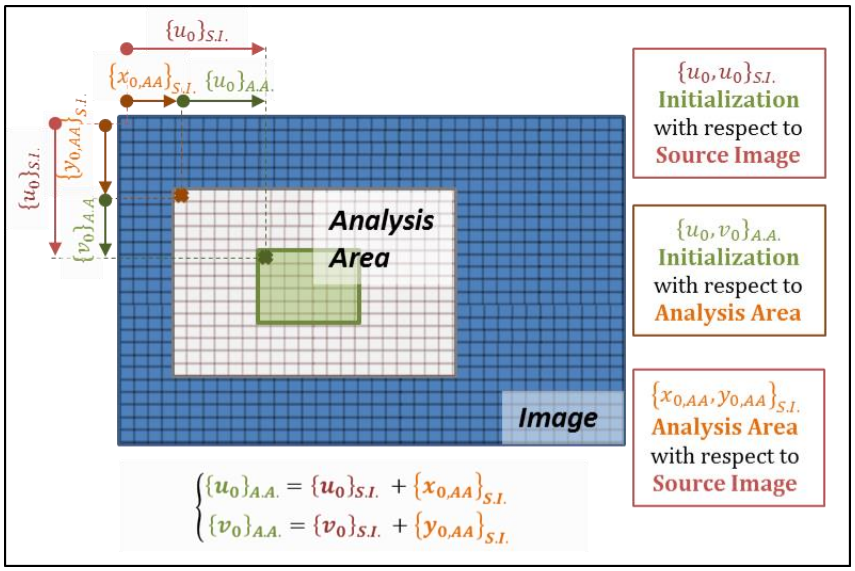


Figure 44: Graphical representation of the relative position between the letter's estimate position initialization, the letter's AA and the Source Image

Hence, in the LK algorithm implementation for template matching the working reference frame is the frame associated to the Analysis Area $\{x, y\}_{A.A.} = \{u_{0,in}, v_{0,in}\}_{A.A.} + \{u, v\}_{A.A.}$

Notice that (u, v) is used for the template's upper-left corner placement on the image; what means that (u, v) is variable during the algorithm; while (x, y) refers to offset between fixed reference points.

The results proved that LK algorithm converges but it is sensitive to the initialization of the image warping. Moreover, while the results manifold for the cases where only scaling DOFs are released presents a clear peak for the optimal solution, the results manifold or the only-translational case is smoother, with multiple local minima. This also indicates that the mixing of the 4 DOFS that we use makes the implementation even more complex given the existence of multiple local optima. This conclusion has made us coma up with the decision to apply the same LK algorithm with multiple variations on the initial parameters vectors so that the manifold exploration is extended and possibility of finding the global optimum increases. This strategy has already been mentioned in Chapter 4, and the complete list of variations applied on the first initializations is in Appendix D.

So, over the described initialization, some variants are applied in order to maximize the sampling and make this step more robust. These variants on the initialization of the process are described by:

$$\boldsymbol{p}_0 = [u_0 \quad v_0 \quad 0 \quad 0] \rightarrow \boldsymbol{p} = \boldsymbol{p}_0 + \begin{cases} [\Delta T_{1,1} & \Delta T_{2,1} & \Delta K_{1,1} & \Delta K_{1,1}] \\ [\Delta T_{1,2} & \Delta T_{2,2} & \Delta K_{1,2} & \Delta K_{1,2}] \\ & \dots & & \\ [\Delta T_{1,\mathrm{n}} & \Delta T_{2,\mathrm{n}} & \Delta K_{1,\mathrm{n}} & \Delta K_{1,\mathrm{n}}] \end{cases}$$

These various initial parameter vectors are all used as algorithm initializations to obtain multiple solutions of the final distortion and position of the template within the AA, what means that multiple solutions are found after the algorithm convergence for each letter, each one being the minima of one algorithm's execution, i.e., multiple local minima. The objective if this sampling is to maximize possibilities to find the optimal distortion.

It can be noticed that to define image warping, we work with 4 Degrees of Freedom. Therefore, the parameter vector corresponds to:

$$\boldsymbol{p} = [T_1 \quad T_2 \quad K_1 \quad K_2]$$

This parameter vector contains the information of the associate warping to be performed at each iteration on the original source image; and therefore, the final warping that represents the optimal solution to the template matching algorithm. In this work, the warping is always applied to the AA image. However, the result is at the end inverted and applied to the template to get the distorted set of templates for next steps in the program.

In summary, over each one of the variants of the initializations of the parameters vector of a letter in the word we apply the iterative algorithm until convergence is reached; or, oppositely, until the limits that we set with respect to maximum of distortion or number of iterations are reached. Each iteration starts with the previous (or initial) warping applied to the image and computes the update of the current parameters to be applied in the next iteration. The expression for the parameters updates with the 4 DOFs that we are considering becomes:

$$\Delta\boldsymbol{p} = \begin{bmatrix} \Delta T_1 \\ \Delta T_2 \\ \Delta K_1 \\ \Delta K_2 \end{bmatrix} = \begin{bmatrix} \sum {i_x}^2 & \sum i_x i_y & \sum i_x i_x x & \sum i_x i_y y \\ \sum i_y i_x & \sum {i_y}^2 & \sum i_y i_x x & \sum i_y i_y y \\ \sum i_x x i_x & \sum i_x x i_y & \sum i_x x i_x x & \sum i_x x i_y y \\ \sum i_y y i_x & \sum i_y y i_y & \sum i_y y i_x x & \sum i_y y i_y y \end{bmatrix}^{-1} \begin{bmatrix} -\sum i_x (t-i) \\ -\sum i_y (t-i) \\ -\sum i_x x (t-i) \\ -\sum i_y y (t-i) \end{bmatrix}$$

and $\boldsymbol{p} = (T_1, T_2, K_1, K_2)$ can be updated in the form $\boldsymbol{p} = \boldsymbol{p} + \Delta\boldsymbol{p}$, iteratively.

Once again, notice that mean-subtraction and normalized issued are not included in the general expression. Template and image functions can be pre-processed as decided in the algorithm implementation (Appendix B). The previous expression is given in its general form.

Given the complexity of the problem and the variability in the results that we observe in this step, we consider the possibility of using multiple solutions of template warping from this step and select the final best result in the next step together with the template alignment with NCC. By this we mean that we consider keeping the set of local minima that we obtain after all different initializations and build an augmented graph representation of the problem that includes multiple distorted templates per letter. The graph-based segmentation would also serve to disambiguate the most convenient distortion for the templates. We discuss this procedure in Chapter 7.

5.3 GRAPH-BASED WORD SEGMENTATION

The segmentation of the handwritten words into letters is done taking into account the constraints that each letter identification imposes on the previous and posterior letters and the matching assessment that quantifies how probable it that a letter is identified in a particular position. So, all the constraints and relevant information have to be reflected in a graph-based representation that enables us to solve the problem by means of a shortest-path algorithm that optimizes the connection joint between letters.

The formalism of this graph has been introduced in Chapter 4. The nodes of this graph correspond to specific translation of the warped letter templates and the edges correspond to the connection of consecutive letters for different template positions. A direct inference of this definition is that the nodes correspond to image pixels in which the warped templates can be placed as a consequence of local translation; and more specifically, for each letter, this set of allowed pixels defines a SW. For this reason in this work we refer to the nodes that belong to a letter's layer in the graph as the pixels in this letter's SW.

Nonetheless, this graph is the conceptual representation of the problem and its implementation in the algorithm is done in matrix-form. As a consequence, graph nodes are represented through the matrix columns and rows, so all the pixels contained in the SW of each letter are reshaped into vector form. The final matrix is block-diagonal given the fact that only the connection between consecutive letters is allowed and in a unidirectional way. Data associated to the graph edges constitutes the body of the blocks in the global problem matrix; subsequently, the matrix contains information related to template matching and inter-letters distance.

The final objective is to connect the source and the sink nodes, as presented in chapter 4. Given the characteristics of the matrix construction, to do this, the algorithm has to select one mid-node from each block of the block-diagonal matrix. This means that one (and only one) pixel of each letter's SW is selected to achieve all letters chaining.

To obtain coherent results, one of the assumptions to build the matrix is that the order of the word letters is known. This means that the order of the multiple SWs that have been used in template matching for comparison is fixed. From the previous statement it can also be derived that all the letters that conform a word have to be included in the segmentation problem representation.

All the matrix spaces from nodes that may not be connected are set to infinity value. This is the way to implement the two main restrictions: non-consecutive letters can not be connected and consecutive letter have to be connected in the appropriate order.

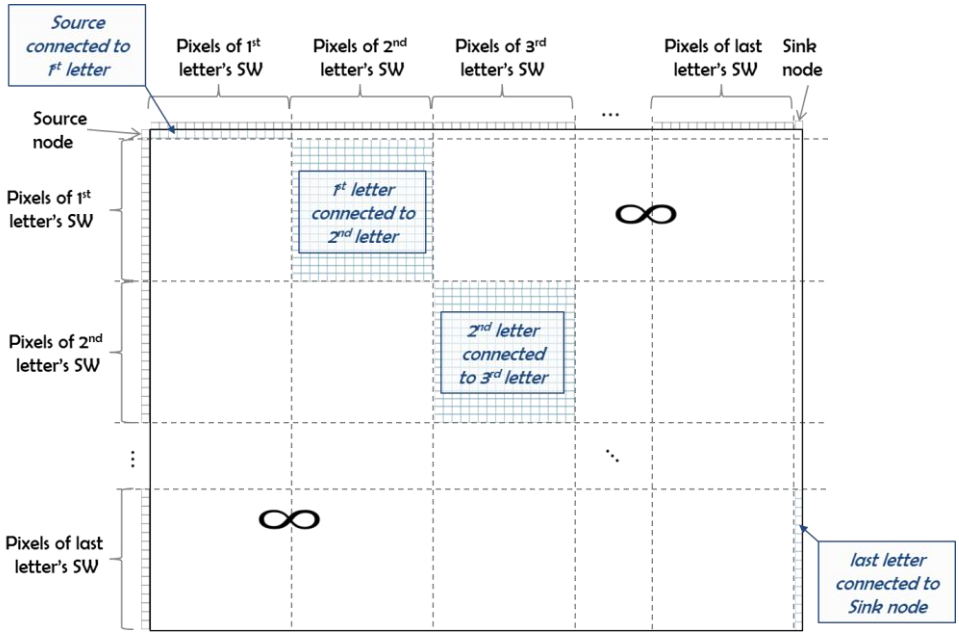


Figure 45: Matrix representation of the graph-based segmentation problem

The following section describes in detail the steps that we have developed to convert this raw data from previous phases into a cost matrix that it can be used to select word segmentation by means of a shortest path algorithm.

We have introduced in chapter 4 that the edges cost (here the matrix) has two main contributions.

On the one hand, template alignment cost comes from executing a fine alignment of the warped templates on the image using template matching with NCC. Therefore, this cost is an assessment of the similarity between the warped templates and the image for the allowed translation of them.

On the other hand, spatial connection is ruled by the inter-letter distance. Therefore, the second component of the problem representation is based on the distance between consecutive letters in a word depending on the translation of the associated templates (i.e. depending on the pixel where the warped templates are placed).

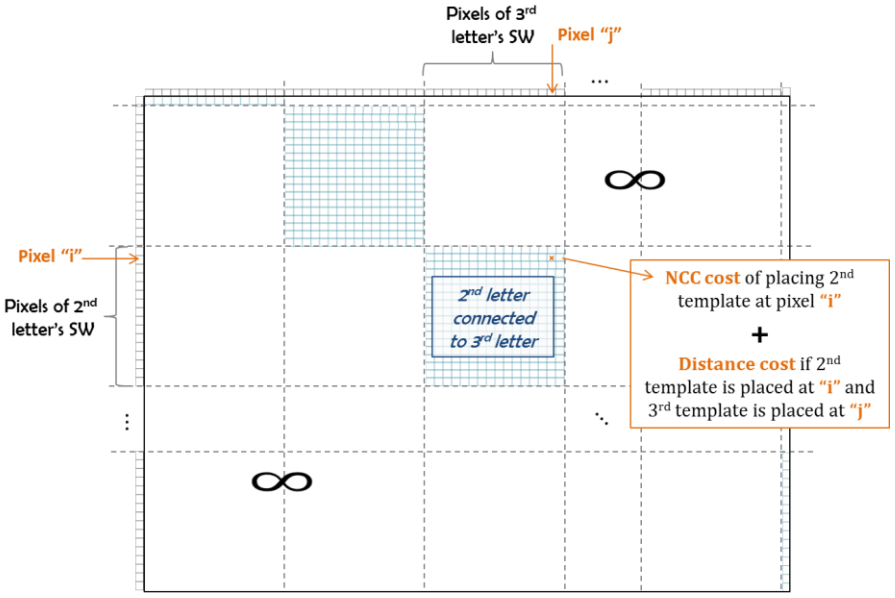


Figure 46: Clarification on the construction of the matrix representation of the graph-based segmentation problem

This cost description has some intrinsic implications. First, it means that the template matching assessment that has been computed by NCC refers to the nodes; i.e., the specific template translation in which the result was obtained. On the other hand, the inter-letter distances are the weights associated to the connecting edges; i.e., connect consecutive letters for their respective templates position.

However, to work following graph theory, it is not convenient to have weights on nodes; instead it is convenient to associate it to the edges. To follow this strategy, the weight associated to a particular pixel is transferred to the edge that leaves that same node. This way, the cost of an edge leaving a node has associated the cost of template matching for the template being placed in the position that the node indicates and besides, the cost of the distance of the connection itself. The NCC results of the template matching of the last letter are transferred as well to the links that connect the last layer's nodes to the sink node. This cost transference from the nodes to the edges is reflected in the implementation of the graph as a matrix (Figure 46).

These two costs are obtained from the raw results of template matching and distance computation, through an evaluation function. The evaluation function for each feature (NCC result or distance) has to make the combination of the two aspects be coherent and representative for the problem. The two cost functions are designed independently but in an iterative process to adjust them; because although they are independent functions, their output costs must be comparable and work in a similar cost-range of values. This is discussed in the following lines, were we present the two stated contributions of the cost separately.

5.3.1 Fine Alignment of Warped Templates: NCC Template Matching

In the graph-based representation of the problem, we retrieve the set of positions in the image over which the warped templates can be translated and compute the matching similarity between these templates and the handwritten word to be segmented. Therefore, the graph-representation includes the matching assessment for the set of possible transformations of each letter, being these transformations the shifting of the warped templates.

Different template matching techniques are nowadays being used for object identification in images. The difference between the different approaches remains on the similarity measure being used. The most common approaches are based on SSD, CC and NCC (Appendix C). Moreover, for each technique some variants can be found depending on the particular algorithm implementation, which can be more or less optimized, resulting in a more or less time-consuming computation. All these approaches have been presented in Chapter 2 and compared in Appendix C. However, the selection of the most appropriate technique to use in this work, and in particular, in this phase, is based on their performance in the specific domain. We have carried out multiple tests to compare the approaches and NCC has proven its efficiency and good performance. The correlation techniques are less sensible to difference grey-scale ranges between the two involved images and other dissimilarities that can take place. The details of these analyses are also included in Appendix E.

In the results, we observe that the NCC objective function manifold clearly shows the optima of the template matching phase (local and global optima); besides, the gradient that leads to these peaks and, in consequence, the whole manifold, are smooth. This means that this method clearly favours the template alignment of the templates on the letters in a word, but does not avoid other solutions to be considered as well. This fact is decisive for the optimization of the joint graph-segmentation of our method. Our final choice is to use NCC for the second phase of template matching in this work.

5.3.1.1 Analysis Area determination

Although Brute Force for translational DOFs is applied and it is not necessary to have any initialization for template matching iterative process, the optimal position obtained with the warped templates is here useful to determine the new AAs.

In the same way as seen before, template matching process is repeated once per each letter in the word; and in each case, the pair of images being compared (warped template and AA) is different. The new set of AAs is selected in such a way that SW dimensions remain constant. To achieve this objective, we define the AA similarly than for the LK step (Section 5.2.3). Nonetheless, this time the proportion being used is not adjustable by the user and they are tighter to the templates.

5.3.1.2 Implementation of Template Matching based on NCC

The distorted templates and the new set of Analysis Areas are used as the input for the fine alignment phase, accomplished with Template Matching under NCC.

To start with, the AA that corresponds to each letter is loaded form available data and so is the associated (distorted) template. For each pair of images, template matching takes place by shifting the template along the 2-dimensional axes of the SW and storing the scalar result of similarity in a new matrix.

The general expression to compute Template Matching through NCC between an image and the template is presented in Chapter 2. We retrieve the mathematical expression of a particular iteration:

$$NCC(u,v) = \frac{\sum_{x,y}\left[i(x,y) - \bar{\imath}_{u,v}\right]\left[t(x-u,y-v) - \bar{t}\right]}{\left(\sum_{x,y}\left[i(x,y) - \bar{\imath}_{u,v}\right]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2\right)^{1/2}}$$

where $i(x,y)$ is the analysis Area, the summation is over the area below the template $t(x,y)$when it is positioned at (u,v). The summation over x,y of the AA below the template is equivalent to the IW for current iteration (Section 2.3.3). Notice that local mean-subtraction of the images is explicitly indicated in the previous expression.

A new matrix is created per each letter in the word to store the NCC results. This matrix has the same dimensions as the letters' SW, and the result of a particular comparison is stored in the matrix position in the same coordinates than the pixel in the SW where the template is placed.

5.3.1.3 NCC cost

As a direct consequence of the NCC similarity measure expression that we have used, the results are interpreted in such a way that high values correspond to good template matching assessments, while low values correspond to poor matchings. It is clear then, that in order to include the results of the matching assessment phase into the cost matrix they have to be modified so that they can be interpreted as a cost.

One of the essential actions to achieve this is to invert the results so that higher values are linked to worse matching quality and low values are linked to best matching cases. However, it is not assumed that tis inversion has to be linear. Instead, some different functions have been tested during the project development to empirically decide the most convenient transform to optimize global segmentation results.

After the testing stage, we found out the function to be used to include the template matching information in the graph-based representation of the problem. This evaluation function is implemented as a piecewise function, to penalize the results according to the specific need of this work. The function and main constraints taking into account for its design are presented below.

i) Interval inversion

As commented, the first step is to invert the results range to convert the reward-assessments into a *cost* interpretation. The initial range where the results belong is $x \in [-1,1]$ and it is converted into $x' \in [0,2]$ in the following way:

$$x' = 2 - (x + 1) = 1 - x$$

ii) Highest values: Linear function

In second place, we have collected data of NCC the results and observed that the best matching for the different image rarely reaches the highest possible qualification as a consequence of the remaining differences between the letters in the word and the templates. Actually, it is important not to penalize all results close to the maxima, because they must be considered for global optimization of word segmentation. For all this, in this work we have empirically determined that the upper 30% of achievable values have to be only slightly penalized because the best solution always belongs to this interval. To implement this decision in the evaluation function, a linear relationship is used to associate a cost to the matching results that belong to this best interval, $x' \in [1.6\ 2]$.

To be comparable to the relative distance's cost, it is set that the low limit of this best-solution interval has to have associated a cost of approximately 2 (see in next section).

$$f_I(x') = m \cdot x' + b$$

$$f_I(x' = 0.6) = m \cdot 0.6 + 0 = 2 \rightarrow m = 3.33$$

$$f_I(x') = \frac{10}{3} \cdot x'; \quad if \quad 1.6 < x' < 2$$

iii) Lowest values: Exponential function

Correlation assessments below the previous bound of 1.6 (in the inverted and translated range) have to be penalized in an increasing and stronger way. To translate this desire into mathematical concepts, an exponential function is used. To work in an appropriate interval of cost-values that is later combined with the distances cost, it looks reasonable to use the function:

$$f_{II}(x') = 5^{(x\prime - a)}$$

To guarantee continuity with the previous function, we impose that $f_{II}(x_l') = f_I(x_l')$, where x_l is the frontier point:

$$f_{II}(x_l' = 0.6) = f_I(x_l') = \frac{10}{3} \cdot x_l' = 2.25$$

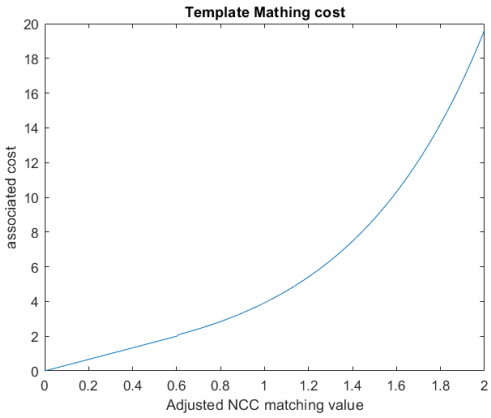
$$f_{II}(x_l' = 0.6) = 5^{(0.6 - a)} = 2.25 \rightarrow a = 0.15$$

Finally, we can determine the second section of the function as:

$$f_{II}(x') = 5^{(x\prime - 0.15)}; \quad if \quad 0 < x' < 1.6$$

iv) Cost evaluation piecewise function

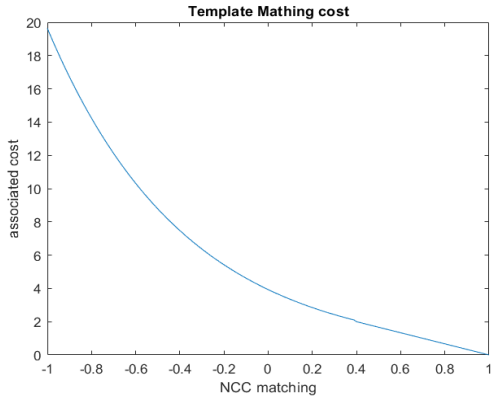
The combination of the two previous functions gives an evaluation piecewise function of the type:



$$f(x = NCC) = \begin{cases} \frac{10}{3} \cdot x'; & if\ \ 1.4 < x' \leq 2 \\ 5^{(x\prime - 0.15)}; & if\ 0 \leq x' \leq 1.4 \end{cases}$$

Figure 47: Graphical representation of the evaluation function on the adjusted NCC results

This evaluation function is applied on the adapted NCC output (inverted and added an offset) to obtain the absolute cost values associated to the letter's matching with the templates. Therefore, to implement this evaluation function in the method using NCC results directly, the function is modified such that:



$$f(x = NCC) = \begin{cases} -\frac{10}{3}x + \frac{10}{3}; & if\ 0.4 \leq x \leq 1 \\ 5^{(0.85 - x)}; & if\ -1 \leq x < 0.4 \end{cases}$$

Figure 48: Graphical representation of the NCC template matching results' evaluation function

5.3.2 Inter-letter distance

In the graph-based representation of the problem, we have included an assessment of the matching for a set letter template transformations, i.e., for the shifting of the warped templates on a specific domain of the word image. To find the combination of letter template transforms that best fit the word globally, we complement the matching assessment with spatial constraints between letters. These spatial constraints are mainly defined by the distance between the end and start of consecutive letters, i.e., the inter-letter distances that is the result of the relative position on the image of the letter's reference points. The concept of inter-letter distance has been introduced in Section 4.6.3, and the reference points Section 4.2.

5.3.2.1 Reference points setting

To compute this inter-letter distance, the algorithm proceeds as follows: It retrieves from memory the reference points for the raw template letters and computes the new vertical and horizontal offsets of these points with respect to the warped templates. For this, the same

warping that has been applied to each template is applied to the coordinates vectors that locate the reference points with respect to its local reference frame.

During template matching stage under NCC, the warped template is vertically and horizontally shifted on the image analysis area. In consequence, the initial and final reference points of a particular letter are shifted simultaneously with the whole template. Therefore, the two offset magnitudes of the two reference points of the template are shifted in the same exact way than the template itself on the SW.

Thanks to this fact it is possible to generate a set of matrices of the same size than the SW storing this data. A particular element of one of these matrices, e.g.$M(i,j)$, contains information of the coordinates of the *start* or *end* reference point of the template if it is placed at that same position on the SW, $SW(i,j)$. To store this spatial information, 4 matrices are required per each letter, which contain, respectively:

- horizontal coordinate of the letter's *left* reference point for all possible template translation on the letter's SW
- vertical coordinate of the letter's right reference point for all possible template translation on the letter's SW
- horizontal coordinate of the letter's right reference point for all possible template translation on the letter's SW
- vertical coordinate of the letter's left reference point for all possible template translation on the letter's SW

Figure 49 shows more specifically the content of these matrixes for a specific letter in word under consideration:

$X_{left,1}$	$X_{left,2}$	$X_{left,3}$	...
$X_{left,k}$	$X_{left,k+1}$	...	
...			
			$X_{left,n}$

$Y_{left,1}$	$Y_{left,2}$	$Y_{left,3}$	...
$Y_{left,k}$	$Y_{left,k+1}$	...	
...			
			$Y_{left,n}$

$X_{right,1}$	$X_{right,2}$	$X_{right,3}$	...
$X_{right,k}$	$X_{right,k+1}$	...	
...			
			$X_{right,n}$

$Y_{right,1}$	$Y_{right,2}$	$Y_{right,3}$	...
$Y_{right,k}$	$Y_{right,k+1}$	...	
...			
			$Y_{right,n}$

- **Figure 49: Graphical clarification of the four matrixes used to store the reference points' coordinate of a letter**

The horizontal and vertical coordinates are converted from local coordinates with respect to the AA reference frame into global coordinates of the whole source image. The local coordinates in the AA are equivalent to the local coordinates in the SW because they share reference point.

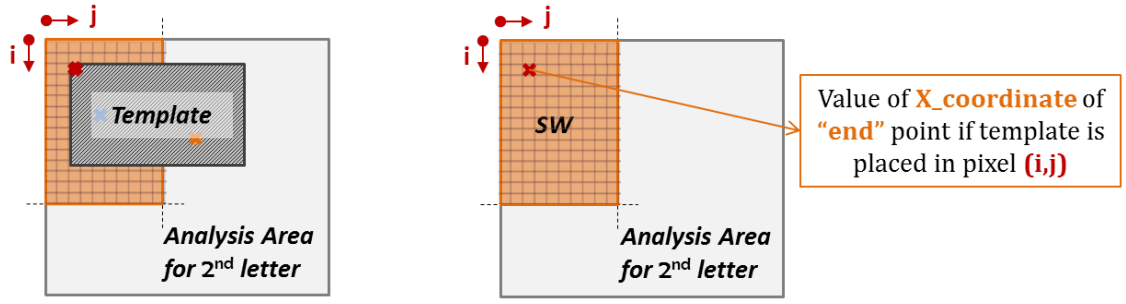


Figure 50: Graphical clarification of the determination of the template reference point per all possible translation of the templates

5.3.2.2 Inter-letters magnitude and sign

Using the reference-points data in the four matrixes created per each letter it is possible to compute the distances between each pair of consecutive letters as a function of the pixel where the two involved templates are placed, i.e., as a function of the local translation of the two involved templates. For this, different distance metrics are tested to select the one that gives best segmentation results. These options are, for example, Euclidean distance, city block distance or only the horizontal projection of the distance.

An important remark is that for the distance evaluation it is convenient to know the sign of the distance magnitude, i.e., to take into account the spatial position of the points being used for distance computation. This means that it must be indicated which of the two points is situated more at the right in the image.

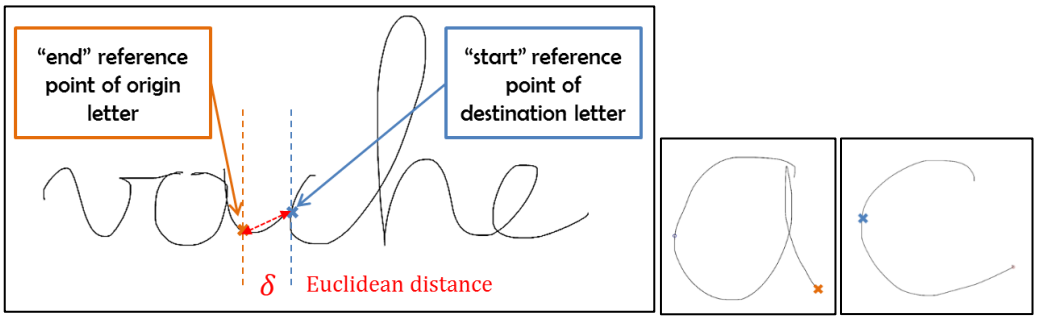


Figure 51: Euclidean link distance between the consecutive letters "a" and "c" in the word "vache" based on the letters' reference points

To better take into account the distance between letters in cursive handwriting, we have determined that the best choice is to use Euclidean distance. However, the sign of the associate horizontal distance between the points is included in the distance magnitude. The formula to compute the inter-letter distance for a particular position of the template "*i*" and a particular position of the template "*i+1*" is:

$$d\left(p_{end,i}, p_{start,i+1}\right) = sign(X_{end,i} - X_{start,i+1}) \cdot \sqrt{(X_{end,i} - X_{start,i+1})^2 + (Y_{end,i} - Y_{start,i+1})^2}$$

The results of the inter-letter distances are stored in a new structure variable to be used for further evaluation of the magnitude.

It can be realised that the previous magnitude can be easily computed with the four previous matrices that contain the information of the coordinates of the reference points.

We recall that these matrixes have the same dimensions of the SW defined for template matching; which are the same per all letters in the word. However, in order to compute the distance between the end point of one letter and the start point of the following one for all possible locations of both templates, this input data has been rearranged in a more useful form.

To clarify how the matrix reshaping is done, we take as example the matrixes with X coordinates information for two consecutive letters, "*i*" and "*i+1*". Naturally, in the program, the same procedure is also applied to the Y coordinates matrixes as well.

$X_{i,right,1}$	$X_{i,right,2}$	$X_{i,right,3}$	...
$X_{i,right,k}$	$X_{i,right,k+1}$	...	
...			
			$X_{i,right,n}$

$X_{i+1,left,1}$	$X_{i+1,left,2}$	$X_{i+1,left,3}$	...
$X_{i+1,left,k}$	$X_{i+1,left,k+1}$	...	
...			
			$X_{i+1,left,n}$

Figure 52: Graphical clarification of the reference points' matrixes reshaping for further inter-letter distance computation

The matrix with information related to the *right* point of the first letter of the pair "*i*", is reshaped into a column vector; and the matrix that contains information about the *left* point of the second letter of the pair, "*i+1*", is reshaped into a row vector. After this, the two vectors are padded by repeating the exact same vector, respectively, in the corresponding directions to end up with two matrixes of the same dimensions.

For the right point of the first letter, it becomes:

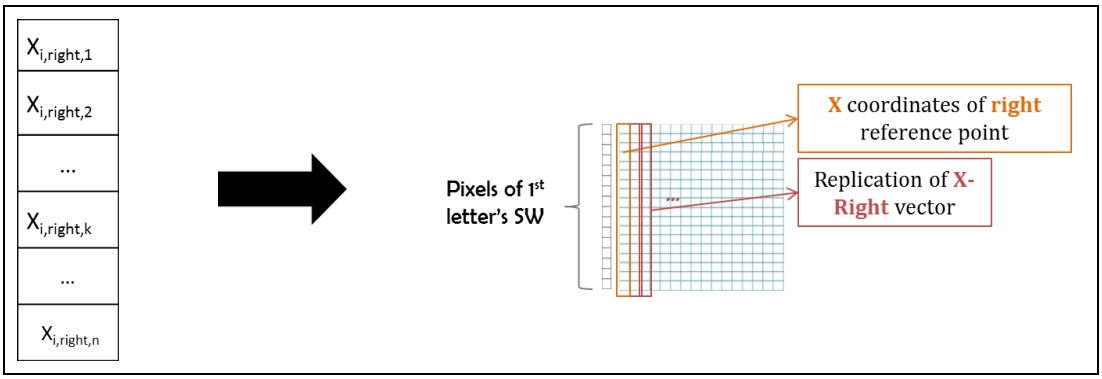


Figure 53: Graphical clarification of the reshaping of the X-coordinates of the right reference point of the first letter

And for the left point of the next letter, it becomes:

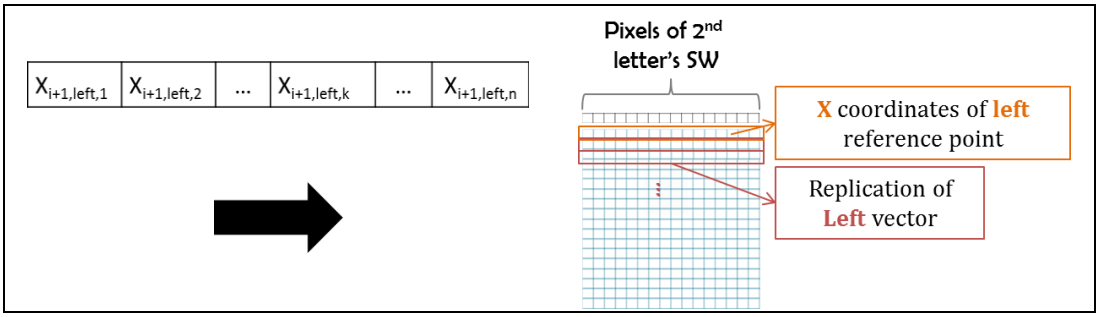


Figure 54: Graphical clarification of the reshaping of the X-coordinates of the left reference point of the second letter

Finally, these two matrixes of same dimensions, together with the similarly reshaped matrixes for Y coordinates, are used for distance computation. The distance result is also stored in a square matrix of the same dimensions.

5.3.2.3 Inter-letters distance cost

a) Distance cost preliminaries

The cost associated to the potential distance between letters depending on where they are identified is the result of an evaluation of the real spatial magnitude. The reason of not using the raw value is to earn control on its contribution on the final result.

In first place, it is necessary to accept a certain distance margin between letters with no penalization. This can be explained by comparing the templates of letters that are used and the complete words handwriting. In cursive handwriting, the different letters are linked using strokes, while the isolated letters in the templates being used do not include this lateral links.

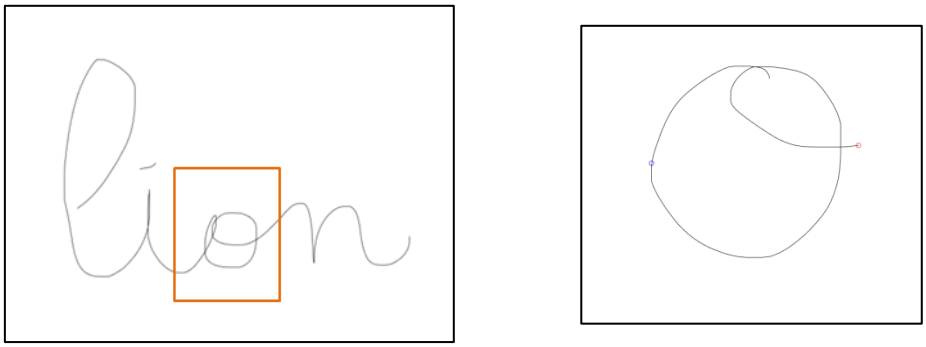


Figure 55: Comparison between the letter "o" in the middle of the word "lion" (left) and isolated (right)

The approximate link distance between letters has been manually measured for a set of handwritten words from the reference word images. Each one of these measures is used as a ratio with respect to the mean letter width so that it not depends on the visualization characteristics or scaling. The mean letter width is the quotient between the total word width and the number of letters that constitute it:

$$\delta = \frac{link\ distance}{letter\ width} = \frac{link\ distance}{word\ width / number\ of\ letters}$$

All the magnitudes being measured and compared in this step are the straight line that connects the ending of one letter and the beginning of the following one, i.e., Euclidean distance.

To measure the distance between letters, it is necessary to know where do the letters start and end. In this step, this is done by hand in an approximated way, using as limits the reference points that are defined for the templates. A big amount of data is collected so that statistics results are not affected by the drawback of the approximation.

Statistical results of a set of $N = 80$ samples are presented in Table 5:

	Value
Max $\overline{\delta}$	0.67
Min $\underline{\delta}$	0.07
Mean $\hat{\delta}$	0.34
Median $\breve{\delta}$	0.33

Table 5: Statistic results of the longitude ratio of links between letters in handwritten words

The fact that the median and the mean values are so close is good to reinforce the decision of using this reference value on the algorithm. For a distance magnitude between letters below the mean value, the associated cost is null. By this decision we accept that consecutive templates might be matched on the word separated by this offset. On the other hand, for distances above this threshold (0.33), the associated cost is increased; and similarly for negative distances; because negative values mean that the reference point of the beginning of a template is found more at the left on the image than the reference point of the ending of the previous letter. Because the reference points belong to the letter contours, this situation is always equivalent to a bad positioning of the templates. In consequence, negative values of distance between points are evaluated with a high cost.

The particular functions to evaluate the link ratios below and above the selected threshold of $\delta = 0.33$ are defined empirically. Some different options have been tested and we have selected the function that has resulted in a better segmentation of the word.

In summary, the previous distance values are expressed as distance ratios and are finally evaluated according to a self-designed function.

b) Distance cost calculation

To implement the previous decisions into a mathematical function to evaluate the relative position between letters, a piecewise function is designed. This function and the constraints that have led to its design are presented below. The output domain of this function is also adjusted to be comparable and coherent with the NCC evaluated results.

Thanks to the fact that magnitudes are divided by letter width estimation, they are less sensitive to changes in the whole word scaling; so this phase is more uniform for all words.

i) Working range of values

Oppositely to the evaluation of NCC, the distance between consecutive letters has already a cost-intrinsic meaning: the furthest the letters are, the highest the distance magnitude is; the worse the solution is. Data from Table 5 is also used in this step to determine the common values obtained when the distance between reference points is computed.

ii) Negative: Exponential function (I)

The reference points of consecutive letters must fulfil the main constraint of spatial relationship that follows: The *right* reference point of one letter must be located in the *X* axis before the *start* point of the following letter. Therefore, negative inter-letter distances have to be particularly penalized because they always correspond to bad matchings of one of the two involved letters (or both). This penalization must lead the graph-based solution to consider other template shifting with also favourable NCC associated cost such that the relative distance has a feasible value.

With this purpose, we use an exponential function with high-value basis:

$$f_I(x) = 20^{abs(x)}; \qquad if\ x < 0$$

Actually, the values below the statistical minima should also be penalized, even being positive. However, statistical data showed a minima value approximately null; thus an interval between 0 and the minima statistical data is omitted.

iii) Low values: Constant function (I)

Also according to Table 5, up to a ratio of 0.33, the link with should not have any associated cost.

$$f_{II}(x) = 1; \qquad if\ 0 < x < 0.3$$

The cost is not set to 0 exactly because perfect NCC matching does not occur; the maximum achieved for each pair of images is usually around 0.7; thus a null NCC cost is never obtained. In consequence, for ratio being in this optimum interval, the cost value is set to constant 1; this way, the best NCC solution and the best distance solution contribute with a similar magnitude so that none of them forces the solution towards a less-desired result.

iv) Medium values: Exponential function (II)

Same statistical results collected in Table 5 show that a maximum link distance of 0.66 is detected. Consequently, penalization of link magnitudes below this threshold must not be very severe either, so that the associate template placements can also be selected as an option: they may be correct. Nonetheless, the probability of an incorrect placement in this interval is much higher than in the previous distance range. For this reason, the penalization is anyway bigger than in the previous scenario.

In this interval, the largest the distance is, the less probable that the selected letters positioning is accurate. To assign the cost according to these facts, the following function is suggested:

$$f_{III}(x) = 2^{(x-a)}; \quad if\ 0.34 < x < 0.66$$

Continuity wants to be guaranteed for the entire cost function for smooth results of the evaluated ratios. To force continuity between these two consecutive graph sections, we proceed as:

$$f_{III}(x_l = 0.34) = f_{II}(x_l) = 1$$

$$f_{III}(x_l = 0.34) = 2^{(0.34-a)} = 1 \rightarrow a = 0.34$$

Finally, the second function that constitutes the distance piecewise function is:

$$f_{III}(x) = 2^{(x-0.34)}; \quad if\ 0.34 < x < 0.66$$

v) High values: Exponential function (III)

We have considered that all the links ratio above the statistically determined maxima of 0.66 belong to poor letter identification. In consequence, we want to strongly increase the associated cost to the magnitudes above this threshold. With this objective, but still to work in coherent range of values given the whole cost context, we suggest the following function:

$$f_{IV}(x) = 10^{(x-a)}; \quad if\ x > 0.66$$

One more time, to guarantee continuity with the previous graph section, we proceed as:

$$f_{IV}(x_l = 0.66) = f_{III}(x_l) = 2^{(x-0.34)} = 1.2483$$

$$f_{IV}(x_l = 0.66) = 10^{(0.66-a)} = 1.2483 \rightarrow a = 0.537$$

Finally, we obtain the last function of the entire evaluation cost function for the distances ratio between letters.

$$f_{IV}(x) = 10^{(x-0.537)}; \quad if\ x > 0.66$$

vi) Piecewise distance cost function

The previous determined functions are used to write the complete piecewise function to evaluate the link distance ratios and get its associate cost. This piecewise function covers all possible values of the distance ratio:

$$f(x = distance\ ratio) = \begin{cases} 20^{abs(x)}; & if\ x < 0 \\ 1; & if\ 0 < x < 0.34 \\ 2^{(x-0.34)}; & if\ 0.34 < x < 0.66 \\ 10^{(x-0.537)}; & if\ x > 0.66 \end{cases}$$

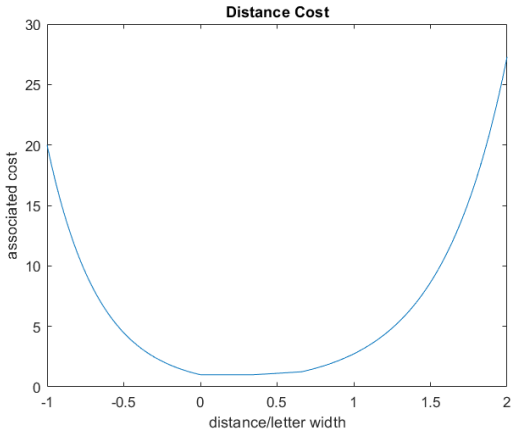


Figure 56: Graphical representation of the inter-letter distances ratio evaluation

5.3.3 Block Diagonal Cost Matrix

The last step to represent the problem in matrix form is to set the total cost results in a convenient matrix structure. This matrix is created by putting together the set of matrix blocks such that the whole matrix is block-diagonal. The amount of blocks that are generated to build the matrix depends on the number of letters of the current word being analysed. The size and content of all these blocks depends on the dimensions of the SW defined during previous steps in the following way:

First Block

The first non-squared block has only one row, connecting the source node to the first letter; and it has as many columns as the total number of pixels in the first letter's SW.

Each of the elements of this cell is referred to the distance cost of connecting the source node to all the possible template placements over the first letter's SW. Although the source node is an auxiliary concept that is used to build the graph, it is associated to the first point of the word's contour line. Then, the first row of the matrix contains the distance between this point and the right reference point of the first letter.

Middle Blocks

The middle blocks are squared cell structures. There is one less squared-block than the total number of letters in the word. Each one of these squared-blocks represents the connection between consecutive letters. Therefore, there is one block to connect the 1st and 2nd letters, another to connect the 2nd and the 3rd… The last squared-block connects the penultimate letter with the last one; what means that there is no squared-block to connect the last letter in the word to anywhere. It can be seen then, that each block connects an *origin* letter to a *destination* letter.

Each one of these blocks' rows refer to the pixels of the *origin* letter's SW; and the blocks' columns refer to the pixels of the *destination* letter's SW. Given the fact that the size of the SW is the same for all letters in a word, these intermediate blocks have squared dimensions. To generate these blocks, the previous cost matrices have to be reshaped.

a) Template matching cost reshape

The template matching results for each letter are stored in a matrix of the size of the associated SW. To generate the global cost matrices, these NCC results stored in matrixes are reshaped into a column vector per each letter.

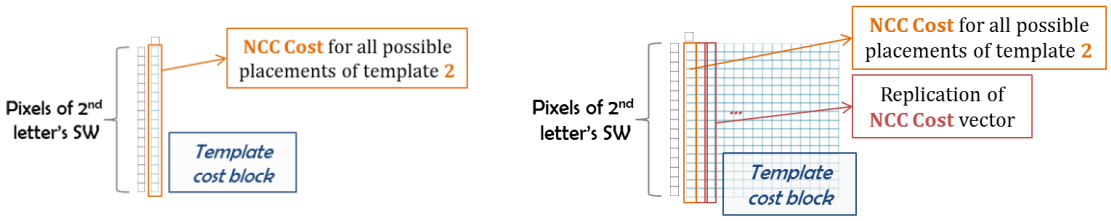


Figure 57: Graphical clarification of the NCC template matching results matrix reshaping

To match the required dimensions for the matrix-blocks this data has to be rearranged. It has been remarked that the results from the template matching stage would actually belong to the nodes but it is transferred to the links in-between nodes. Take the set of edges that represent the situation of a particular placement of a letter's template towards all possible placements of the next letter's template. All these edges must contain the weight of the value of the first template placement and each one must add to this value the cost distance that depends on the destination node of the edge, meaning the positioning of the next template. In other words, the NCC contribution to the total cost of all edges leaving a node must be exactly the same while the distance contribution may be different per each edge depending on the arrival node.

The fact that the template matching cost between a template placement for one letter and all possible placement of the following letters is the same one has to be reflected in the matrix. This is done by replicating the column vector as many times as the size of the search window.

b) Distance cost reshape

On the other hand, the distance cost evaluates the magnitude associated to the origin's template placed at each pixel of origin SW and the destination template placed at each pixel of destination SW; reason why it is represented by an squared-matrix.

The description of how to arrive to this matrix shape and values is covered in previous section. Given the fact that the output of the distance assessment is already a square matrix of the desired dimensions, there is no required posterior reshaping. Output matrix of that previous step can be directly evaluated to obtain its associated cost.

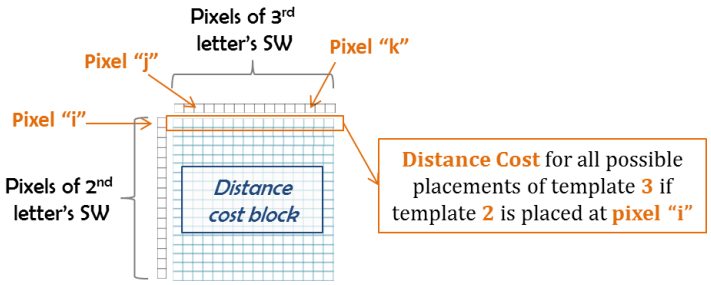


Figure 58: Graphical clarification of the inter-letter distances results matrix reshaping

Last Block

The last non-squared block has only one column, connecting the last letter to the sink node; and it has as many rows as the total number of pixels in the last letter's SW.

Each of the elements of this cell is referred to the template matching cost of the last letter, i.e., the evaluated assessment of placing the template of the last letter in all possible pixels of its SW. It also contains the associated distance of connecting the "end" point of each possibility to the sink node. In this case, it is not necessary to replicate the vector containing this information because all edges are connected to a single node.

Whole cost matrix

The last step is the addition into a single matrix of the evaluated and reshaped results of Template Matching and inter-letters distance. The total cost is then computed as a linear combination of the two magnitudes. Different options are possible to generate this total cost, depending on the weight given to each one of the components. For this reason, during the algorithm development, this has been set as a free parameter to be tuned.

It must be highlighted that at this point, there are multiple design decision that have been taken empirically, by comparing the quality of the results that are obtained in each case. Although this is explained in this work in a sequential way, to develop the program we have worked in an iterative way so that the possible design options of different stages can be combined. In this mode it can be guaranteed that the sampling of possibilities is enlarged, what makes the greedy solution more robust.

5.3.4 Shortest path

After the cost matrix is completely set up, the segmentation problem can be solved. Thanks to the graph-based formulation that we have introduced with this work we can solve the problem by means of a shortest-path algorithm. The shortest-path solution selects, for each letter, the warped template translation (i.e. warped template position) that leads to global maximum assessment. All the costs included in the matrix act as penalties for the word segmentation into letters. Thus, the graph-segmentation optimizes the similarity between the templates and the word image and the spatial coherence imposed by consecutive letters.

The shortest path has to be found between the source node and the sink node, so it is never a choice to select other initial or ending nodes. This is set in this way to force the algorithm to include all the letters of the word in the solution. Thanks to the matrix block-diagonal structure it is necessary to include all the letters in the solution to connect the source and the sink nodes.

To solve this problem we use the Dijkstra algorithm. This method explores the possible paths from the source node towards the sink node by always giving priority to the least cost path.

The shortest path solution contains the source and sink nodes and one middle-node per each one of letters in the word. This solution is returned using the global numeration of all the nodes in the total cost matrix. To decode the solution, each one of the middle nodes is assigned back to its corresponding SW pixel, from the corresponding letter; and the numbering of this pixel is also moved back to the original coordinates of the SW. The relative position between the SW and the original source image is used to convert these local coordinates into global coordinates.

In summary, the algorithm selects the set of pixels of the source image where the corresponding warped templates must be placed for the best letter recognition solution as a result of the joint optimization of template matching and spatial consistency.

5.4 CONCLUSION

There are three main elements that represent the core of our method. For each one, we have presented its exact formulation and also why are they relevant four our purpose. Besides, we have reviewed the challenges that come up with its implementations followed by the solutions that we have encountered to face it.

In particular, in this chapter we have described the template warping phase with Lucas Kanade optimization algorithm. The inclusion of this phase in the method has been one of the main issues through all the development. The reason for this is that this algorithm has been proven to be sensitive to the initialization and also to the specific implementation considerations, such as image pre-processing. As a direct consequence of the weak points of this first phase, we find another key element of the method: the distance transforms and image processing. The reason to include this feature in this chapter is its huge impact on template matching phases, especially, in LK method; other than its intrinsic complexity. As we have observed in this chapter, the results of the LK phase are strongly linked to the characteristics of the images that we use. Thus it has been crucial to analyse these two phases together to optimize the algorithm performance,

Finally, the third element key is the graph-based segmentation, which includes the fine alignment of the warped templates and the spatial constraints between consecutive letters. The main challenge of this stage is to build the graph in such a way that it accurately represents the whole problem and that it constraints the solution for an optimal segmentation.

6 RESULTS VALIDATION

The proposed method has been evaluated on the reference handwriting data and later on the samples of children's handwriting. We have analysed the results in a quantitative way to validate the method and its various phases.

6.1 VALIDATION CRITERIA

The presented work is a methodology for handwritten words segmentation into letters. The result, therefore, is a graphical representation of the start and end points of all the letters in the word. However, to validate this method it is convenient to assess this result numerically and quantify the accuracy that is achieved.

Multiple data sets were provided to us to develop this work. This is an advantage for the validation phase, because the algorithm can be tested on multiple problems and under very different characteristics: different content of the words, different handwriting regularity of quality....

Along this work, this big amount of data has been split in three main groups:

a) Algorithm-Design Data: This data is compounded by the reference handwritten words; i.e., the words written by C. Gosse. This data has been used in the first steps of the algorithm construction, being the most basic and ideal cases where the complexity level is reduced.
b) Algorithm-Tuning Data: Among all the data files that belong to the children's handwriting samples, approximately the 40% has been used to tune the preliminary implementation of the method. In these second set of data, variability in handwriting increases significantly. This fact, together with the decrease in correctness and handwriting skills, adds complexity to the problem. Under these new circumstances, the method can be tuned to be more robust and cover more difficult segmentation cases.
c) Algorithm-Validation Data: Finally, the rest of data that has not been used until now, to test the built algorithm in new cases.

An important remark that we must do at this point is that, from all the available data, those samples that have orthographical errors or were considered to be of anomaly bad quality, have been discarded. A further version of the program could implement new functions to face these cases. This is discussed in Chapter 7.

6.1.1 Quantification methodology

We just mentioned that it is necessary to define a way to assess the obtained results so that the algorithm can be validated and also to compare the functioning under different working conditions. Thus, numerical quantification of the accuracy is desired.

For this, we proceed in the following way: We take all the words that we will use to test the algorithm and manually determine the solution that we expect. This means that we segment the word into letters by hand according to the reference points that we have defined for the letters and trying to be as accurate as possible. Of course, the obtained solution is a good approximation of the optimal one.

In parallel, the images are loaded into the program and it is executed to obtain the digital solution for the same word images. As presented before, the solution is provided as vertical straight lines on the links between letters that indicate the right and left point of each letter.

With the manual and digital solutions, it is possible for us to quantify the method accuracy. For the method solution to be considered as a valid segmentation of a particular letter, its right and left points must be close to the manually determined ones. More specifically, the manual frontier between letters must be contained within the interval delimited by two consecutive digital points, one being the end of one letter and the other, the beginning of the following one. Hence, we associate to each letter a binary variable that is worth *1* if both, right and left manual segmentation points belong the link interval; and *0*, otherwise.

6.2 REFERENCE DATA

The reference data (handwritten words by C. Gosse) has been initially used to design the program; and once the program has been adjusted with new data; the initial data set has been reloaded to test the new version performance on the reference data. The expectations are that the program must still work properly for the reference image sets; even the results may have improved from the initial ones.

The main objective of this section is to analyse the results that the designed program can provide to validate that the implemented solution reaches the required function.

Along the algorithm development, multiple decisions have been taken with respect to the program implementation. Most of them are intrinsic to the different steps digital implementation and have been validated in the previous chapter. However, the use of the quadratic version of Distance Map is still to be validated on complete examples. For this reason, its validation is also included in this chapter.

A part from the global validation and the analysis of the Distance Map variant to be used, this section aims to prove that the different steps included in the method are bringing some improvement. This means that in this section we discuss if the template modification implemented through the Lucas Kanade algorithm in a preliminary template matching has any positive effect on the solutions or it is non-essential. And, finally, we also discuss the other element that represents the core this work: the cost graph representation to solve the segmentation in a shortest-path approach. To validate this, the results that are obtained under the complete method are compared to the results that are obtained if this graph is not constructed; this is, if only template matching steps are executed.

To sum up this, we determine that the main features being tested in the section are: the inclusion of the Lucas Kanade template modification, the graph approach for problem solving and the use of the quadratic version of the distance map.

We point out that the main configuration, which has been designed through all this work, is the one where the quadratic variant of the image distance map of the images is used; the one that includes LK algorithm for templates warping; and the one in which the final graph representation of the problem takes into account the cost contribution from template matching and the cost contribution from the relative position between consecutive letters.

For each comparative approach, one of the main factors has been changed, one at a time, to isolate its effect. Therefore there are 3 approaches for comparison with the reference one. In the first one, the Lucas Kanade preliminary template matching is skipped. This means that the raw templates are used for the NCC template matching directly.

In the second approach, the relative position between letters has been ignored; this means that only the results from template matching are taken into account for word segmentation. This is equivalent to select the best matching for each letter locally, not considering the whole word accuracy (and in consequence, not performing graph-based segmentation). This is done by not including to the total cost matrix the contribution of the distance between letters.

The third approach is done by using the standard version of the DT in which the pixels are simply labelled with the Euclidean value of the distance to the closest contour pixel and saturated at 255. This is applied in both, the word images and the templates.

Approach / Features	**Reference (A)**	**Omitted LK (B)**	**Null Distance Cost (C)**	**Linear DM (D)**
LK	Included	*Omitted*	Included	Included
Cost	NCC + distance	NCC + distance	*NCC*	NCC + distance
DM	Quadratic	Quadratic	Quadratic	*Linear*
Success ratio	**73.67%**	**71.51%**	**59.54%**	**55.19%**

Table 6: Scheme of the 4 algorithm configurations that we compare for the reference data-set

The same words have been loaded to the program under the four different configurations and the segmentation result is obtained. Following the quantification methodology described above, the number of good letter-matches for each word is counted.

6.2.1 Results on reference data

The main configuration showed a mean value of 73.67% of success in letters identification among the tests. This means that approximately the 74% of the letters in the reference words were segmented in an accurate way. However, this percentage is linked to only the 17,87% of words completely good-segmented. This means that the algorithm is able to correctly identify the letters in most of the cases in general; but it rarely provides a whole word good segmented. In most of the cases, at least one of the letters in the word is miss-matched. This fact reinforces the suspect that working in a global-word level is advantageous for the objective accomplishment; but further work is required.

On the other hand, the use of the quadratic DM instead of the simple linear one represents the most significant improvement in the method performance for the reference data. The main configuration achieves a18.48% more of success in letter good-matchings with respect to the approach in which the linear DM used.

In second place, the fact of using the graph-formulation and combine the NCC information with the relative position between letters also has proven to have big impact. The effect of using this approach against the simple approach where distance is not taken into account represents an increase of the 14.13% in successful letter's matching. Actually, without the mixed contribution of costs, the algorithm probability of performing a whole word correct segmentation in letters is below the 5%.

Finally, with the reference words, it hasn't been found a significant effect of including the Lucas Kanade preliminary step matching; the improvement has been quantified as a 2,16%. This fact is consistent with the context of the work, because in these reference tests, the templates and the words are written by the same subject, in a very regular way given the fact that it is a psychological researcher and not one of the children. In consequence, variability between the letters is very scarce, what makes the Lucas Kanade template warping not to have significant impact; oppositely, templates are barely modified.

This comparison between the 4 different approaches can be better appreciated in the following image.

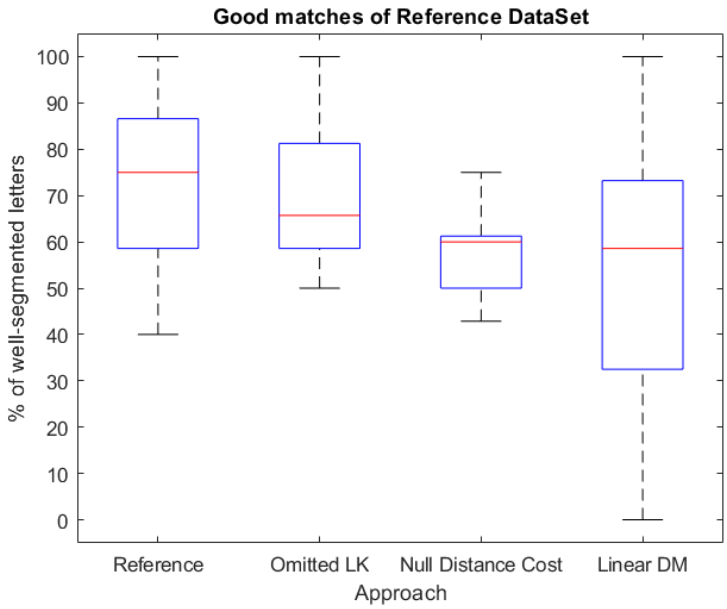


Figure 59: Boxplot graph of the results' statistics of word segmentation results under the 4 different strategies

Figure 59 reflects the percentage of letters that have been correctly identified in each word, for various cases. To obtain this information, for each analysed word we count the number of well-segmented letters and express it's as a percentage of total number of letters in the word; thus we get a success ratio. We plot the statistic results as a boxplot diagram, in which it is observable, not only the mean value of good-matches' ratio but also the most common interval of percentage achieved. This second fact also gives us an idea of the solution correctness variability.

As it has been pointed out, the success ratio of words segmentation in letters for approaches (A) and (B) is very similar, not being statistically significant. Nevertheless, a substantial decrease in performance is observed in the mean value of success ratio in approaches (C) and (D) with respect to main approach (A).

We have compared the graphical result of the same words between approaches (A) and (C) in more detail to also analyse the effect of the spatial constraints in a qualitative way. Interestingly, we observe that when we include the inter-letters distance cost, the algorithm systematically displaces the vertical lines as we expect, reducing the letter's overlapping and large link distances. This usually leads to a significant improvement in the result.

Figure 60: Segmentation result of the word "bouche" when spatial coherence is ignored (left) and when it is used to constraint the solution (right)

One more fact that can be appreciated in Figure 59 is that when the liner DT is used, not only the mean value of the success ratio decreases, but also its variability vastly increases. This is not desirable for the program because accuracy would be more difficult to predict and improve. Similarly though, it can be noticed that the graph-based formulation of the problem adds variability to the solution effectiveness, what has to be taken into account for new changes that may be added to the method.

Another remarkable observation of the main configuration is that the failures in letter identification are mostly spread across the words. We do not find many errors gathered in a few words, instead we find few errors spread across many words. Actually, with the program configuration (A) we have a 16.71% of words completely good segmented and an 8.34% of words with more than half of the letters being poorly identified, hence most of the words segmentation accuracy percentage belongs to the 50-73%. This clearly demonstrates our previous remark.

Another conclusion of the previous fact is the advantage that an incorrect letter identification in the word does not have a significant impact on the whole word segmentation. If that was the case, we would find that the percentage of good matches per word is inverted; being mostly below the 50%. In contrast, in most of the words we find few segmentation mistakes and it does not lead to a complete miss-matching of the word. However, we observe that actually, an error in one letter's identification does impact to the identification of its immediately subsequent letter(s), probably due to the spatial constraints. After one or two letters usually the error is counteracted by the graph constraints.

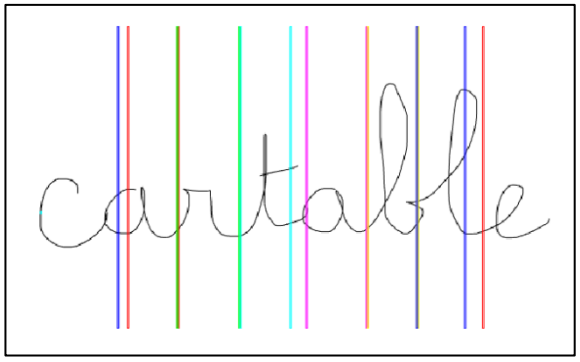


Figure 61: Segmentation of a word where a mistake in one letter's identification leads to a mistake in the subsequent letter

For example, in Figure 61 we can observe that a mistake in the identification of the first letter leads to a mistake in the identification of the subsequent letter. But in the third letter, the error

has been already counteracted and it does not affect the rest of the words' segmentation in letters.

6.3 CHILDREN'S HANDWRITING

We have proceeded in a similar way to analyse the results of word segmentation results on children's handwriting. However, in this case, the quadratic version of DM is always used because it has already been justified. In this new phase, the main objective is to assess performance of the program in its real working context. Consequently, the approaches to compare in this section are:

Approach / Features	**Reference (A)**	**Omitted LK (B)**	**Null Distance Cost (C)**
LK	Included	*Omitted*	Included
Cost	NCC + distance	NCC + distance	*NCC*
Success ratio	**70.76%**	**43.70%**	**52.08%**

Table 7: Scheme of the 4 algorithm configurations that we compare for the children's data set

As before, the same words have been loaded to the program under the three different configurations and the segmentation result is obtained. Following the quantification methodology described above, the number of good letter-matches for each word is counted.

6.3.1 Results on children's handwriting

This main program configuration shows a mean value of the success ratio in letter identification of the 70.76% among the tests. Using this dataset, the variation induced for the different approaches (B) and (C) has also proved to have a relevant effect.

Firstly, the fact of using the graph-based segmentation gives a notable improvement on the results. Concretely, the effect of using this approach (A) against the alternative approach (C) where distance is therefore not taken into account represents an increase of the 18.68% in the success ratio. This statement is very relevant because it justifies the whole work included in this method. The construction of a mixed graph with template matching results and 2D spatial information is the core of this work; with this idea, we introduce a new methodology that is not being used in the state of the art. So, this is the main concept that has to be validated, being a new practise for handwriting character recognition. Hence, this improvement, in fact, is the proof that the proposed methodology is a valid tool on words' segmentation and that the designed strategy for handwriting digitalization is applicable. However, these results are still below the desired threshold of achieved performance. For a better performance of the algorithm, there are still some modifications that might be necessary to add to this work. This is discussed in next chapter.

In second place, an impact is also observable with respect to the inclusion of the Lucas Kanade template warping phase. This improvement has been quantified as a 27.06% more in the success ratio in segmentation. This result is very different from the impact that this same variation has shown for the reference handwriting.

So, this program phase has as a substantial impact in the children's words segmentation but its influence is barely appreciable for the reference words segmentation. This is also consistent with the previous results and the expected scenario. Actually, these two statements lead to the conclusion that the specific templates that are used are also an essential point to have in mind. On the one hand, it demonstrates that the fact that the templates are similar to the words handwritten letters matters for the solution quality. And on the other hand, the children's data tests prove that the isolated action of modifying the template letters shape changes the method efficiency. This opens new horizons of improvement for the work, by more accurately selecting the templates that will be used, or even working with a larger set of templates. This is also discussed in next chapter.

The comparison of the method performance under the different approaches can be better appreciated in the following image. As in the previous case, we represent the results through a success ratio, i.e. the number of well-segmented letters with respect to the number of letters in a word (as a percentage). Moreover, we use the boxplot graph representation to show the results' statistics, in which it is possible to observe the mean value of the segmentation success percentage in the analysed words and also the common interval of good matches; as a measure of variability.

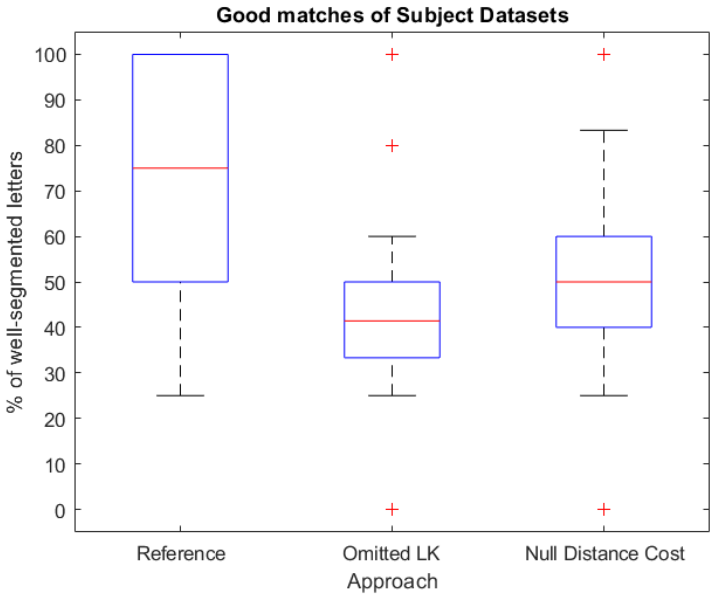


Figure 62: Boxplot graph of the results' statistics of word segmentation results under the 4 different strategies

Figure 62 undoubtedly shows the increase in performance for the works' objective of word segmentation if the templates are modified through a preliminary template matching phase. It is also observable that the success ratio is substantially increased by using this new procedure where word segmentation is formulated as a shortest path problem by reflecting the constraints as costs of a graph that connects all segmentation possibilities.

Nevertheless, variability of the results obtained by the main configuration is still large; fact that is not-desirable for a long-time functioning of the program.

6.4 WEAK POINTS

In the two previous sections of this chapter, we have analysed the general performance of the algorithm for word segmentation in two situations: when we use reference data and when we use children's handwriting. In the reference words, calligraphy is regular, accurate and highly similar to the templates. This means that the segmentation of these words in letters should not represent a big challenge for the program, if we assume that the biggest complexity is added due to handwriting irregularity common in children learning to write. Therefore, we can consider that the errors in the reference data processing are a direct consequence of our method's weak points.

For this reason, we have retrieved the tests of Section 6.2 and we have done a more detailed analysis on the most common mistakes and its causes.

In first place, we find some letters that are written one in a different way when they are isolated than when they are connected to some particular letters. For example, in Figure 63 we can see the letter "r" isolated and the same letter after a "v". Similar link modifications happen to many letters after a "b", "v" or "o". This leads us to the conclusion that more templates may be necessary, including different shapes per letter or including pairs of letters as well (Chapter 7).

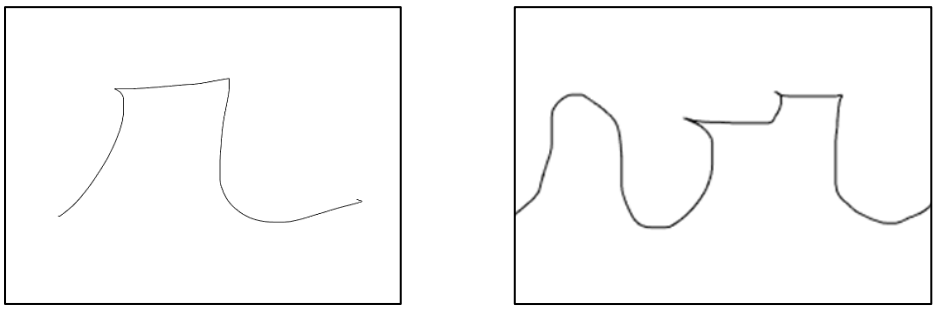


Figure 63: Handwritten letter "r" when it is isolated (left) and when it is connected to a previous "v" (right)

After we have realised this fact, we have reviewed the results of the segmentation test and we have found out that this particular characteristic leads to many mistakes. This is due to the fact that, even if the template has been approximately well-positioned on the image, its reference point is placed in a position away from the word contour. Thus when we search the closest point, it usually finds a wrong match.

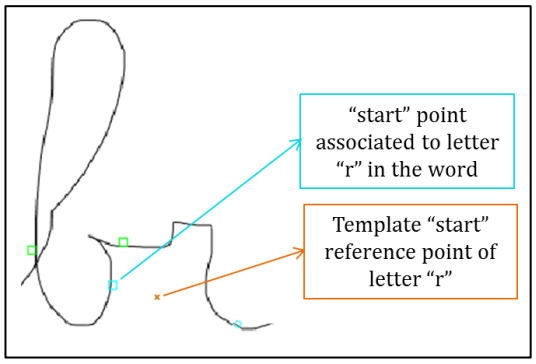


Figure 64: Segmentation error induced by a link modification of the letter "r" when it is written after the letter "b"

The previous issue can be sometimes counteracted if the vertical lines are used on the template reference point to show the result. However, the vertical lines representation other times reduces

accuracy of the result, when the segmentation solution is quite accurate but the points belong to a curved part of the contour line. The vertical lines can also be confusing when a letter crosses its limits in multiple points.

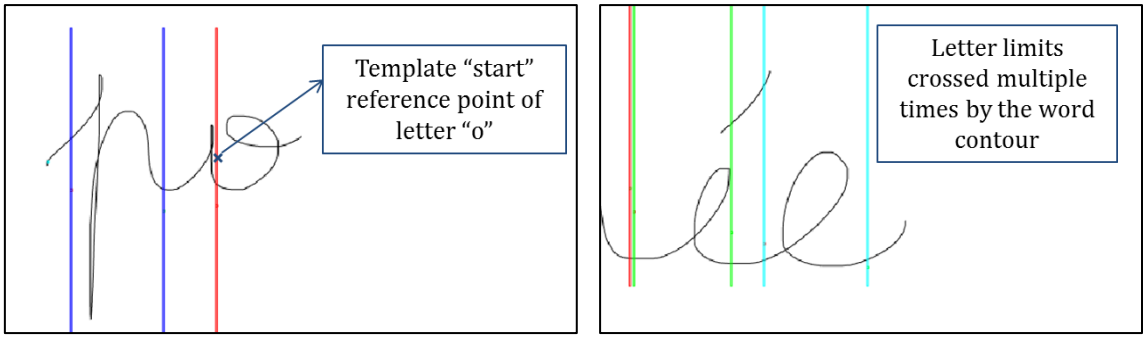


Figure 65: Common segmentation mistakes induced by the solution visualization

Another common mistake that we observe is due to a set of letters whose templates does not have a precedent link, thus its *start* reference point belongs to more deep points in its contour. This is the case of the letters "a", "c", "d", "g", "o", "q".

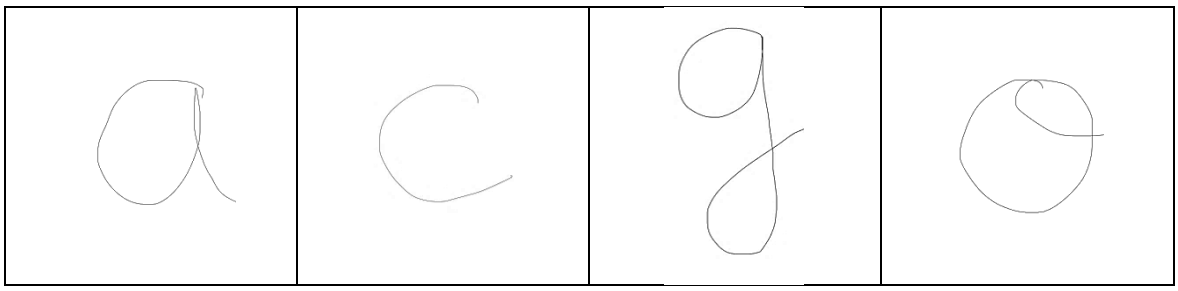


Figure 66: Example of template letters that do not have a precedent link

We observe that in most of the miss-identifications of the letters, one of these letters is involved, because the degree of difficulty of finding the good segmentation point increases. If the warped template does not have the exact size and orientation as the letter in the word, at the end of the algorithm the template's reference point is often placed in some point inside the letter. This interior point is associated to the closest point on the word contour, thus the segmentation solution is found in some mid-point of the letter.

This error strongly penalizes the result with the vertical lines visualization, where the user can see that the limit of the letter actually traverses it. We think that a good strategy to avoid this might be to write all templates with links in both sides; even if they are later compared to initial letters that are written without links in the word. That would only affect to the first letter in a word, which is the easiest case because the first point in the whole word can be used as the left point of the first letter.

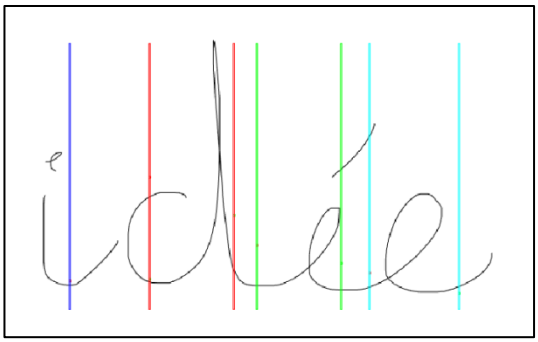


Figure 67: Segmentation error penalized by the solution visualization where the letter "d" in "idée" is vertically traversed with the line that delimits it

The two previous examples also introduce another weak point, which is selecting the most appropriate way to interpret the results from the shortest-path solution of the method.

6.5 CONCLUSION

The previous results prove that the proposed method accomplishes its function of word segmentation. However, the quality of the word segmentation can be argued, given the low mean value of successfully segmented words and the variability of this value when it is applied in its real context.

It is favourable that the effect of the different variations can be demonstrated and quantified. This fact is essential to deeply understand the method and it means that the result is controllable, and, therefore, that it can still be improved. This encourages us to continue this work by deeply analysing these parameters, selecting new possible variations and increasing this way the scope of the work.

Additionally, through a more detailed analysis on the common errors, we gain understanding of the weakest points of the method. This opens new horizons of research and improvement for this word segmentation development. We observe that not all the weight of the method accuracy remains on the algorithm implementation but also on the data that we use and how we process it. Mainly, we can appreciate the importance of the templates, directly connected to the quality of the results.

In conclusion, the validation of the method that we present in this chapter gives us deep understanding on the viability of the method design for the desired purpose and moreover gives us the clues for a significant improvement.

7 CONCLUSIONS AND FURTHER WORK

7.1 GENERAL CONCLUSIONS

In this work, we present a new methodology to face handwriting recognition and digitalization, based in a combination of template matching and graph theory. For this, not only the letter recognition through comparison with templates is considered, but also the constraint from spatial distribution of the text. The main idea behind this methodology is to include all the word information in a graph-based representation of all possible letter connections and select the most appropriate one so that the global segmentation is optimized. This step is executed by means of a shortest-path algorithm on the graph representation of the problem. This work is centred in a particular context that has been accurately described in the first chapters and remarked through this entire document, in which the words to be recognised are known a priori.

It has been pointed out in the previous chapter that this new methodology of using together template matching (NCC) and spatial information in a graph-based formulation has proven its strength. This development can become a powerful tool in the handwriting recognition domain.

Besides, the different steps included in this method have been accurately validated one-by-one; demonstrating also that its inclusion in the complete method is justified.

The main drawback of the method is its still low performance and its variable and low ratio of segmentation success; which is even more penalized when handwriting irregularity increases. However, we have appreciated that we can modify some of the features of the work and substantially vary the quality of the solution obtained. This is an indicative fact of the possibility to control the algorithm and improve the results.

An important conclusion comes from the fact that in this work we are introducing a new procedure for the moment. The idea of converting a handwriting recognition and digitalization problem, in a particular context, into a shortest graph problem is something that hadn't been introduced yet. So what this work truly validates is the fact that this methodology is applicable for segmentation purposes on handwriting text. This work may even be exported onto less restricted contexts, such as including it in a phase of standard handwriting text recognition algorithms.

Despite of further work required to accomplish full functionality of our method, after the general program functioning is validated, we can therefore declare that it is a potentially powerful method; and we consider that this method's efficiency can be improved by working on its weakest points and adding new features.

7.2 FURTHER WORK

One of the weakest points of the current work is the template warping phase, which is implemented through the Lucas Kanade optimization method applied on template matching.

The reason of the weakness is due to the two involved factors: the LK algorithm implementation and the templates that we use. On the one hand, this algorithm strongly depends on the initialization and the pair of DM images being used. Thus, this phase can be further developed to make it more robust and cover more cases. One of the ways to improve it is by releasing more degrees of freedom, such as shearing and rotation. Another improvement on this

optimization of the templates warping has been mentioned along this work, and it consists of the selection of multiple warped templates per letter. We can keep the results of the local minima obtained with LK after each variation on the initialization and include all of them (or the best ones) into the graph-based representation. This graph can then become especially useful to disambiguate between several warped templates taking into account the rest of the constraints.

On the other hand, we have the matter of the templates. Very related to the conclusions on chapter 6, we suggest using a larger set of templates. Lots of options can be considered for this set design; for example, children could be asked to write the alphabet letters individually before proceeding to the words dictation task, so that each child's words could be compared to its own letter templates.

Another option is to use the available data of handwritten material to manually segment the words into letters and look for multiple shapes in which children write each letter. Then, separate these letters to create new templates during the development phase. This way, we could have many different templates per letter. This set of handwritten letters, would not be graphically as correct as the reference ones but they would more similar to the children's handwriting that can be found in new words to be analysed. We also suggest adding to this set of letters more variants, such as pairs of letters that are often written together in a different manner.

Consequently, the Lucas Kanade method for template warping could also be improved by using a wide set of templates per each letter instead of a single one but various initializations. This may, of course, increase computational cost of the method; but given the fact that this is not a key constraint in this work, this is not considered as a relevant inconvenient. Results obtained in this work prove that the impact of this improvement may be especially relevant for the whole method efficiency. The combination of the two previous suggestions involves the LK template warping phase. In this new scenario, each letter is compared to various templates and moreover, for each template, multiple initializations are also tested. Therefore, per each letter and template, multiple local minima are reached. The set of best solutions, based on the optimization function value and, probably, some geometrical aspects, are maintained and included in the following stages.

Not only to improve results, but also to use this method in other fields, new templates could be used as well. For example, the method could be applied for other handwriting styles, such as upper-case or printed-style handwriting; even a mixed style more personalised, by including different ways of writing the same letters…. all this can be achieved by selecting an appropriate set of new templates. Another example is to extrapolate the method for other languages which include different characters as well.

Until now, the templates importance has been remarked, at the same time that suggestions for new work related to this point were given. However, other ideas of further work have been encountered throughout the design of this method.

In first place, there is some available data from the handwriting experiment sessions that we have not used; for example, the time instant in which each sample was recorded. Thus, the first of our suggestions is to use this data together with the present algorithm to refine segmentation. For example, a decrease on writing fluidity could be a signal of a border between letters, if the child is thinking of what letter is next. Or time-logical order of contour points can also be a relevant factor to discard some matching options.

In second place, this method combines letter matching information with spatial distribution data. We believe that other sources of information could also be used to add new constraints that improve the results. One example is to use feature extraction techniques, such as SIFT, which give more clues about where the letters might be recognised. These techniques could be applied in a preliminary stage, on the images (before distance map construction); by previously identifying in the templates those features that make us distinguish the letters between them.

Another option is to combine this method with other existing methods for the handwriting character recognition domain, such as OCR algorithms. Lot of new options can be thought and tested as a source for new constraints that improve the current method.

Also, we have considered the use of Artificial Neural Networks. Its power has been demonstrated by the application of this method in many fields being developed nowadays, among which, we find pattern recognition (classification) and machine learning. Artificial Neural Networks achieve high values of performance for relative low-development cost, thanks to the learning phase. The current problem can be faced with ANN, being a finite number of known elements that can be identified in the words: the alphabet characters. Naturally, we want to preserve and remark the value of the proposed methodology, thus our suggestion is to include ANNs as a preliminary or additional step in the work that might be able to tune the algorithm and substantially improve results.

Finally, there are cases that are left uncover by this work because considered out of scope. Among these, we find the cases were children's handwritten words include orthographical mistakes or particularly poor handwriting performance; being sometimes even hard to recognise the characters by hand. How to face these cases is some work that may be included in the method after the standard results have been improved.

8 APPENDIXES

APPENDIX A: BASIC IMPLEMENTATION OF TEMPLATE MATCHING

A.1. INTRODUCTION

In the state of the art of template matching we find multiple techniques and various modes of implementation of these techniques. To select the most suitable implementation for our work, we carry out various tests on the different techniques that lead us to a deeper understanding of the advantages and disadvantages of the available methods.

To better understand the effects of the possible approaches, we have first built a preliminary example by generating pair of simple images in which we can have notions of the expected result. We construct the image transform of these images to checking this test, not only the general implementation of template matching, but also that it is possible to apply template these techniques for smooth images. Thus we test the algorithm in a similar environment to our work's context and we check if proper results can be obtained.

A source image of low dimensions has been generated, so that the results can be easily understood and extrapolated to the real cases. A region of the same image has been used as a template to ensure that it belongs to the image so that it can be perfectly matched. The simple image to work with and the template are shown below.

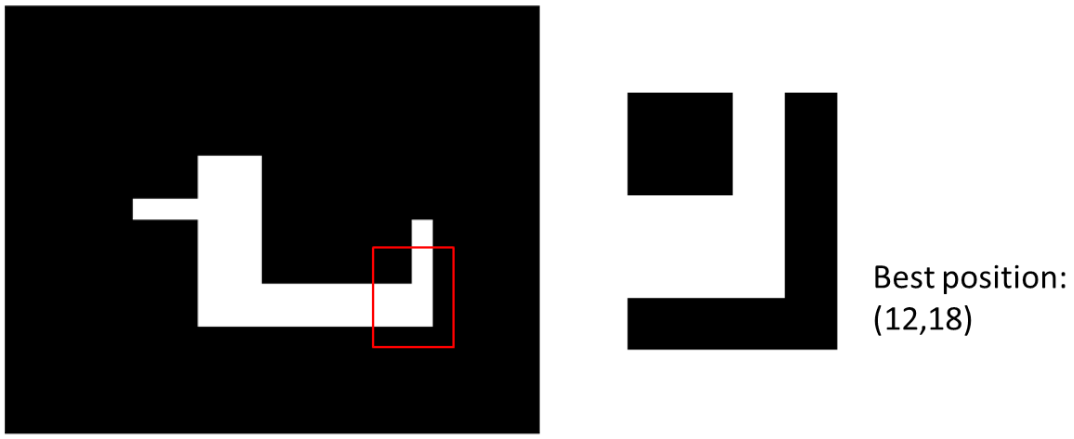


Figure A- 1: Pair of image-template used for the analysis. Left: Source image of dimensions 20x25pixels with a red square indicating the image crop used as a template. Right: Augmented template, of dimensions 5x4pixels and the coordinates for perfect matching in the image

The hypotheses to solve this example are that template size and orientation are the same of the corresponding object in the image, so the template does not have to be resized or rotated for perfect matching; hence only translation is allowed.

The distance transform with Euclidean distance are constructed over the two images, in its Standard linear version. Here we do not need to saturate or normalize the results because the relatively small size of the images results in a low-value distance label for all pixels in the image. The opposite, we have augmented the range of the grey-scale levels reached in the DM images for the visualization examples included in this text.

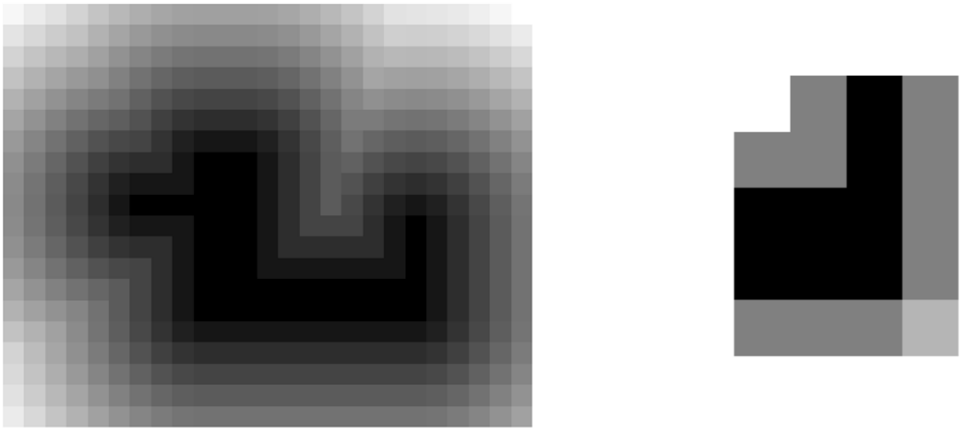


Figure A- 2: Distance transform of the image (left) and the template (right). For better visualization, greyscale has been modified and template is augmented.

Then Search Window is defined. In this case, because of the reduced dimensions of the problem the Analysis Area that we consider is the whole source image. In consequence, the Search window is the one shown in the image below:

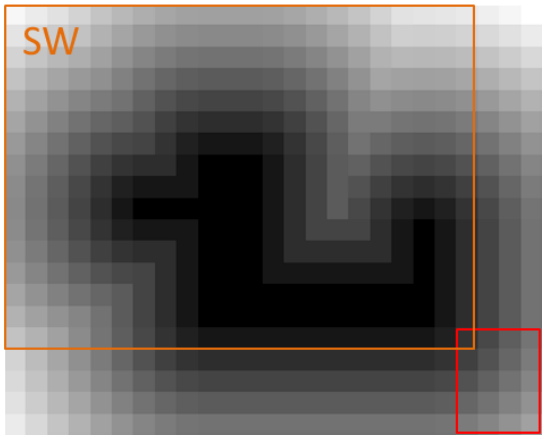


Figure A- 3: Graphical clarification of the Search Window (in orange) of the source image with current template (in red). Greyscale range is adjusted for normalization.

The two main branches to apply template matching are differentiated by the similarity metrics being used. The first is the approach that is based on SSD for matching assessment, while the second branch is the set of techniques that use multiple Cross-correlation variants for the same purpose. For this validation stage, both main approaches are being used and compared with two objectives: Validate each one of the methods in the current context and get some notions of the main differences in accuracy and effectiveness of applying one or the other.

After some research of the state of the art of template matching techniques for different applications, everything points that NCC is probably the most efficient and accurate algorithm. Actually, NCC is the default choice in many applications given the advantages in performance that have been demonstrated [42]. For this reason, in this simple example NCC will be used as a second choice for comparison with SSD. Nevertheless, the best technique to use in the final algorithm is determined with some new tests applied to real cases to guarantee best solution for current work.

A.2. TEMPLATE MATCHING IMPLEMENTATION

So, Template Matching is applied to the simple case to gain understanding of the available techniques. To execute this step, brute Force is used by comparing the template and the image for all possible positions of the template on the image; this is, by placing the reference pixel of the template on all pixels of the search window of the source Image. In each instance, the template is compared to the Image Window below with the same dimensions.

The output of the brute force strategy is the complete set of values that the similarity function can get; this is, the complete matching results manifold.

An important remark of this phase comes from the way we apply mean-subtraction and/or normalization of the images. A detailed analysis if the most appropriate strategy is covered in Appendix B. In this section we only mention strategy that we have applied for the subsequent tests, consisting in non-normalization of the images (keep original greyscale values from DM) and local mean-subtraction. This last term means that we subtract the mean value of the Image window below the template for each template position (u, v).

Squared Sum of Differences

The template is positioned at each pixel of the image Search Window and SSD is computed between the template and the image patch below the template, following the expression:

$$SSD(u,v) = \sum_{x=1}^{N}\sum_{y=1}^{M}[i(x,y) - t(x+u, y+v)]^2$$

where $i(x,y)$ is the image and the sum is over x, y , under the window containing the template $t(x,y)$ positioned at (u,v).

The results are stored in an array of the same dimensions of the Search window, so that every pixel of the results matrix has the value of the SSD assessment for the matching when the template's upper-left corner is placed on that same pixel coordinates of the image. Figure A- 4 clarifies this explanation.

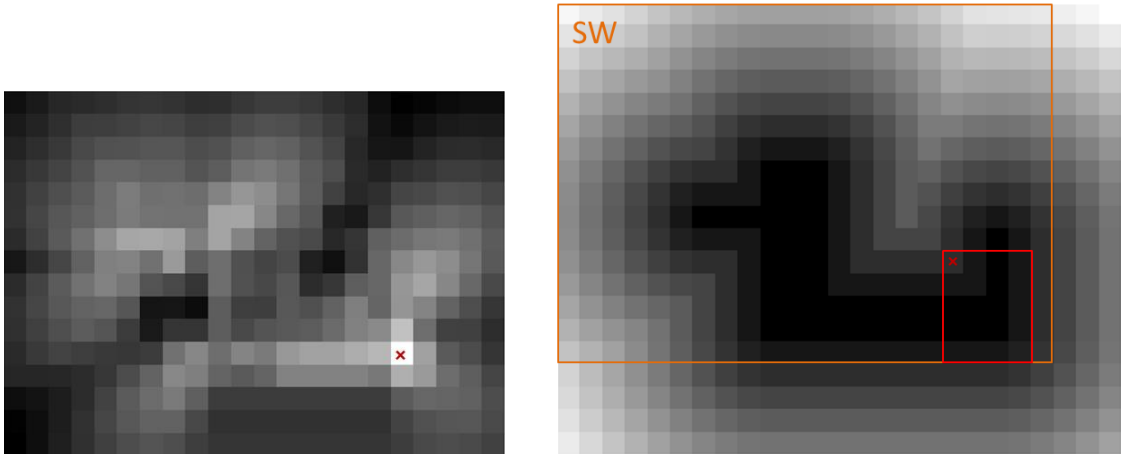


Figure A- 4: SSD template matching results. Left: 2D visualization of the matrix results. Right: Template positioning (red) on the source image for the best obtained solution.

Because SSD is a summation of difference between the images, the minimum value of the results matrix corresponds to the best position for template matching on the image. However in this work the results of SSD are inverted to make them easy to compare with other technique's results. Thus, the higher values (white and lighter grey in Figure A-5) stand for the best results.

The results are plotted in 3D by associating to each pixel a height equal to its grey-level (that corresponds to inverted and normalized SSD results), to see the smoothness of the results manifold.

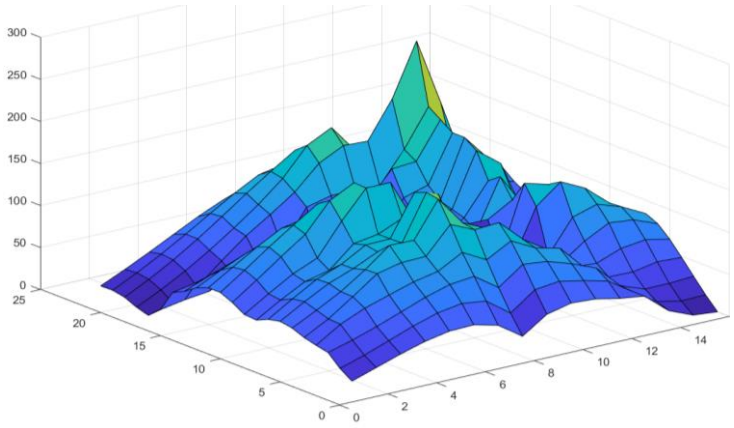


Figure A- 5: 3D visualization of the SSD template matching results in which graph height corresponds to inverted and normalized SSD output.

It can be appreciated in the results image that the achieved solution corresponds to the expected one; actually it can also be noticed that this best solution is seen as a high peak with respect to the rest of possibilities. Concretely, the highest peak of the 3D representation of the results corresponds to the image pixel where the reference point of the template (upper-left corner) must be placed for perfect matching. The whole manifold presents a gradual slope towards the optimum; what makes us think that convergence can be reached if an iterative optimization algorithm is applied on this approach.

Normalized Cross Correlation

The second of the branches that we compare is a correlation-based method; in particular, using Normalized Cross Correlation. Once again, the template is positioned in each pixel of the image Search Window (Figure A- 3) in a Brute Force Search, but this time, NCC is computed between the template and the image patch below the template for every position, following the expression:

$$NCC(u,v) = \frac{\sum_{x,y}[i(x,y)\cdot t(x-u,y-v)]}{(\sum_{x,y} i(x,y)^2 \sum_{x,y} \mathrm{t}(x-u,y-v)^2\ )^{1/2}}$$

The results are again stored in an array of the same dimensions of the Search window, so that every pixel of the results matrix has the value of the NCC assessment for the matching if template's upper-left corner is placed on that same pixel coordinates of the image.

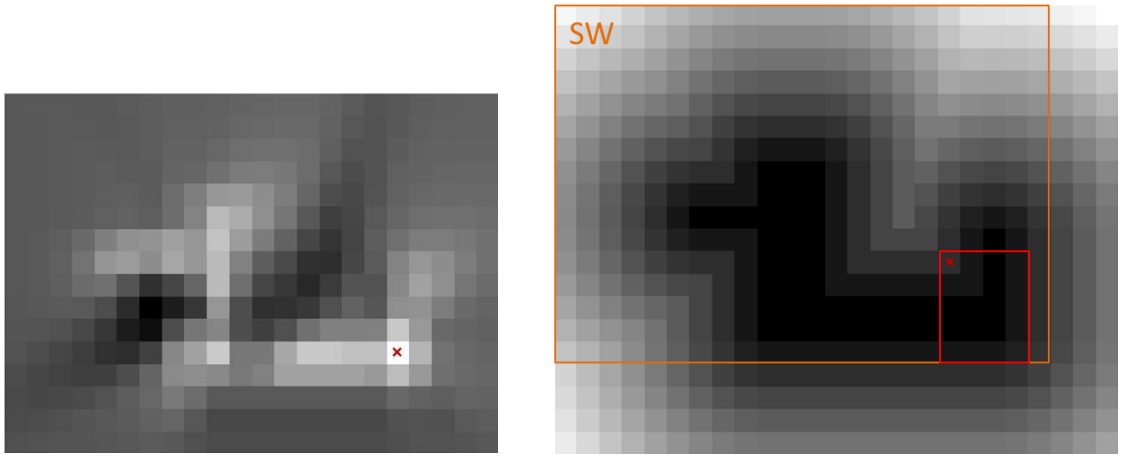


Figure A- 6: NCC template matching results. Left: 2D visualization of the matrix results. Right: Template positioning (red) on the source image for the best obtained solution.

Because NCC assesses matching results belong to the interval $NCC \in [-1, 1]$, being 1 total coincidence, the maximum value of the results matrix corresponds to the best position for template matching on the image. However, the results are adjusted to the range $NCC \in [-255, 255]$ to facilitate comparison with the SSD results.

The results are plotted in 3D by associating the pixel grey-levels (that corresponds to NCC results) to the coordinates height to see the smoothness of the results manifold.

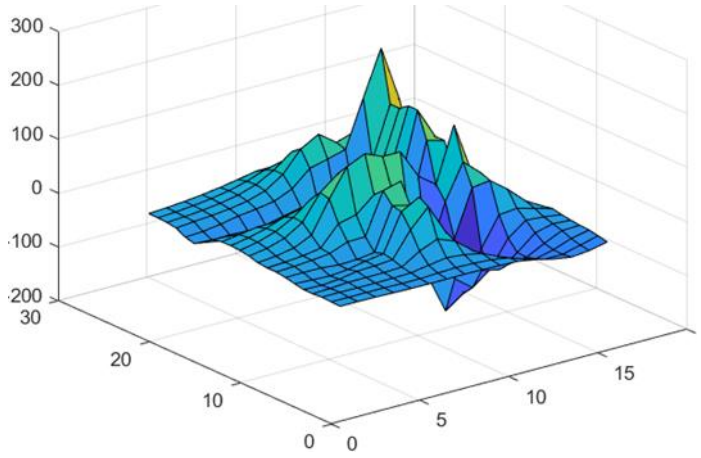


Figure A- 7: 3D visualization of the NCC template matching results in which graph height corresponds to NCC output.

In the results image it can be appreciated that the obtained best solution corresponds once again to the expected one, which is observable in the image as the highest surface peak. However, the whole image presents a flatter aspect, mostly at regions that are far from the best solution, and in consequence, a bigger slope between neighbouring pixels. A remarkable fact is that, the more uniforms the 2D results image is, the less smooth the 3D representation seems to be.

A.3. CONCLUSION

The test has been repeated for multiple variations on the template to enforce deductions.

It can be concluded that both techniques are able to identify the correct template position under the test assumptions. However, SSD approach gives a more smooth output matrix, whose values decrease slower while getting away from the best solution. On the other hand, the change in the results for NCC in a given neighbourhood is more abrupt, reaching their minimum value closer to the best solution than in the SSD approach. In consequence, the output presents a region with higher values around the optima but also bigger descent of the result, giving place to larger regions of null results.

This difference is taken into account for the final choice; but at this point, what is interesting is to validate that these techniques can be applied to the current work and to test our implementation. In next sections, the specific technique to be implemented in the proposed methodology is further discussed.

APPENDIX B: GENERAL CONSIDERATIONS FOR TEMPLATE MATCHING IMPLEMENTATION

B.1. INTRODUCTION

The application of the basic template matching algorithm demonstrated in Appendix A on word images was not successful. Before proceeding to further implementations or comparison between the different available approaches on template matching, some prior considerations are necessary.

As mentioned, the first attempts to implement template matching under the different approaches, the results were not looking reliable at all: They were sometimes very abrupt, or provide very inaccurate solutions or did not converge. However the previous simple test had proven that the algorithm was applicable and that its implementation on the program was correct.

We analysed the main points that may represent a significant difference on the algorithm characteristics between the two sets of images, simple test images and handwriting distance maps; and we found out the most substantial difference were the image dimensions and the greyscale range in the template and Image window being compared at every step. This led us to the decision to deeply analyse with a single pair of images, multiple options of normalization and centring the image values. The tests are applied to a same pair of source image – template under each template matching approach: CC, NCC and SSD.

B.2. ANALYSIS IMAGES DESCRIPTION

One of image words has been used as a testing sample. The image corresponds to the French word “arbre”, written by researcher C. Gosse (UCL), so that calligraphy is easy for preliminary phases. The strategy is to tune and validate the algorithm for this simple case and extrapolate to more complex cases where calligraphy may be worse.

For now, the image word is processed as explained in Chapter 4. This means that data from input files is read to generate the word binary image, on which mathematical morphology is applied and finally its Distance Map is built.

Figure B- 1: Overview of the image processing applied to the digital image of the handwritten word “arbre”, following the steps from our method (mathematical morphology, inversion and distance transform)

Similarly to what happens in real analysis cases, the word in the image is known; so the letters to be matched are also known. For this, the pre-generated templates are used, starting by the one that corresponds to the letter “a”. We use the letter template in its raw shape, omitting possible warping with Lucas Kanade method. This is justified by the fact that both handwriting styles correspond to the same person, thus distortion is very low.

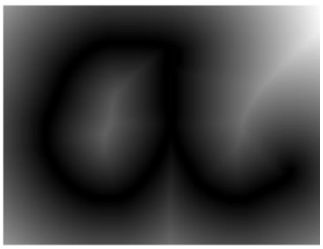


Figure B- 2: Distance Transform of the letter "a" image to be used as template

It has been argued that multiple variants of the Distance Map aspect can be used. At this stage distance maps are left in its linear version as the direct output of applying the transform with Euclidean metrics and saturated at 255.

Template Matching is not applied on the whole source image but only in a reduced region of it: the Analysis Area. To select this region, the estimated position of the letters is used, similarly to the manner in which we proceed in our method. This initialization is described in Section4.4.2 of this work. Due to the fact that Lucas Kanade step is skipped, the first estimate is used for template matching and the template is resized according to that information.

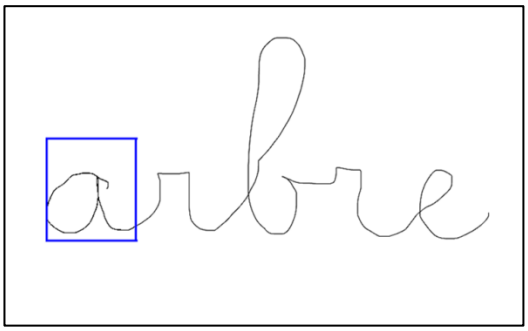


Figure B- 3: Initialization (blue) of the estimated position of the letter "a" in the word "arbre"

In this case, the Analysis Area is set to be two times bigger than the letter estimate position. The initial estimate is not exactly centred in the Analysis Area because of the image limits. In particular, the size of the source image is 656x875pixels, the Analysis Area is 420x403pixels and the template is resized to 155x135pixels. Note that template's resizing is executed after the distance transform is build, similarly to the way we proceed in our method.

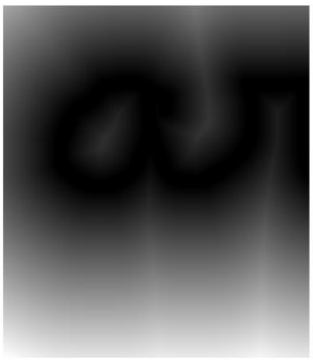


Figure B- 4: Analysis Area for Template Matching of the letter "a" in the word "arbre"

The range of values of the Distance Transform of the region comprised in the Analysis Area is:

$$DM(Analysis\ Area) \in [0, 217.80]$$

B.3. TEMPLATE MATCHING ANALYSIS OVER DIFFERENT MEAN-SUBTRACTION AND NORMALIZATION STRATEGIES

The characteristics of the different tests are related to the image pre-processing to apply template matching. The two working images, the letter and the word distance map images are generated in such a way that their greyscale values belong to the interval [0, 255]. The compared options are:

i) No mean-subtraction or normalization of the images
ii) Work with globally mean-subtracted images. This is, subtract mean of the whole image: $\hat{\imath}(x,y) = i(x,y) - \bar{\imath}$. This is done prior to template matching iteration; and Patch of this mean-subtracted image $\sum_{x,y} \hat{\imath}(x,y)$ is used during iterative process.
iii) Work with locally mean-subtracted images. This is, subtract the mean of the Image Patch being used in each iteration: $\hat{\imath}_{u,v} = \sum_{x,y} i(x,y) - \bar{\imath}_{x,y}$.
iv) Do not subtract the mean of the images but normalize the greyscale range of both images inside the loop. This is, modify the Image Window (subregion of the Source Image being compared to the template at each iteration) so that its greyscale level covers the whole [0 255] range.
v) Combine option (ii) and (iv). This is, work with globally mean-subtracted images and moreover, normalize greyscale range of the Image Window in each iteration.
vi) Combine option (iii) and (iv). This is, work with locally mean-subtracted images and moreover, normalize greyscale range of the Image Window in each iteration.
vii) Apply previous option (iv) in inverted way. This is, inside iterative process, first normalize greyscale range and then subtract the mean of both images.

The main variations being considered are related with two main factors: image-means subtraction and values normalization. Some clarifications are done with respect to these two features.

In the different strategies, the options for mean-subtracting are: doing it considering the whole images (globally), doing right before the template-Image Window comparison (locally) or not doing it at all. The difference between doing it globally or locally is only relevant for the Source Image. This is due to the fact that the region of the Source Image being used for comparison varies all the time, and so does its mean-value. These multiple mean values are also different from the mean value of the whole image. Therefore, the mean-subtraction of the image window is strongly related to the current sub-region being used in the particular iteration step. Differently, the whole template is always used, so its mean value remains constant.

In consequence, for computational reasons, the requirement of some strategies of local mean-subtraction only applies for the source image; the template is mean-subtracted before entering in the iterative part of the algorithm, for strategies in which mean-subtraction is set to be local and global.

Something similar happens for the values normalization. The initial images do already belong to the range [0 255]. Thus it is not significant to normalize the images globally. Given the fact that the whole template is always used, its normalization is actually non-significant in all cases. However, the normalization of the Image may have an impact if it is done inside the iterative loop, because it is then applied to only a sub-area of the image. The values in the sub region of the image under consideration might be in some cases very different form the whole image

ones. This means that in this case too, when the strategy requires to normalize the images inside the iterative part (locally), it is actually only applied for the Image Patch.

To sum up, the template is either left in its original way (strategies (i) and (iv)) or mean-subtracted a priori (other strategies, independently of "local" or "global" requirements); while the options for the Source Image are more diverse (described in strategies (i) to (vii)).

Specific details of template matching techniques applied to get the results of these tests are presented later, when CC, NCC and SSD approaches are compared with the objective of selecting the most appropriate one to use in this work (Appendix C). Hence, algorithm details are omitted now and only the results are shown to centre all the attention to the images pre-processing, common for all strategies.

At first, only translation is allowed for template matching to use the simplicity of the case to get clear conclusions. Brute Force is applied; so the template is placed in all possible positions on the Search window and the Energy Function value for each step is stored in a matrix. The energy function values are plotted in a 3D figure to gain understanding on the computation behind the algorithm. The objective function of the correlation approaches is a measure of the similarities between the template and the different Image Patches; therefore, it has to be maximized. The value of the energy function in these approaches is presented in the exact way that the algorithm provides it, where the highest maxima peaks correspond to the template placements on the Source Image that lead to the best matching results. Oppositely, for the SSD approach, the objective function is a measure of the dissimilarity between the template and the different Image Patches; therefore it has to be minimized. In this case, then, the lowest minima peaks correspond to the best results. To make the results of all approaches comparable, the value of the energy function in the SSD approach is inverted, so that the highest peaks also point to the best matchings. Moreover, for the same reason, all values have been normalized in the range of [0 255]. To plot the results, resolution has been lowered by sampling them in a ratio of ½; so that on every two pixels is used to build the 3D function of the results. The height of the function corresponds to the normalized value of the objective function and the X-Y axis represent the increase of the column and row of the Search Window, respectively, where the template is placed; or, equivalently, to the value of the translation parameters (u, v) applied to reach each result.

Table below collects the optimization function values for the three approaches under the 7 presented strategies.

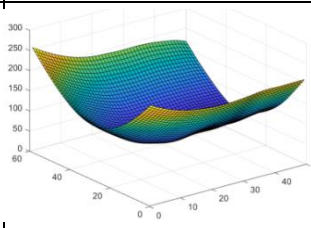
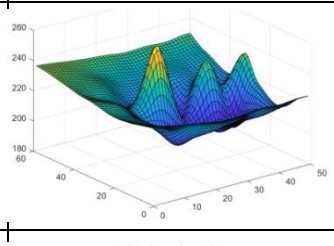
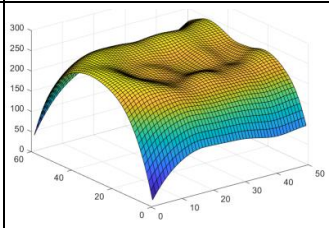
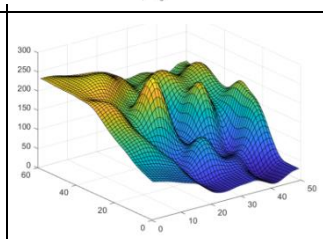
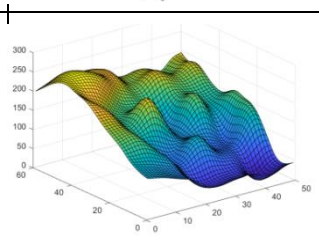
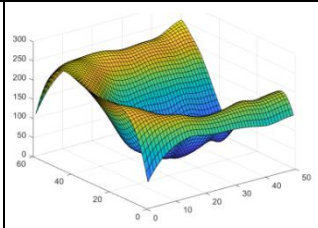
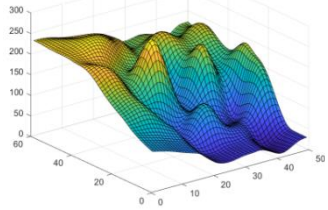
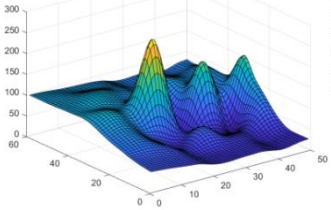
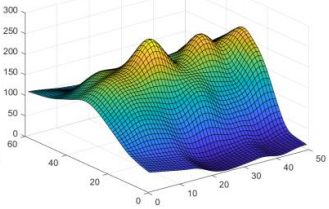
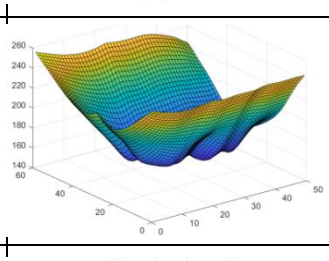
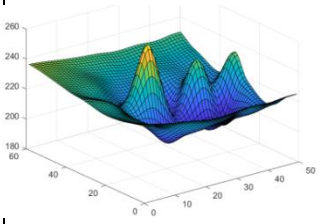
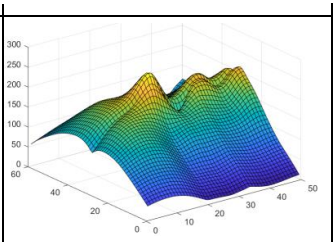
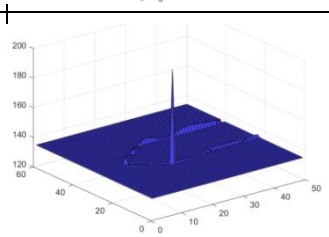
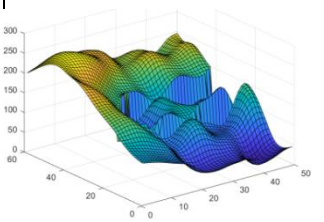
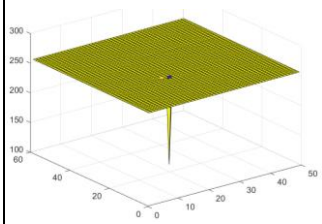
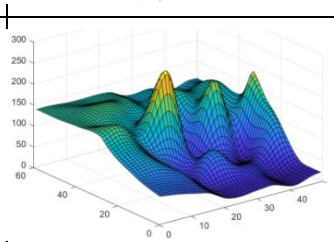
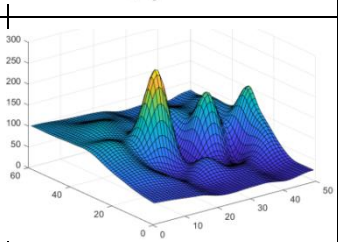
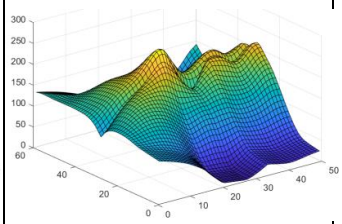
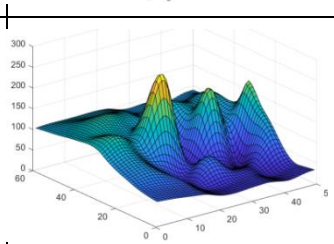
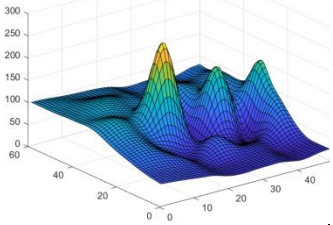
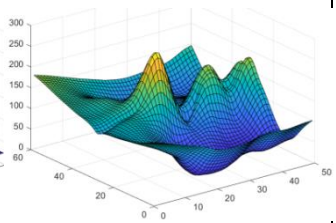
	CC	NCC	SSD
i) -Mean: non-subtracted -Range: non normalized			
ii) -Mean: globally subtracted -Range: non normalized			
iii) -Mean: locally subtracted -Range: non normalized			
iv) -Mean: non-subtracted -Range: locally normalized			
v) -Mean: globally subtracted -Range: locally normalized			
vi) -Mean: locally subtracted -Range: locally normalized			
vii) -Range: locally normalized - Mean: locally subtracted			

Table B- 1: Comparison of the optimization function values manifold for template matching with CC, NCC and SSD under different mean-subtraction and normalization strategies (i to vii).

The first remarkable thing that can be observed in this analysis is the huge impact of the image pre-processing on the template matching results. The comparison between the three different approaches is not analysed here, but however, it is important to take into account that the variations on the images pre-processing affects in a different way the three approaches in most of the cases. Notably, the CC and SSD are much more sensitive to these differences than the NCC approach.

Making a more detailed sight of the results, it can be noticed that some of the tested strategies give bad performance in all three approaches. This is the case for tests (ii) and (v), in which the image mean is subtracted globally, prior to the iterative part of the algorithm. In the first of this two cases, the images are not normalized to a specific values range while in the second one, the Image window being used is normalized in [0 255] every time. Even if none of these two choices gives good results, the second strategy, where the Image Window is normalized every iteration, is remarkably worse. This means that the normalization of the Image Window before comparison does not bring any improvement. Actually, the results when these two choices are combined (v) are the worst that have been obtained in all cases, being very abrupt and meaningless.

Another pair of strategies that showed bad performance is (i) and (iv), in which the image mean was not subtracted at any moment. The difference between the two strategies is that in the second one, (iv) the Image Window greyscale levels are normalized before comparing it to the template. This has a positive impact for the non-normalized techniques, CC and SSD; while is not significant for the NCC. However, the shape of the objective function of any of these approaches under (i) and (iv) strategies is not desirable to guarantee convergence on the optimal solution.

Finally, then, the strategies that give better impression based on the objective function manifold, are (iii), (vi) and (vi). The common feature in all these strategies is that the image window mean is subtracted in each iteration; without prior mean-subtraction of the whole image; and the feature that distinguishes these three strategies is how the images are normalized. This fact explains why the results of the three cases under the NCC approach do not show any significant difference; being the NCC an intrinsically-normalized algorithm.

B.4. CONCLUSION

The conclusion of this analysis is, first of all, that it is necessary to be aware on the details of the algorithm implementation because its impact on the solution is significant. And more particularly, that the best strategy depends on the similarity expression used for template matching, but, for sure, working with local mean-subtraction of the images.

The selection of the most appropriate way to proceed is done according to further analysis to compare the three matching assessment approaches; which are presented below. However, Table B- 1 also gives a clue for the assessment method selection, because it proves that the available options respond differently to image variations; being NCC more robust to greyscale changes than CC or SSD.

APPENDIX C: TEMPLATE MATCHING TECHNIQUES COMPARISON: SSD, CC AND NCC

In the previous analysis, we have validated the basic template matching implementation for distance transform images and we have also tested various implementation strategies with respect to image processing, particularly mean-subtraction and normalization. The conclusion is to use local mean-subtraction of the windows patch being compared to the template every time.

We assume the previous conclusions to proceed with a new analysis. We retrieve the same pair of template-source image that belongs to the handwriting context (reference data) that we have used previously, whose construction and details are described in Appendix B.

The selected pair of images is corresponds to the letter ‘a’ Distance map on the corresponding Analysis Area of the word ‘arbre’ distance map image; both from the reference data set of the work.

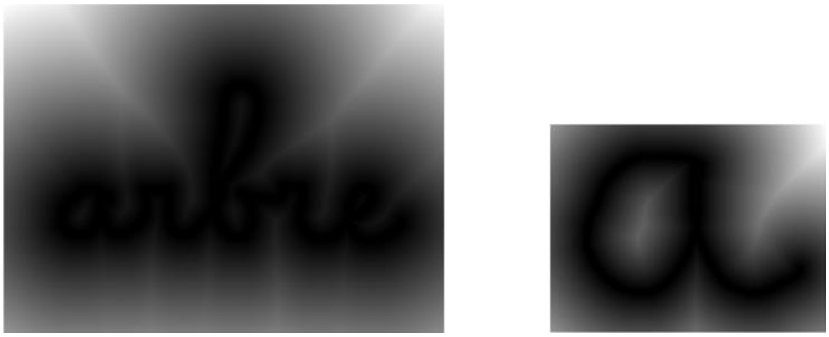


Figure C- 1: Analysis images for template matching analysis. Left: Source image. Right: Template

C.1. Template matching implementation with SSD, CC and NCC

In this analysis, we proceed to the deep comparison between various template matching approaches to select the most appropriate to use in this work. The same strategy and the same pair of images are used in the three different template matching implementations so that the differences in the result are only a direct consequence of the similarity measure being used.

Brute force is applied to compute the similarity between the template and the image for all possible template locations within the SW. This similarity magnitude is different per each of the approaches being compared: CC, NCC and SSD.

SSD

In first place, SSD is applied as the measure of similarity between the two images being compared. SSD is computed in the same way than in the previous example; according to the expression:

$$SSD(u,v) = \sum_{x=1}^{N}\sum_{y=1}^{M}\left[\hat{\imath}_{u,v}(x,y) - \hat{t}(x+u,y+v)\right]^2$$

The results are stored in an array of the same dimensions of the Search window, so that every pixel of the results matrix has the value of the SSD assessment for the matching when the template's upper-left corner is placed in that same pixel coordinates of the image.

Like in the previous image analysis, in SSD assessment the minimum value of the results matrix corresponds to the best position for template matching on the image. However, this time too the results have been inverted and normalized in the range [0, 255].

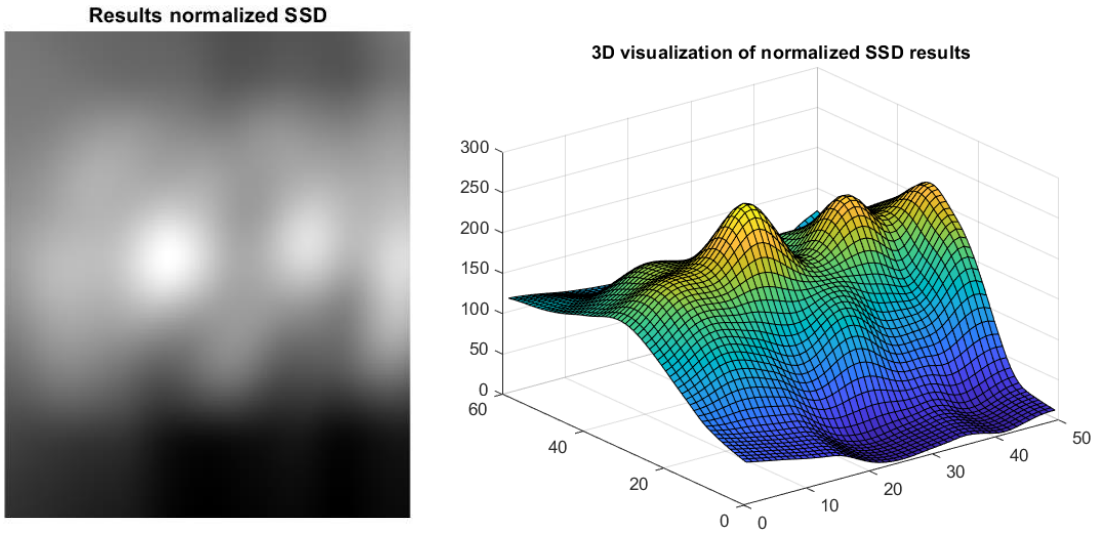


Figure C- 2: SSD template matching results. Left: 2D visualization of the matrix results. Right: 3D visualization of the matrix results in which graph height corresponds to inverted and normalized SSD results

The results are plotted in 3D by associating to each pixel the height of its grey-level (that corresponds to SSD results); and by sampling the image in a ratio 1/5.

The results manifold clearly shows some peaks that indicate the positions where, if the template is placed, matching is optimized. However, only one of the peaks corresponds to the best solution and with this approach, the difference between peaks is not solid. This means that template matching under this approach convergence might be influenced by the existence of multiple local optima; and the template may not be forced to be placed where the maximum is located. On the other hand, the surface of the objective function is very smooth, what is an advantage for convergence in an optimization problem.

Correlation-based techniques

In second place, correlation techniques are applied to the same pair of image and template to be able to compare performance and smoothness. Although it is true that in literature NCC seems to be the most effective approach, basic CC is also tested to check that in the current context this is also true.

CC

Firstly, simple Cross-correlation is used for matching assessment. CC is computed according to the expression:

$$CC(u,v) = \sum_{x,y} \left[\hat{\imath}_{u,v}(x,y) \cdot \hat{t}(x-u, y-v)\right]$$

The results are stored in an array of the same dimensions of the Search window, so that every pixel of the results matrix has the value of the CC assessment for the matching when the template's upper-left corner is placed in that same pixel coordinates of the image.

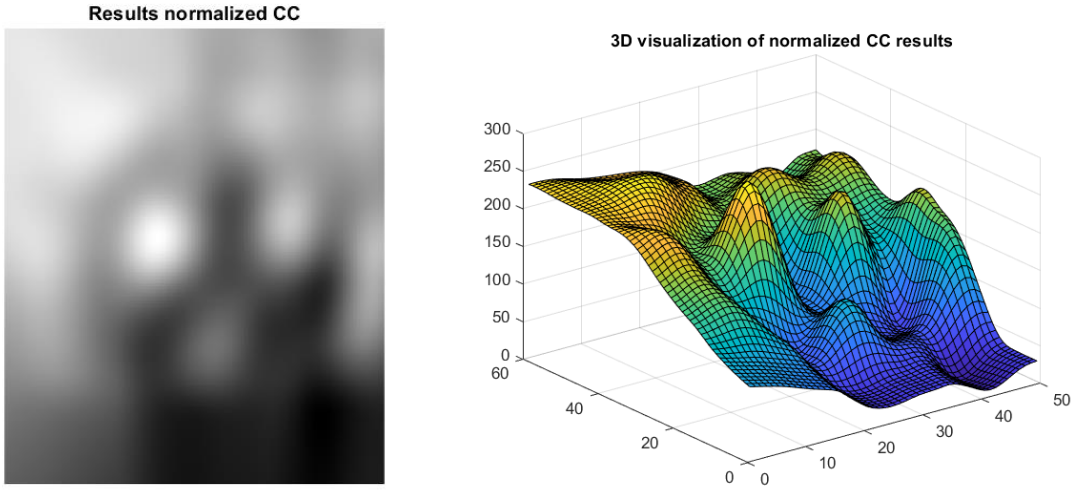


Figure C- 3: CC template matching results. Left: 2D visualization of the matrix results. Right: 3D visualization of the matrix results in which graph height corresponds to normalized CC results

Differently from the NCC, CC is not normalized, what means that its results are note delimited in a particular range; they will belong to any symmetric $\mu \cdot [-1,1]$, being μ any real number $\mu \in \mathbb{R}$. However, the results are normalized and shifted to bring them to the range [0, 255] to make them easily comparable to the other approaches' results.

The results are also plotted in 3D by associating to each pixel the height of its grey-level (that corresponds to CC results) to see the smoothness of the results. The results matrix is sampled in a ratio 1/5 to generate this graphic.

Under this second approach, some peaks can also be observed for those positions that represent the best template placements, but the whole surface is very abrupt. Here, the results manifold is not indicative enough of the best template placement. This lack of smoothness may increase difficulties for the algorithm to converge in its iterative version, and, furthermore, it increases probability of convergence in local optima.

NCC

Finally, Normalized Cross-Correlation is used for matching assessment. NCC is computed according to the same expression that was used for the simple case:

$$NCC(u,v) = \frac{\sum_{x,y} \hat{\imath}_{u,v}(x,y)\hat{t}(x,y)}{\left(\sum_{x,y} \hat{\imath}_{u,v}(x,y)^2 \sum_{x,y} \hat{t}(x,y)^2\right)^{1/2}}$$

The results are stored in an array of the same dimensions of the Search window, so that every pixel of the results matrix has the value of the NCC assessment for the matching if template's upper-left corner is placed in that same pixel coordinates of the image.

The results belong in this case to the range $[-1,1]$, being 1 the value for maximum similarity. The results have been multiplied by the scalar 255 to normalize them in the same range than in the previous cases, [0 255]. The results are also plotted in 3D by associating to each pixel

greylevel (that corresponds to CC results) to a pixel height to see the smoothness of the results. The results matrix is sampled in a ratio 1/5 to generate this graphic

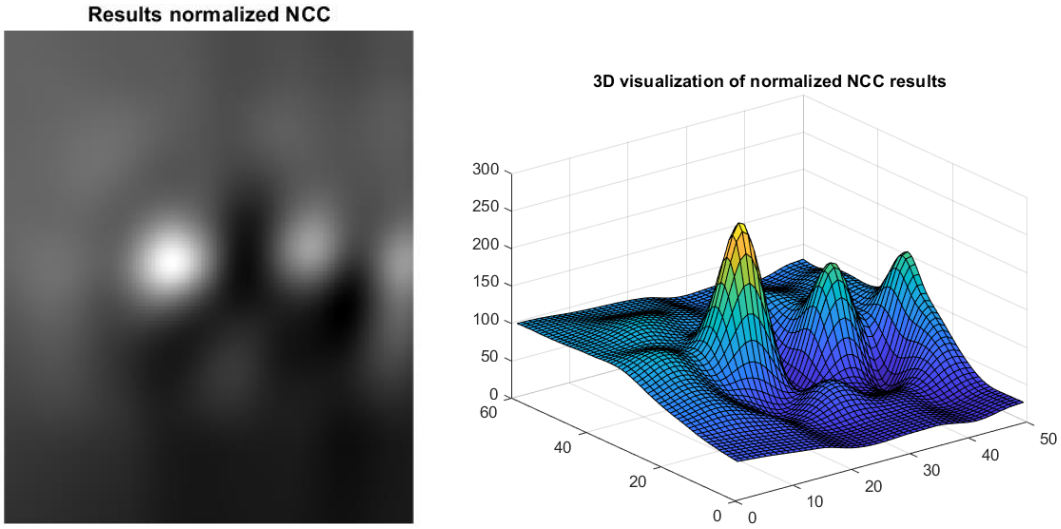


Figure C- 4: NCC template matching results. Left: 2D visualization of the matrix results. Right: 3D visualization of the matrix results in which graph height corresponds to normalized NCC results

The 3rd approach shows a results surface that with higher resolution of the maxima peaks. Bigger difference exists between the value of global maximum and local peaks, what is an advantage to escape from convergence around local maxima. Moreover, the surface is smooth with very weak values for non-desirable position for template placement.

C.2. CONCLUSION

The same template matching example has been solved under three different approaches: CC, NCC and SSD. The objective function that is minimized in each case has been plotted by aping brute force search in all three cases. The reason for this is to understand the background on which each strategy works and how probable is to find convergence of the algorithm around the maximum matching position.

Some clear advantages have been observed of using NCC approach instead of CC or SSD. On the one hand, the function is smooth; hence convergence under gradient descent techniques is improved. Moreover, the value for good matchings is strongly favoured; while it is also penalized for bad matching. This means that this approach clearly reflects the matching accuracy of the images being compared.

It is also necessary to point out that the smoothness of the output surface under SSD is more remarkable. This may be a feature of interest if the single-best position is not the main objective but to obtain an accurate enough result, with a given threshold and making convergence the easiest possible.

In particular in this work, the SSD approach seem appropriate for a preliminary step in which the best solution is still not required but instead it is important to guarantee an approximate matching between the template and a region of the image. After this preliminary phase, the results are refined in a more complete template matching phase where the value of the matching assessment is relevant to make the placement choice. This second objective can be achieved by applying template matching under the NCC.

APPENDIX D: LUCAS KANADE ALGORITHM FOR TEMPLATE MATCHING

The LK optimization algorithm is one of the two main key elements in this work, together with the graph-based representation for segmentation solution. We use this optimization algorithm applied to template matching to warp an initial set of reference letters into a new set of templates more suitable for the word analysis being carried out. Thus, we rely on the output of this LK phase to work with the distorted templates in all the steps that follow until the solution is obtained.

The implementation of for template matching has been a big issue in this work development. Given its relevance on the complete method, we have performed an exhaustive validation phase. The description of this phase and the results discussion is presented in this Appendix.

We have started with the analysis of simple problems, in which the results can be easily validated by comparing with the expected output. These simple problems also help to better understand the algorithm performance and extrapolate it to other cases. From the basic cases, we have gradually added complexity until we have been able to test the LK implementation in the main context of this work, i.e., images of handwriting material.

D.1. LK ALGORITHM ON SIMPLE GREYSCALE IMAGES

In a first stage, we have applied the LK optimization algorithm to solve template matching with a set of simple images that we have created for this purpose. This is, we do not use yet the handwriting images, instead, we use smaller images with basic geometric shapes in which the results are easily predictable.

Moreover, among the 6 available parameters to define the image warping, we also start by only releasing translational DOFS and, gradually we increment the problem complexity.

We work with self-created small images in which we specifically assign the desired value to the pixels to design the tests problems and look for the expected output. Consequently, in these analyses we do not apply any kind of mean-subtraction or normalization of the images; except in the cases in which we explicitly say the opposite.

c) Translation

To test the simplest version of this algorithm, only shifting is allowed between the image and the template during the template matching optimization. For this, only two parameters are required to describe the image warping. To simplify notation, the parameter vector in this case is considered to be 2 dimensional:

$$\boldsymbol{p} = (p_1, p_2)$$

i) Mathematical manipulation

The general formulation of the Lucas Kanade algorithm can be particularized for this configuration in a more simple and compact way.

The Image warping where only translation is allowed can be rewritten as:

$$\boldsymbol{W}([x,y];\,\boldsymbol{p}) = \begin{pmatrix} 1+0 & 0 & p_1 \\ 0 & 1+0 & p_2 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

what results in the equations:

$$\boldsymbol{W} = [W_1, W_2] \rightarrow \begin{cases} W_1 = x + p_1 \\ W_2 = y + p_2 \end{cases}$$

where (p_1, p_2) are the components of the 2-dimensional vector $\boldsymbol{p}$ such that $\boldsymbol{p} = (p_1, p_2) = (u, v) = \left[T_x\ T_y\right]^T$. The warping applied to the image becomes

$$i\big(\boldsymbol{W}([x,y];\,\boldsymbol{p})\big)$$

Then, the objective function can be expressed as follows:

$$e(\boldsymbol{p}) = \sum_{x,y} \left[i\big(\boldsymbol{W}(x,y;\boldsymbol{p})\big) - t(x,y)\right]^2$$

As always, the summation is applied to the Image Window below the template.

Note that as a consequence of the notation that we use, the warping is applied to the image; this means that we assume that the template remains fixed in the global reference frame while the image is shifted (only translation is allowed now) below the template. The algorithm implementation must be coherent with this fact.

We apply the 10-steps description of the method that is presented in the preliminaries (Section 2.3.4) but with the particularized warping description.

Note also that for each algorithm iteration, we use the last update of the parameters vector $\boldsymbol{p}$ to get the new update $\Delta\boldsymbol{p}$ until convergence. The different terms that are needed to compute the parameters update are:

i) Image gradient:

$$\nabla \mathrm{i} = \left[\frac{\partial i}{\partial x}\ \frac{\partial i}{\partial y}\right] = \left[i_x\ i_y\right]$$

ii) Jacobian of the warping:

$$\frac{\partial W}{\partial \boldsymbol{p}} = \begin{bmatrix} \dfrac{\partial W_1}{\partial p_1} & \dfrac{\partial W_1}{\partial p_2} \\ \dfrac{\partial W_2}{\partial p_1} & \dfrac{\partial W_1}{\partial p_2} \end{bmatrix} = \frac{\partial \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix}}{\partial \boldsymbol{p}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

iii) Hessian matrix:

$$\boldsymbol{H} = \left[\sum_{x,y} \left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}}\right)^T \left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}}\right)\right] =$$

$$= \sum_{x,y} \left(\left[i_x\ i_y\right] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)^T \left(\left[i_x\ i_y\right] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) =$$

$$= \sum_{x,y} \begin{bmatrix} i_x \\ i_y \end{bmatrix} [i_x \; i_y] = \sum_{x,y} \begin{bmatrix} {i_x}^2 & i_x i_y \\ i_x i_y & {i_y}^2 \end{bmatrix}$$

The Hessian results in a 2x2 matrix. Then, it is inverted and $\boldsymbol{H^{-1}}$ is obtained.

iv) $\sum_{x,y} \left[\nabla i \cdot \frac{\partial W}{\partial \boldsymbol{p}}(x,y)\right]^T \cdot e(\boldsymbol{p}^*) =$

$$= \sum_{x,y} \left(\begin{bmatrix} i_x \\ i_y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)^T [t(x,y) - i(\boldsymbol{W}([x,y]; \boldsymbol{p}^*)] = \sum_{x,y} \begin{bmatrix} i_x(\mathrm{t} - i) \\ i_y(t - i) \end{bmatrix}$$

v) Finally, $\Delta \boldsymbol{p}$ is computed with the previous terms and the general formula:

$$\Delta \boldsymbol{p} = \boldsymbol{H}^{-1} \left[\sum_{x,y} \left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}} \right)^T \left(\mathrm{t(x,y)} - i\left(\boldsymbol{W}([x,y];\, \boldsymbol{p}^*)\right) \right) \right]$$

$$\Delta \boldsymbol{p} = \begin{bmatrix} \Delta T_1 \\ \Delta T_2 \end{bmatrix} = \begin{bmatrix} \sum {i_x}^2 & \sum i_x i_y \\ \sum i_x i_y & \sum {i_y}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum i_x d \\ -\sum i_y d \end{bmatrix}$$

where $d = d(x,y) = i(x,y) - t(x,y)$ is the difference between the template and the image patch under consideration, and $\boldsymbol{p} = (p_1, p_2) = (T_1, T_2)$ can be updated in the form $\boldsymbol{p} = (T_1 + \Delta T_1, T_2 + \Delta T_2)$. Summations are over x, y below the template.

It is necessary to iterate on parameters update until $\boldsymbol{p}$ converges to its optimal solution.

ii) Application on images

The iterative algorithm has to optimize the position of the template on an Image Window without any resizing, rotation or shearing of the image, and we have implemented it in Matlab.

Stripped Image

To validate the implementation of the algorithm for translation DOFs, an image created by two perpendicular stripes is used. Although the basic idea of the image is based on the two stripes, it is created as a smooth image to make it more similar to the word images where the algorithm is finally being applied.

The template is small region of source image with no distortion, i.e., with the same size and orientation in which it can be found in the source image. The image size is 200x500pixels

In the first test, the template is set to be 100x100pixels. It can be observed in the images below.

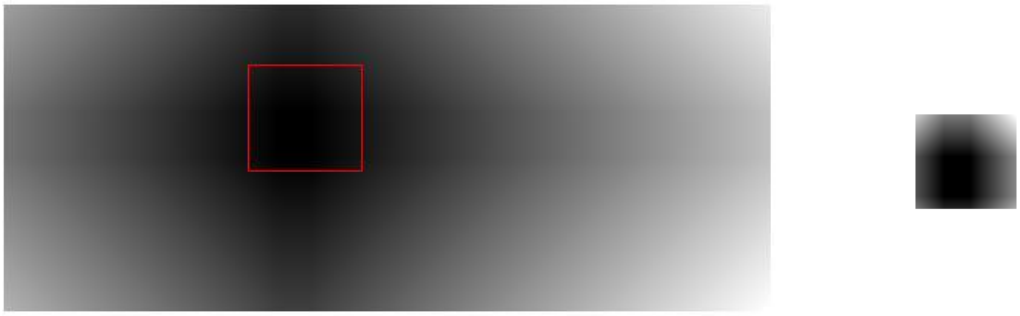


Figure D- 1: Pair of image-template used for the analysis. Left: Source image of dimensions 200x500pixels with a red square indicating the image crop used as a template. Right: Template, of dimensions 100x100pixels that belongs to the IW when the image is placed ad (-160, -40).

Different initializations of the translation parameters $\boldsymbol{p_0} = (p_{1,0}, p_{2,0}) = (T_{1,0}, T_{2,0})$ have been tested, and the optimization algorithm is executed. In all cases, after few iterations (3-4 iterations) the algorithm is able to provide the correct coordinate where the image has to be placed with respect to the template so that the image window below the template perfectly matches it.

Figure D- 2: Image window being compared with the template in the algorithm iterations. Left: initial IW, when the image is positioned at (-100,-50). Right: IW at the 3rd iteration, when the image is positioned at (-159, -39).

Multiple templates have been taken by cropping different areas of different regions of the source image. In all cases, convergence is reached on the optimal solution for perfect matching.

For a more complete validation of the method, the manifold of the objective function values for the current problem is represented. With this objective, we use the previous LK implementation and execute a Brute Force search over the entire Search Window of the translational version of the template matching problem.

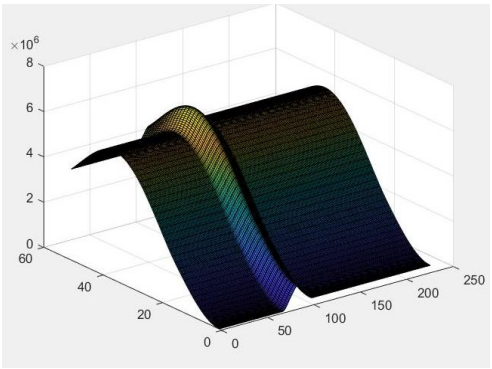


Figure D- 3:3D visualization of the objective function manifold where the X,Y axes correspond to the translational DOFs

To build the 3D representation of the objective function, the height corresponds to its scalar value while the coplanar axes X, Y correspond to the translational DOF's. It can be appreciated the simplicity of the case problem being solved and the coherence between the expected results for the best matching solution and the optimization function manifold.

d) Scaling case

In a second step of the LK implementation validation, we release the 2-direction scaling DOF but we block translation between images. Thus, in this new scenario, we still work with 2 DOFs.

i) Mathematical manipulation

The general formulation of the Lucas Kanade algorithm can be particularized for the scaling problem in a more simple and compact way.

The Image warping where only translation is allowed, can be rewritten as:

$$\boldsymbol{W}([x,y];\,\boldsymbol{p}) = \begin{pmatrix} 1+p_3 & 0 & 0 \\ 0 & 1+p_4 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

what results in the equations:

$$W = [W_1, W_2] \rightarrow \begin{cases} W_1 = x + p_3 x \\ W_2 = y + p_4 y \end{cases}$$

where (p_3, p_4) are the components of the 2-dimensional vector $\boldsymbol{p}$ such that $\boldsymbol{p} = (p_3, p_4) = [K_1\ K_2]^T$

The different terms that are needed to compute the parameters update are in this scenario:

i) Image gradient:

$$\nabla \mathrm{i} = \left[\frac{\partial i}{\partial x}\ \frac{\partial i}{\partial y}\right] = [i_x\ i_y]$$

ii) Jacobian of the warping:

$$\frac{\partial W}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial W_1}{\partial p_1} & \frac{\partial W_1}{\partial p_2} \\ \frac{\partial W_2}{\partial p_1} & \frac{\partial W_1}{\partial p_2} \end{bmatrix} = \frac{\partial \begin{bmatrix} x + p_3 x \\ y + p_4 y \end{bmatrix}}{\partial \boldsymbol{p}} = \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$$

iii) Hessian matrix:

$$\boldsymbol{H} = \left[\sum_{x,y} \left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}}\right)^T \left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}}\right)\right] =$$

$$= \sum_{x,y} \left([i_x\ i_y]\begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}\right)^T \left([i_x\ i_y]\begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}\right) =$$

$$\sum_{x=,y} \begin{bmatrix} i_x x \\ i_y y \end{bmatrix} [i_x x\ \ i_y y] = \sum_{x,y} \begin{bmatrix} (i_x x)^2 & i_x x \cdot i_y y \\ i_x x \cdot i_y y & (i_y y)^2 \end{bmatrix}$$

The Hessian results in a 2x2 matrix. Then, it is inverted and $\boldsymbol{H^{-1}}$ is obtained.

iv) $$\sum_{x,y}\left[\nabla i \cdot \frac{\partial W}{\partial \boldsymbol{p}}(x,y)\right]^T \cdot e(\boldsymbol{p}^*) =$$

$$= \sum_{x,y}\left(\begin{bmatrix} i_x \\ i_y \end{bmatrix}\begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}\right)^T [t(x,y) - i(\boldsymbol{W}([x,y];\boldsymbol{p}^*)] = \sum_{x,y}\begin{bmatrix} i_x x(\mathrm{t}-i) \\ i_y x(t-i) \end{bmatrix}$$

v) Finally, $\Delta\boldsymbol{p}$ is computed with the previous terms and general formula:

$$\Delta\boldsymbol{p} = \boldsymbol{H}^{-1}\left[\sum_{x,y}\left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}}\right)^T \left(\mathrm{t(x,y)} - i\left(\boldsymbol{W}([x,y];\,\boldsymbol{p}^*)\right)\right)\right]$$

$$\Delta\boldsymbol{p} = \begin{bmatrix} \Delta K_1 \\ \Delta K_2 \end{bmatrix} = \begin{bmatrix} \sum (i_x x)^2 & \sum i_x x \cdot i_y y \\ \sum i_x x \cdot i_y y & \sum (i_y y)^2 \end{bmatrix}^{-1}\begin{bmatrix} -\sum i_x d \\ -\sum i_y d \end{bmatrix}$$

where $d = d(x,y) = i(x,y) - t(x,y)$ is the difference between the template and the image patch under consideration, $\boldsymbol{p} = (p_3, p_4) = (K_1, K_2)$ can be updated in the form $\boldsymbol{p} = (K_1 + \Delta K_1, K_2 + \Delta K_2)$. Summations are over x, y below the template. It is necessary to iterate on these steps until $\boldsymbol{p}$ converges to its optimal solution.

ii) Application on images

The iterative algorithm is applied to resize of the Image to optimize the matching between the image and the template if both images are placed on the same point on the reference frame, i.e., the upper-left corner of both images is coincident. This way, the only allowed distortion on the image is scaling with respect to the image reference point.

The scaling problem has represented a significant increase in the problem complexity, being more sensitive to the multiple strategies discussed in this work (different mean-subtraction, normalization, initialization…). Although all these aspects are covered in other sections, we have needed a more exhaustive analysis to validate this phase.

Linear Image

A linear greyscale image is defined to start with. To create this image, the value of the greyscale pixels is increased by one pixel by pixel from left to right and from top to down. Actually, this is equivalent to the distance transform of an image with a single obstacle-pixel in the upper-left corner. The initial dimensions of the image are 200x500pixels. We construct the template in a similar way, but normalizing it to have the same greyscale range than the image.

1	2	3	4	5	6		
2	3	4	5	6	7		
3	4	5	6	7			
4	5	6	7				
5	6						

Figure D- 4: Graphical clarification of the linear image construction

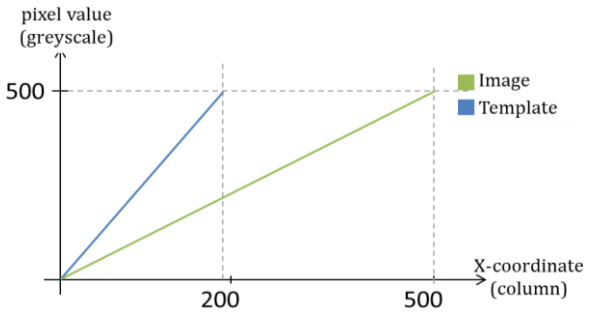


Figure D- 5: Graphical representation of the greyscale value of the pixels of the first row of the image along the X direction

As a direct consequence of the use of first order Taylor approximation, the algorithm applied on a linear image should converge in just one step. The expected result is that the algorithm forces the image to reduce its horizontal dimension to perfectly match the template. Hence we expect no change on the vertical dimension of the image

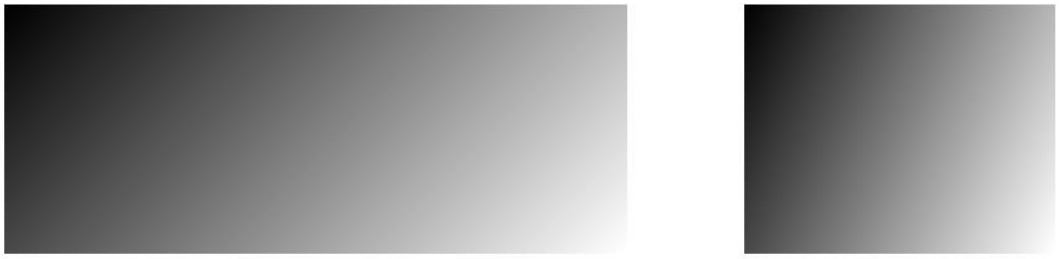


Figure D- 6: Pair of image-template used for the analysis. Left: Source image of dimensions 200x500pixels. Right: Template of dimensions 200x250pixels. Greyscale levels have been normalized for visualization enhancement.

As expected, after only one iteration, the algorithm brings the image to the correct size to exactly match the template in the upper-left corner of the warped image.

Quadratic Image

After the algorithm is validated for the linear case, it is tested on a quadratic image. To create the quadratic image, the previous image is used and the value of each pixel is substituted by its squared value. In this phase, it is expected that the resizing algorithm brings the image to the adequate size but in some more iterations, probably few ones, given the size of the images.

In this case, the image initial size is 200x700pixels and the template 200x350. Hence in this case too, we do not expect any variation on the vertical size ($\Delta K_2 = 0$).

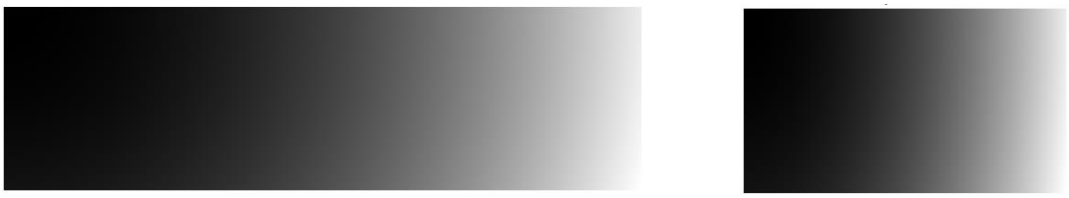


Figure D- 7: Pair of image-template used for the analysis. Left: Source image of dimensions 200x700pixels. Right: Template of dimensions 200x350pixels. Greyscale levels have been normalized for visualization enhancement

As predicted, the algorithm resizes the image and reaches the template dimensions in more than one step. A remarkable fact is that during this process, the warped image for an intermediate parameter vector $\boldsymbol{p}$ is smaller than the template. To solve this, we have included an automatic padding of the image (following the image design) that is applied when necessary. In particular, for the exposed problem, after 5 iterations we reached the perfect result.

Figure D- 8: Resized image after certain algorithm iterations. Left: first iteration, where image dimensions are 200x280pixels. Right: fifth iteration, where image dimensions are 202x347pixels. Greyscale levels have been normalized for visualization enhancement

Stripped Image

After the two first steps to validate the implementation of the algorithm for scaling DOFs, the image created by two perpendicular stripes is retrieved and used again. The template in this case is a resized version of the complete source image.

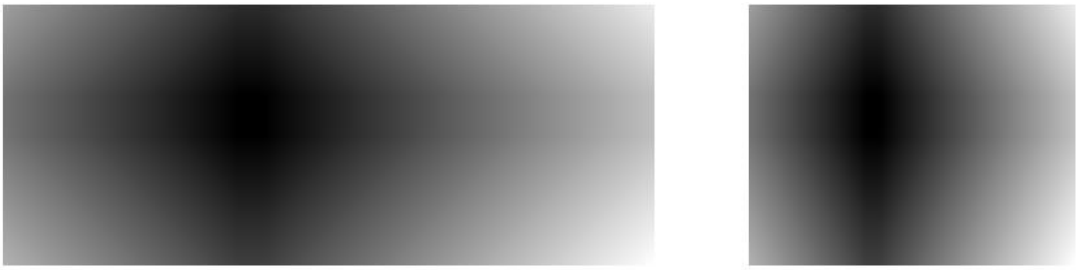


Figure D- 9: Pair of image-template used for the analysis. Left: Source image of dimensions 200x500pixels. Right: Template of dimensions 200x250pixels. Greyscale levels have been normalized for visualization enhancement

In a first try, the range of the greyscale level of both images is made to be coincident, from 0 to 400. Given the fact that the dimensions of both images are different, we increase the greyscale level between consecutive pixels varies in both images to be able to work with the same global range of values. Note that we are not applying any mean-subtraction during this phase; however, this last step is equivalent to a global normalization applied to the template after its resizing.

Like in the previous problems both images have always coincident reference point; and the resized image is padded when necessary to be comparable with the template. After 4 iterations, the algorithm finds a very accurate resizing of the original image so that it matches almost perfectly the template.

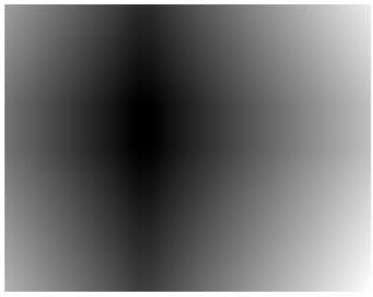


Figure D- 10: Resized image after 4 algorithm iterations, where image dimensions are 199x255 pixels. Greyscale levels have been normalized for visualization enhancement

A second application of the algorithm with the same type of images is held, but the images are generated slightly different: The increase of the greyscale value between consecutive pixels is preserved the same in the image and in the template. In consequence, the global range for the greyscale level of both images will be different, being [1, 400] for the source Image and [1, 250] for the template.

In this case, after 5 iterations the algorithm parameters have converged but the result is slightly different than the optimal solution of the problem. This fact is matches the results of Appendix B, in which the mean-subtraction and normalization aspects of the image are covered. However, this is not the objective of this analysis; instead, we want to validate the algorithm implementation by comparing the expected and the obtained results.

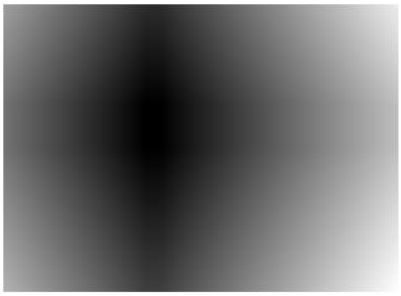


Figure D- 11: Resized image after 5 algorithm iterations where image dimensions are 229x314pixels

For a more complete validation of the method, the manifold of the objective function values for the current problem is represented. With this objective, we use the previous LK implementation and execute a Brute Force search over the entire Search Window of the scaling version of the template matching problem.

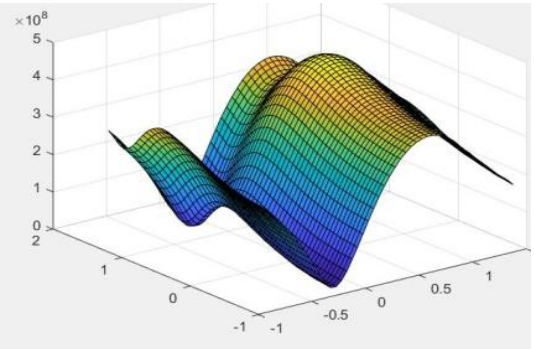


Figure D- 12: 3D visualization of the objective function manifold where the X,Y axes correspond to the scaling DOFs

In this case, the height of the graph also represents the value of the objective function but the X, Y axes correspond to the scaling amount in the respective directions, K_1, K_2. The results, are this time too, coherent with exepcted ones; being possible to observe a smooth peak for the optimum and a minima for the worst position, defined by the particular characteristics of the set of images of the problem.

e) <u>Scaling and translation case</u>

Finally, we arrive to the final stage of the validation with simple images. In this stage we combine the two previous cases. In consequence, 4 parameters are necessary to describe the image warping from the 6 parameter available for the complete affine transform: $\vec{p} = (p_1, p_2, p_3, p_4, 0,0)$. This is the same problem configuration that we have decided to include in our method; therefore, the results of this stage are of big interest.

iii) <u>Mathematical manipulation</u>

The generalized formulation of the Lucas Kanade algorithm can be particularized for the 4 DOFS case in a more simple and compact way.

The Image warping where only translation is allowed can be rewritten as:

$$(\boldsymbol{W}([x,y];\, \boldsymbol{p}) = \begin{pmatrix} 1+p_3 & 0 & p_1 \\ 0 & 1+p_4 & p_2 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

what results in the equations:

$$W = [W_1, W_2] \rightarrow \begin{cases} W_1 = x + p_3 x + p_1 \\ W_2 = y + p_4 y + p_2 \end{cases}$$

where (p_1, p_2, p_3, p_4) are the components of the 4-dimensional vector $\boldsymbol{p}$such that $\boldsymbol{p} = (p_1, p_2, p_3, p_4) = [T_1\ T_2\ K_1\ K_2]^T$.

As we have mentioned from previous instances of the Luca Kanade algorithm, in each iteration, all terms are computed with the last update of the parameters vector $\boldsymbol{p}$ and the warped image**.** The different terms that are needed to compute the parameters update are in this scenario:

i) Image gradient:

$$\nabla \mathrm{i} = \left[\frac{\partial i}{\partial x}\ \frac{\partial i}{\partial y}\right] = [i_x\ i_y]$$

ii) Jacobian of the warping:

$$\frac{\partial W}{\partial p} = \begin{bmatrix} \frac{\partial W_1}{\partial p_1} & \frac{\partial W_1}{\partial p_2} & \frac{\partial W_1}{\partial p_3} & \frac{\partial W_1}{\partial p_4} \\ \frac{\partial W_2}{\partial p_1} & \frac{\partial W_2}{\partial p_2} & \frac{\partial W_2}{\partial p_3} & \frac{\partial W_2}{\partial p_4} \end{bmatrix} = \frac{\partial \begin{bmatrix} x + p_3 x + p_1 \\ y + p_4 y + p_2 \end{bmatrix}}{\partial \boldsymbol{p}} = \begin{bmatrix} 1 & 0 & x & 0 \\ 0 & 1 & 0 & y \end{bmatrix}$$

iii) Hessian matrix:

$$\boldsymbol{H} = \left[\sum_{x,y} \left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}}\right)^T \left(\nabla i \frac{\partial W}{\partial \boldsymbol{p}}\right)\right] =$$

$$= \sum_{x,y} \left(\begin{bmatrix} i_x\ i_y \end{bmatrix} \begin{bmatrix} 1 & 0 & x & 0 \\ 0 & 1 & 0 & y \end{bmatrix}\right)^T \left(\begin{bmatrix} i_x\ i_y \end{bmatrix} \begin{bmatrix} 1 & 0 & x & 0 \\ 0 & 1 & 0 & y \end{bmatrix}\right) =$$

$$= \sum_{x,y} \begin{bmatrix} i_x \\ i_y \\ i_x x \\ i_y y \end{bmatrix} \begin{bmatrix} i_x & i_y & i_x x & i_y y \end{bmatrix} = \sum_{x,y} \begin{bmatrix} {i_x}^2 & i_x i_y & i_x i_x x & i_x i_y y \\ i_y i_x & {i_y}^2 & i_y i_x x & i_y i_y y \\ i_x x i_x & i_x x i_y & i_x x i_x x & i_x x i_y y \\ i_y y i_x & i_y y i_y & i_y y i_x x & i_y y i_y y \end{bmatrix}$$

The Hessian results in a 4x4 matrix. Then, it is inverted and $\boldsymbol{H^{-1}}$ is obtained.

iv) $\sum_{x,y} \left[\nabla i \cdot \frac{\partial W}{\partial \boldsymbol{p}}(x,y)\right]^T \cdot e(\boldsymbol{p}^*) =$

$$= \sum_{x,y} \begin{bmatrix} i_x \\ i_y \\ i_x x \\ i_y y \end{bmatrix} [t(x,y) - i(\boldsymbol{W}([x,y]; \boldsymbol{p}^*)] = \sum_{x,y} \begin{bmatrix} i_x(\mathrm{t} - i) \\ i_y(t - i) \\ i_x x(\mathrm{t} - i) \\ i_y y(t - i) \end{bmatrix}$$

v) Finally, $\Delta \boldsymbol{p}$ is computed with the previous terms and the general formula:

$$\Delta \boldsymbol{p} = \begin{bmatrix} \Delta T_1 \\ \Delta T_2 \\ \Delta K_1 \\ \Delta K_2 \end{bmatrix} = \begin{bmatrix} \sum {i_x}^2 & \sum i_x i_y & \sum i_x i_x x & \sum i_x i_y y \\ \sum i_y i_x & \sum {i_y}^2 & \sum i_y i_x x & \sum i_y i_y y \\ \sum i_x x i_x & \sum i_x x i_y & \sum i_x x i_x x & \sum i_x x i_y y \\ \sum i_y y i_x & \sum i_y y i_y & \sum i_y y i_x x & \sum i_y y i_y y \end{bmatrix}^{-1} \begin{bmatrix} -\sum i_x (t-i) \\ -\sum i_y (t-i) \\ -\sum i_x x (t-i) \\ -\sum i_y y (t-i) \end{bmatrix}$$

and $\boldsymbol{p} = (p_1, p_2, p_3, p_4) = (T_1, T_2, K_1, K_2)$ can be updated in the form $\boldsymbol{p} = (T_1 + \Delta T_1,\ T_2 + \Delta T_2,\ K_1 + \Delta K_1,\ K_2 + \Delta K_2)$. Summations are over x, y below the template.

It is necessary to iterate on these steps until $\boldsymbol{p}$ converges to its optimal solution.

iv) Application on images

The iterative algorithm is applied to resize and translate the Image to optimize the matching between the template and a selected image Window of the Warped Source image.

Stripped Image

To test this new implementation of the algorithm, the previous stripped images are recalled. In this case, though, the template used is a resized version of a region of the source image not containing the upper-left corner of it. This way, to find the best matching, it is necessary to resize and translate the image.

As a result of the problem configuration, in this case, the image is not placed always on the upper-left corner of the template but in a pixel designed by the translation parameters at each iteration; while the size of the source image varies too.

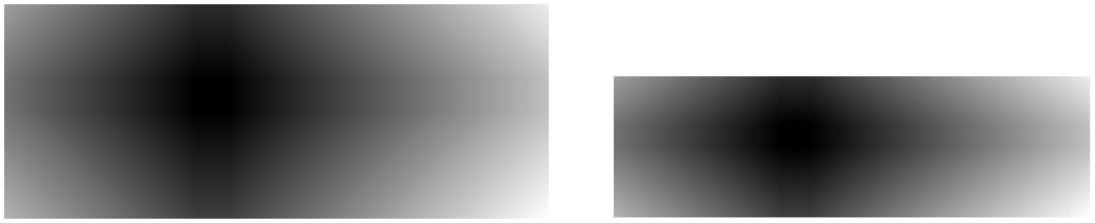


Figure D- 13: Pair of image-template used for the analysis. Left: Source image of dimensions 200x510pixels. Right: Template of dimensions 100x340pixels. Greyscale levels have been normalized for visualization enhancement

As a consequence of the increment in the problem complexity, we expect that more iterations are required in this phase to reach convergence of the parameters vector. In this particular example, after 5 iterations the algorithm reaches a very accurate solution for the current template matching problem; which is actually not superior to the required iterations in the previous analysis. We execute the algorithm step- y-step, so that we are able to check that the value of the objective function decreases with each iteration.

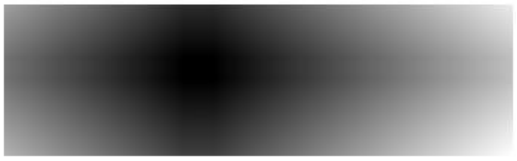


Figure D- 14: Resized image after 5 algorithm iterations where image dimensions are 101x341pixels

After obtaining the problem solution of this problem, we conclude that the 4-DOFs implementation is valid and we proceed with its application in the words context. Therefore, any troubleshooting in the algorithm execution can be associated to the particular characteristics of the word images; looking, therefore, for a solution based on its pre-processing.

D.2. LK ALGORITHM ON WORD DISTANCE MAP IMAGES

It has been proven that the implementation of the Lucas Kanade optimization method for template matching is valid by using simple images. The next step is to validate that it is also able to provide accurate results with the images that belong to this work: distance map images associated to handwritten words and letters.

We proceed as we have done for the previous analysis. At first, only translational degrees of freedom are allowed, and the template that is used is a crop of the complete source image. As the validity of the algorithm is guaranteed, more complexity is added, such as adding DOFs or changing the ideal templates created by image crops for the actual templates that are used for the real complete implementation of the word segmentation methodology.

An important highlight that we must do before continuing this analysis description is that for the words problem, the local mean-subtraction strategy defined after the analysis in Appendix B is used. That same analysis proves that, otherwise, template matching performance is too poor to get any conclusions.

Ideal Template

Moreover, in this first stage of working with words' distance maps, a crop of the image containing the letter being searched is being used. Using this ideal templates that can be perfectly matched it is possible to guarantee that the implementation of the algorithm is correct, because a perfect matching exists; thus it can be found.

The results under this strategy have been satisfactory for different sets of images being compared and under different sets of allowed DOFs, among the ones being considered in this work:$\{T_1, T_2, K_1, K_2\}$.

Real Template

After the correct implementation of the algorithm is guaranteed by the ideal problem, it is convenient to use the real templates so that it can also be validated that the algorithm can manage the current work images and be applied for the purposed methodology.

The same way of proceeding has been applied, by starting from the simplest problem configuration and increasing complexity gradually. To be able to do this, the images have been manually adjusted to be given as inputs for template matching. So the parameters associated to the degrees of freedom that are blocked in the algorithm are set to their optimum value.

Although this test is hold for multiple problem configurations, this is, for multiple word images and its corresponding templates; in this work the results of a single pair of images is presented. In particular, throughout this work, the matching between the letter "a" in "arbre" is used as main example, which is the same problem being used as example for many other explanations The detailed description of these images can be reviewed in Appendix B.

a) Translation

In a first attempt to validate the method, only translation is permitted. For this, scaling factor is manually set for us to its approximated optimum value. This way, it is possible for the algorithm to reach a good matching without modifying the scaling DOFS. Mathematical formulation of the problem has been presented in the previous section.

A part from applying the optimization algorithm in its iterative version, we execute brute force search too to observe the objective function manifold and understand the computational process behind the algorithm.

We remind that in this case we are using local mean-subtraction of the images. However, we have plotted the objective function manifold for the same parameters domain if global mean-subtraction is done instead. This way we can also validate the previous choice for the Lucas Kanade implementation of the general template matching problem formulation.

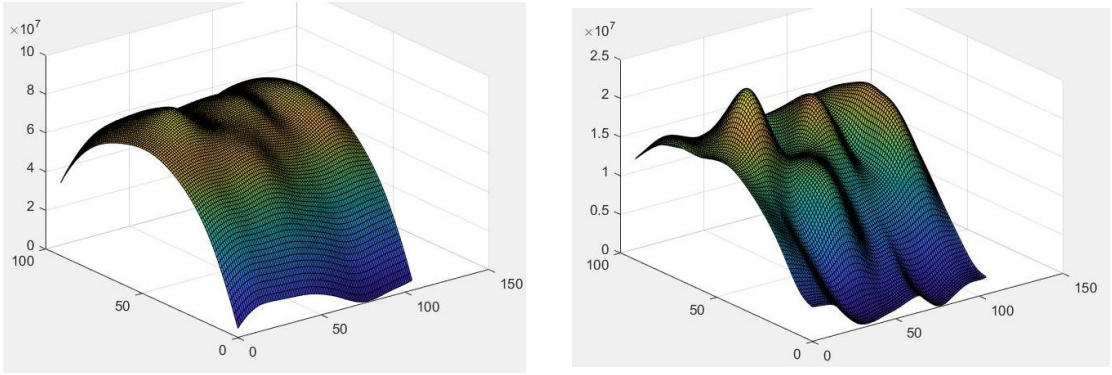


Figure D- 15: 3D visualization of the objective function manifold where the X,Y axes correspond to the translation DOFs under two different implementation strategies. Left: Global mean-subtraction of the images. Right: Local mean-subtraction of the images window

Similarly than in the previous case, the height of the graph corresponds to the scalar value of the objective function for all possible template positions; and the template positions correspond to the X, Y axis. It can be observed that there is a peak that corresponds to the best matching solution and the surface remains smooth. Moreover, the peaks are also smooth enough to permit taking into account multiple solutions.

This test phase is also used to validate the previous decision of mean-subtraction of the image window during the algorithm iterations. For this, Brute force is also applied for translational DOFS in NCC for the case in which the images are only mean subtracted at the beginning of the method, at a global level. The two graphs can be compared and it can be strongly confirmed that the selected strategy improves the performance of the template matching techniques.

b) Scaling

The second phase for the problem validation is to work by only releasing scaling factors. To execute this test it is also necessary to adjust the set of images for template matching so that no translation is required to find a good matching assessment. The image has been manually cropped so that the template matches the first letter in the word if placed at the reference pixel (upper-left corner) with the correct scaling values for the 2 dimensions (K_1, K_2). Mathematical formulation of the problem has been presented in the previous section. However, we remind that in this case we are using local mean-subtraction of the images.

Likewise in the previous tests, the objective function manifold is plotted; being the X, Y axes the corresponding value for the scaling parameters of the image warping. The best scaling factors can be clearly identified with the surface peak of the image; what proves convergence of the algorithm towards the optimum solution.

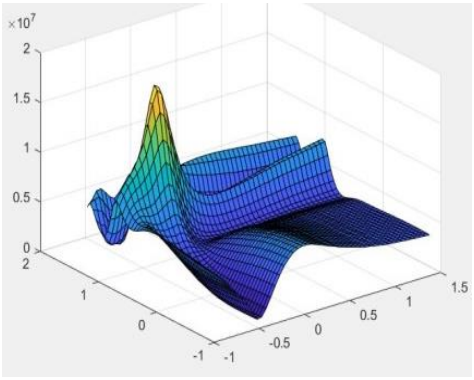


Figure D- 16: 3D visualization of the objective function manifold where the X,Y axes correspond to the scaling DOFs

c) Translation and Scaling

The last step is to implement the Lucas Kanade algorithm with the 4 selected DOFS, $\{T_1, T_2, K_1, K_2\}$, to the word analyis, in the same way that it is applied in the purposed methodology for word segmentation.

This subsection describes, therefore, the same problem configuration that we find included in our method. In it, we use local mean-subtraction of the images; so the previous presented problem formulation has to be adjusted to this fact.

Multiple problem configurations have been set as inputs for validation and convergence of the algorithm is proved. Also, multiple parameters initializations have been tested for each problem, by manually selecting different initialization of the parameters vector. However, by this second step we have been able to observe that one of the main drawbacks of the algorithm working with the 4 DOFS released is that it is sensitive to the initialization. The combination of the translational and scaling DOFs as free parameters for the algorithm increases significantly the problem complexity, increasing simultaneously, the failure ratio of convergence. In some cases, convergence is held around local optima.

To face this challenge in our method implementation, we have designed a set of variation over the initial parameters vector that increases exploration of the results manifold. We have proved that this strategy enforces convergence of the LK templates warping phase.

As it is presented in Chapter 5, we apply the LK algorithm letter-by-letter sequentially; and for each letter we have an initial estimate of the letter position. With this new modification, for each letter in the word, we moreover execute multiple LK optimization stages, one per each variation on the initialization of the parameters vector. Consequently, for each letter we obtain a set of local minima solution, one per each tested initialization. At the end of a letter's analysis, we analyse the set of distorted template and its associate value of the objective function and we select the global optima.

The different variations applied over the initialization of the estimate letter location in this work are presented in the table below:

#	Variation Description	New initialization of the parameters vector
1	Positive horizontal translation	$\boldsymbol{p} = \boldsymbol{p}_0 + [\Delta T_1 \quad 0 \quad 0 \quad 0];\ \Delta T_1 > 1;$
2	Negative horizontal translation	$\boldsymbol{p} = \boldsymbol{p}_0 + [\Delta T_1 \quad 0 \quad 0 \quad 0];\ \Delta T_1 < 1;$
3	Positive Isotropic scaling and Negative horizontal translation	$\boldsymbol{p} = \boldsymbol{p}_0 + [\Delta T_1 \quad 0 \quad \Delta K_1 \quad \Delta K_2];\ \Delta T_1 < 1; \Delta K_1, \Delta K_2 > 1;$
4	Negative Isotropic scaling and Negative horizontal translation	$\boldsymbol{p} = \boldsymbol{p}_0 + [\Delta T_1 \quad 0 \quad \Delta K_1 \quad \Delta K_2];\ \Delta T_1, \Delta K_1, \Delta K_2 < 1;$
5	Negative Isotropic scaling and Positive horizontal translation	$\boldsymbol{p} = \boldsymbol{p}_0 + [\Delta T_1 \quad 0 \quad \Delta K_1 \quad \Delta K_2];\ \Delta T_1 > 1, ; \Delta K_1, \Delta K_2 < 1;$
6	Positive vertical scaling	$\boldsymbol{p} = \boldsymbol{p}_0 + [0 \quad 0 \quad 0 \quad \Delta K_2];\ \Delta K_2 > 1;$
7	Negative horizontal scaling:	$\boldsymbol{p} = \boldsymbol{p}_0 + [0 \quad 0 \quad \Delta K_1 \quad 0];\ \Delta K_1 < 1;$
8	Negative horizontal scaling and Negative horizontal translation	$\boldsymbol{p} = \boldsymbol{p}_0 + [\Delta T_1 \quad 0 \quad \Delta K_1 \quad 0\ ];\ \Delta T_1, \Delta K_1 < 1;$
9	Negative horizontal scaling and Positive horizontal translation	$\boldsymbol{p} = \boldsymbol{p}_0 + [\Delta T_1 \quad 0 \quad \Delta K_1 \quad 0];\ \Delta T_1 > 1, \Delta K_1 < 1;$
10	Positive horizontal scaling	$\boldsymbol{p} = \boldsymbol{p}_0 + [0 \quad 0 \quad \Delta K_1 \quad 0], \Delta K_1 > 1;$
11	Negative vertical scaling	$\boldsymbol{p} = \boldsymbol{p}_0 + [0 \quad 0 \quad 0 \quad \Delta K_2];\ \Delta K_2 < 1;$
12	Negative vertical scaling and Negative horizontal translation	$\boldsymbol{p} = \boldsymbol{p}_0 + [\Delta T_1 \quad 0 \quad 0 \quad \Delta K_2];\ \Delta T_1, \Delta K_2 < 1;$
13	Negative vertical scaling and Positive horizontal translation	$\boldsymbol{p} = \boldsymbol{p}_0 + [\Delta T_1 \quad 0 \quad 0 \quad \Delta K_2];\ \Delta T_1 > 1, \Delta K_2 < 1;$

Table D- 1: Description of the 13 variations that we apply on the initial parameters vector of the LK phase per each letter

D.3. CONCLUSION

With this exhaustive analysis we have covered all the details on the Lucas Kanade algorithm implementation.

First of all, we have validated the applicability of the method in the template matching field; with our basic examples. Simultaneously, we have been able to validate our particular implementation in multiple problem configurations, i.e., if different sets of DOFs are released. We find this very important because our method can be used as the starting point for new applications, in which case, it may be convenient to readjust some steps; such as varying the number of released DOFs.

Secondly, we have gain understanding on the algorithm performance in our method. This fact has permitted us to achieve more control of the required modifications on the implementation for specific improvements on the results. Hence, the knowledge of the weak points of the algorithm gives us the idea of the possible actions to counter the drawbacks.

For example, we have encountered that one of weak points is related to its sensitivity to the initialization; and we have mentioned that we select the optimal solution among the set of warped templates that we obtain after each initialization. However, to counter this drawback, we consider the option of keeping the set of local minima solutions and use the next graph-based representation of the problem to finally disambiguate and select the most advantageous distortion per each template. This proposition is discussed in Chapter 7.

APPENDIX E: FINE ALIGNMENT OF WARPED TEMPLATES WITH NCC TEMPLATE MATCHING

E.1. INTRODUCTION

Among the different tested options to assess template matching, NCC has proven its efficiency and good performance. The correlation is less sensible to difference grey-scale ranges between the two images and other dissimilarities that can take place. Moreover, the difference between a good matching and a worse one is clearer using this normalized measure.

However, to guarantee validity of results, some more tests are applied to template matching with NCC. These tests are carried out with images that belong to our method context; so we use the same word and letter distance maps as source image and template, respectively. The complete description of the images and its characteristics can be found in Appendix B.

E.2. TEMPLATE MATCHING WITH NCC ON WORD IMAGES

The template is placed in all possible positions of the Search Window of the Analysis Area and Normalized Cross correlation between the template and the Image window is computed. Hence, we obtain the results manifold in a brute force mode. Then, we can plot this output manifold to get some conclusions on the algorithm's performance and robustness. The implementation details of the NCC template matching are not included in this section; they can be found in Appendix C.

Once again, the results are presented as a greyscale matrix which corresponds to the Search window and as a 3D plot of this same surface where the greyscale determines each pixel height.

It is of big interest to appreciate in the 3D plot that there is actually a peak, which is the optimum of the NCC results within the Search Windows; but the gradient that leads to this peak is smooth. Therefore, the best position for the current template can be known but pixels around, where NCC optimal result is close, have also good matching assessment.

This fact is interesting when it comes to select the template position not only of one letter of the word but all the letters, enchaining all of them. When this is done, the optimal solution for each letter's recognition may not be the best choice, thus it is good to have more candidates to explore.

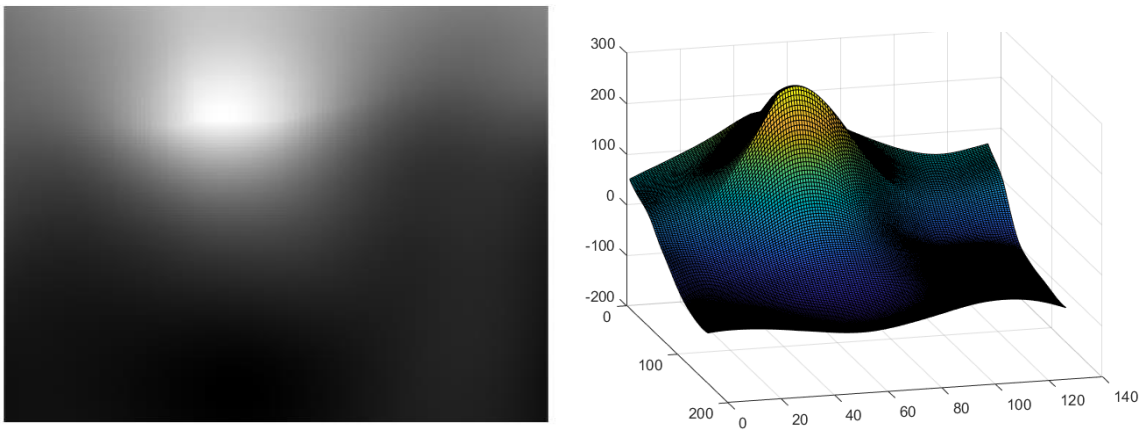


Figure E- 1: NCC template matching results for "a" in "arbre" when AA is twice the size of the template. Left: 2D visualization of the matrix results. Right: 3D visualization of the matrix results in which graph height corresponds to NCC value

The NCC test on the handwriting text is extended by increasing the Search window for the same word and letter. Thus, same template is being used on an Analysis Area that is three times bigger than the initial letter estimate, i.e., three times bigger than the template. This allows the algorithm to explore a larger region.

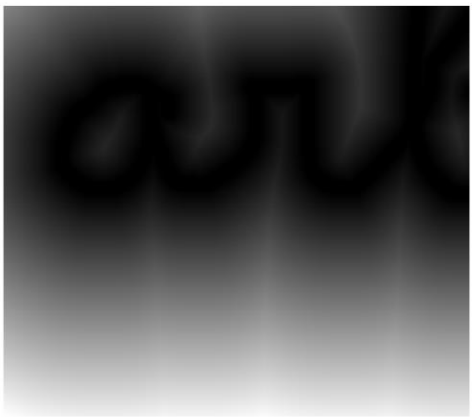


Figure E- 2: Analysis Area for template matching with NCC for the letter "a" in the word "arbre". AA is three times the template size

The computation steps are exactly the same that have been applied to previous tests. Obviously, the results matrix is bigger in this case because so is the Search window.

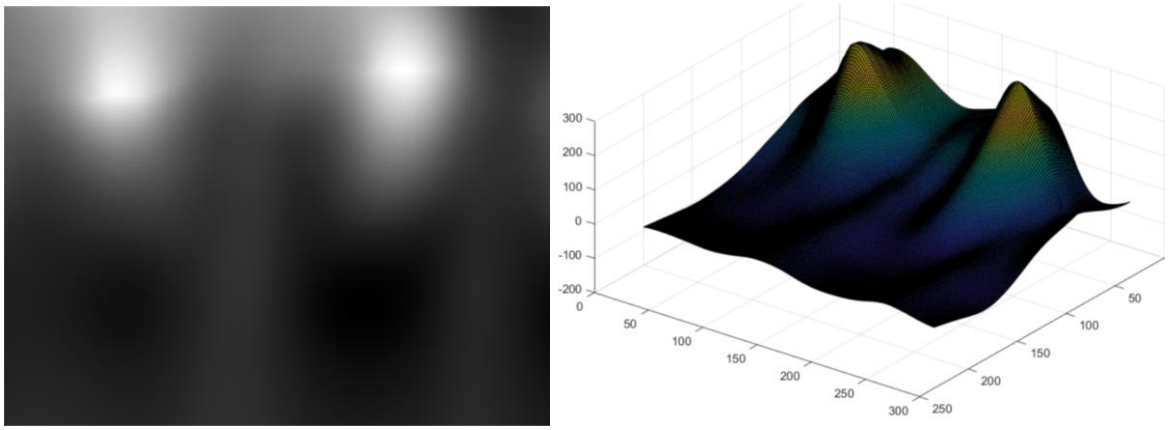


Figure E- 3: NCC template matching results for "a" in "arbre" when the AA three times bigger than the size of the template. Left: 2D visualization of the matrix results. Right: 3D visualization of the matrix results in which graph height corresponds to NCC value

The results are also presented in a 3-dimensional function, where the height corresponds to the pixels' greyscale level. In this case the results matrix is not sampled; all the pixels information is used to generate the 3D graph.

At this point, the greyscale image of the results shows that there are two maxima of normalized cross-correlation evaluation for the template matching problem being solved. One of these peaks is the global maxima while the other one, slightly smaller, is local. The 3D plot of the results shows that actually, the peak that places the template where the real "a" is, corresponds to the global one, being higher than the peak that would place the template in a very wrong position.

Nevertheless, it is interesting to see that this situation can appear when template matching is solved to take in into account for the final algorithm. The complete implementation of this work

works at a global word-level. This strategy is a great advantage to face situations like the one showed in this test, because the constraints imposed between letters help scape from these mistakes. Reasonable sequence of connection between matched templates discards non-logical matches even if the NCC assessment is acceptable.

If the template is positioned where the global maxima indicates there is the best NCC value, the matching result is:

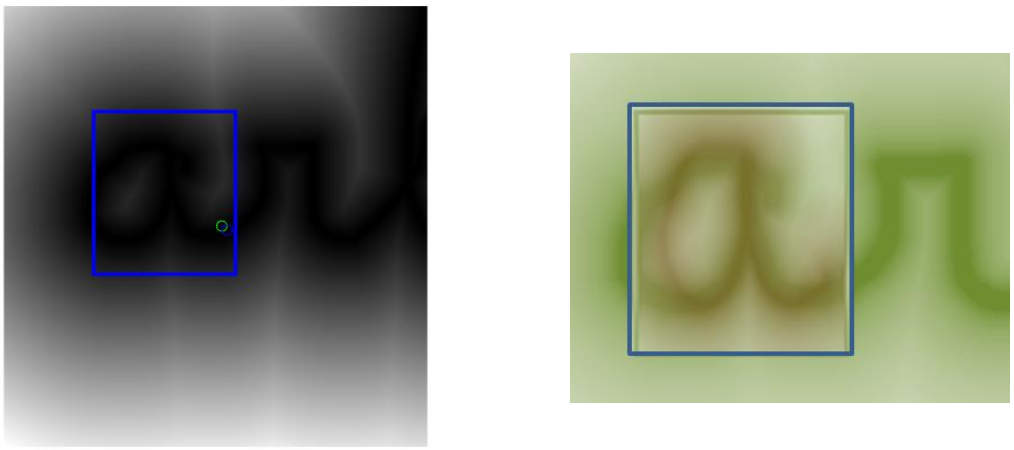


Figure E- 4: Template matching best solution for "a" in "arbre". Left: AA with a square (blue) that indicates the best template position, a small for template's end point (green) and a small circle for end point on the word (blue). Right: Template (soft red) overlapped with the image (soft green) at the indicated position by NCC results (blue square).

In Figure E- 4:, the final point of both, template and letter are marked. The final point of real letter in point is here computed as the closest one to the template end point that can be found on the word contour.

To better appreciate correspondence between image and template for the obtained results, Figure E- 4: shows both images plotted in different layers. The template is shown on top of the word image, at the indicated position, with some transparency to also appreciate the image at back.

E.3. CONCLUSION

We have observed that with this algorithm we may find convergence in local optima. However, we consider that this drawback is countered in our method, because we have a preliminary phase, implemented with LK, which does a coarse template matching phase. Therefore, the template alignment that we perform with NCC is conducted in a very fit AA around the first TM result. This fact decreases probability of finding local minima within the SW. Besides, the graph-based segmentation solution also compensates this disadvantage as a result of constraining the NCC solutions to spatial relationship between letters.

In conclusion, we can confirm that performance of the NCC algorithm is valid for our work. We have validated its implementation and moreover we have been able to proof accuracy in the results that we obtain. In consequence, we decide that the warped templates fine alignment within the graph-based representation of the problem is controlled by template matching with NCC.

APPENDIX F: WORD SEGMENTATION APP USER-GUIDE

The Word Segmentation APP is designed with the objective to read a sequence of samples (from a CSV file) that belong to handwriting words recorded with a pressure-sensitive device and process them to obtain the words' segmentation in letters.

F.1. ASSUMPTIONS

Words known a priori

One of the main assumptions for the current work is that the content of word being processed is known. Therefore, the user must provide the digital encoding of the words that corresponds to the loaded samples to analyse. An association of the samples to the wrong (list of) words will results in very poor results.

Non-regular writing

This program is designed to manage handwriting from children learning how to write, thus usually irregular handwriting. However, a pre-visualization of the digital word images is advised, to discard words with orthographical mistakes or too poor calligraphy. The program may not be able to find a solution for these cases.

Cursive writing

Currently, the set of templates of the program corresponds to lower-case cursive handwriting. In consequence, any attempt of word segmentation applied to other handwriting styles is not contemplated and the program may not be able to find a solution.

F.2. DATA

F.2.1. Pre-Computed Data

The program code download provides of a set of pre-computed data.

a) Alphabet

A Matlab variable (*alphabet.mat)* of all the letters of the alphabet concatenated as a single character. It includes the characters "´é", "è" and "â".

b) Letters' reference points

A Matlab variable (*reference_points.mat)* that contains the coordinates of the templates' reference points.

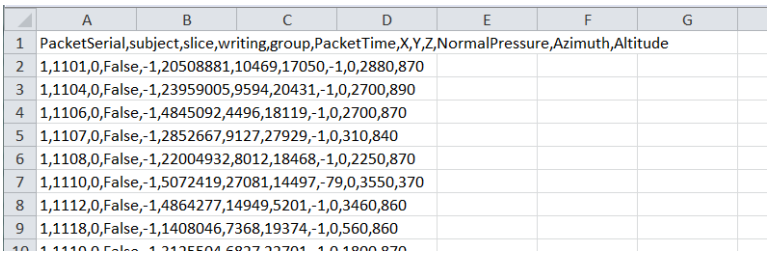
c) Template Distance maps

A list of Matlab variables *(WS_letter#.mat)* that represent the templates (DM of the letter images). There is one per each alphabet letter.

F.2.2. Handwriting data

Two input files are required to the user to execute the program.

i) First, a single CSV file that contains the complete set of samples that belong to the handwriting words to be processed, in the same format specified in this document.



	A	B	C	D	E	F	G
1	PacketSerial,subject,slice,writing,group,PacketTime,X,Y,Z,NormalPressure,Azimuth,Altitude						
2	1,1101,0,False,-1,20508881,10469,17050,-1,0,2880,870						
3	1,1104,0,False,-1,23959005,9594,20431,-1,0,2700,890						
4	1,1106,0,False,-1,4845092,4496,18119,-1,0,2700,870						
5	1,1107,0,False,-1,2852667,9127,27929,-1,0,310,840						
6	1,1108,0,False,-1,22004932,8012,18468,-1,0,2250,870						
7	1,1110,0,False,-1,5072419,27081,14497,-79,0,3550,370						
8	1,1112,0,False,-1,4864277,14949,5201,-1,0,3460,860						
9	1,1118,0,False,-1,1408046,7368,19374,-1,0,560,860						

Figure F- 1: Abstract of the input data file of children's handwriting recorded by WACOM SmartPads (CSV)

This file must contain data samples from all subjects and all words. However, it should not contain more than 1million samples, otherwise it is necessary to split it in multiple files (using an external tool). The expected parameters for all samples are presented in the Table below:

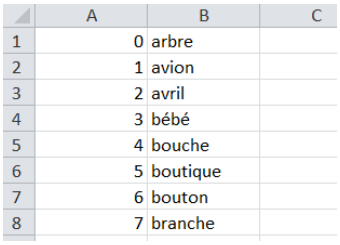
Feature	Description
index	Indexing of points starting at 0
PacketSerial	Indexing of points (=packets)
slice	Index of the stroke (i.e. sequences of consecutive points that are writing (or not writing)) to which the point belongs
writing	True or False depending if the pen touches the tablet or not
group	Index of the word to which the point belongs, starting at 0. (-1)index is used for strokes without writing between two groups
subject	Index of the subject that is writing. Index, PacketSerial, slice and group indexing restart (at 0 or 1) for each subject.
PacketTime	Time of arrival of the point/packet in milliseconds
X	Horizontal coordinate. 0 - 44703 points (1pt = 0.005mm).
Y	Vertical coordinate. 0 - 27939 points.
Z	Height coordinate. (-512) - (-1) points.
NormalPressure	Pressure of the pen on the tablet. If zero, the pen doesn't touch the tablet
Azimuth	Angle of the pen
Altitude	Angle of the pen

Table F- 1: Information associated per each SmartPad sample of children's handwriting

In particular, for this work development, samples recorded with a WACOM SmartPad are used

ii) Second, a CSV file that contains the list of words that corresponds to the first file (data samples), numerated.

This file must have two columns: A first one with the words numbering and a second one, with the digital encoding of the words. All the words must be written in lower-case and including the accents when necessary.



	A	B	C
1	0	arbre	
2	1	avion	
3	2	avril	
4	3	bébé	
5	4	bouche	
6	5	boutique	
7	6	bouton	
8	7	branche	

Figure F- 2: Abstract of the List of words input file (CSV)

F.3. PROGRAM EXECUTION

When user has the necessary data, Matlab App "Word Segmentation" has to be launched.

F.3.1. START Tab

The ***START Tab*** is divided in 4 main areas:

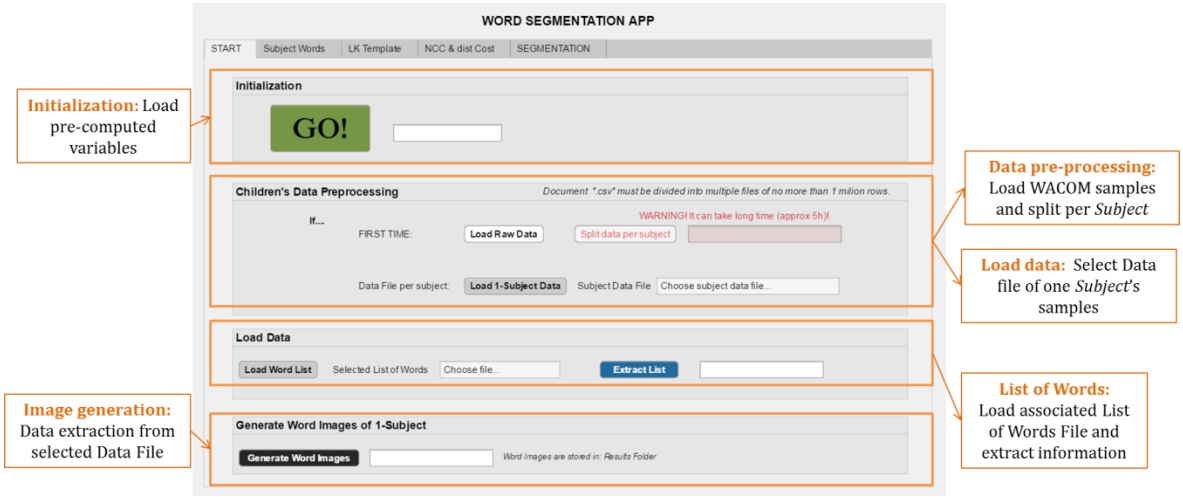


Figure F- 3: START Tab from the Word Segmentation App

1) Initialization: Loads pre-computed data and generates other temporary variables needed for current execution. Also generates a new folder in working directory to store results (*WS_Results_Data)*

2) Children's Data Pre-processing: Two options of data processing depending on the user needs:

    a. Load Raw Data: First time of analysis of a (WACOM) set of samples.

This option must be used only the first time that the App is launched for a new experiment data set. The user must load the CSV file(s) that contain the samples from all words and all subjects. The file should be uploaded and *Split data per subject* calls a function to split the original file into multiple CSV files with the same format, each one with one subject's samples.

New files are stored in the same folder than input file and named as *Data_Subject####.csv*

    b. Load 1-Subject Data: Analysis of a new subject's handwriting.

If the user has the set of CSV that belong to the different children, this option is used to pick the specific data file to be processed.

3) Load Data: The CSV file that contains the list of words must be uploaded using *Load Word List* button. *Extract list* calls the function to obtain important characteristics of the words in the list and store them into a new variable *List_words.mat*. User must guarantee that the list of words matches the samples-file uploaded in (2).

4) Generate Word Images of 1-Subject: Read the content of the samples data file and generate the corresponding digital images of the handwritten words. The images (.tif) are stored in a new folder named after the selected subject's code, inside the previous generated directory (*\WS_Results_Data\Subject####).* The images are numbered in the same order as they appear in the initial data file (WACOM *group).*

F.3.2. Subject Words Tab

The ***Subject Words Tab*** is divided in 4 main areas as well:

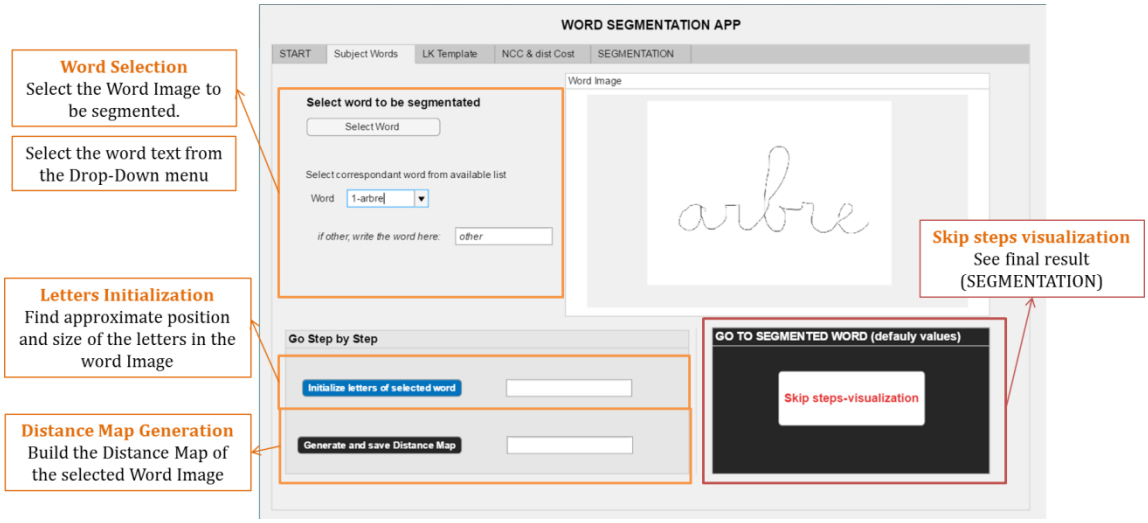


Figure F- 4: Subject Words Tab from the Word Segmentation App

1) Word Selection: The user can select from its local directory one of the previously generated images that corresponds to the word to be segmented (*Select Word).* Then, the same word has to be selected from the Drop-Down menu. The words are numerated to help finding the word in the list.

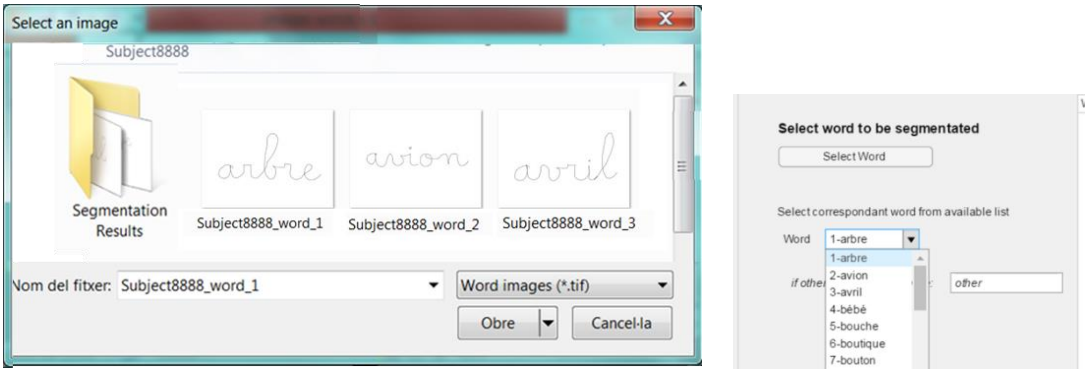


Figure F- 5: Word Selection menu from the Word Segmentation App

2) Word Image: Once the word is selected by user, the program shows the binary image on this area. User can validate his choice.

Next two options are not sequential; only (3) or (4) is necessary to continue, depending on the user's preference.

3) Go Step By Step: With this choice, the user decides to continue all the words segmentation following the steps one by one. This means that this choice affects subsequent tabs as well.

    a. Initialize letters of selected word: The program matches the selected image with its main characteristics based on the words' list variable (*List_words.mat).* This variable is modified by adding an estimate of the position and size of all the letters in the word, based on a spatial grid.

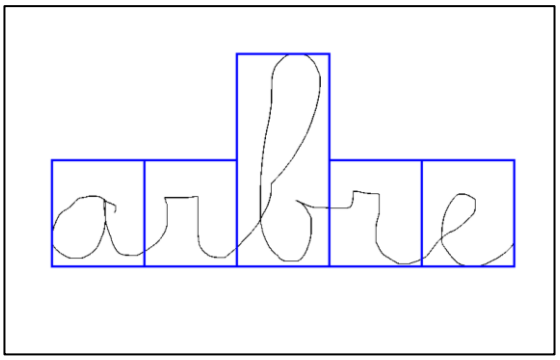


Figure F- 6: Estimate position and size of the letters of the word "arbre" obtained

    b. Generate and save Distance Map: Generate the distance map of the word under analysis and stores it as a Matlab variable.

4) Go to segmented word: With this option, all intermediate steps are skipped and the App shows final result. With this option, the user has no more interaction with the algorithm, thus default parameters are used.

F.3.3. LK Template Tab

The ***LK Template Tab*** is divided in 3 main areas:

1) Define Analysis Window: Based on the estimate position and size of the letters, an Analysis Area is defined per each letter.

2) Template Matching elements: The Analysis Area and the resized template associated to each letter in the word are plotted. User can vary the Analysis Area dimensions with the slide-controls for horizontal and vertical longitude (Ax, and Ay, respectively).

3) Modify templates using LK: Use LK template matching optimization algorithm to warp the original template (showed in (2)) and get a specific set of templates more similar to current word's handwriting. Once computed, the new templates substitute the raw ones in (2). User can compare new templates with the real letters in the word. The new templates are also stored as temporary Matlab variables.

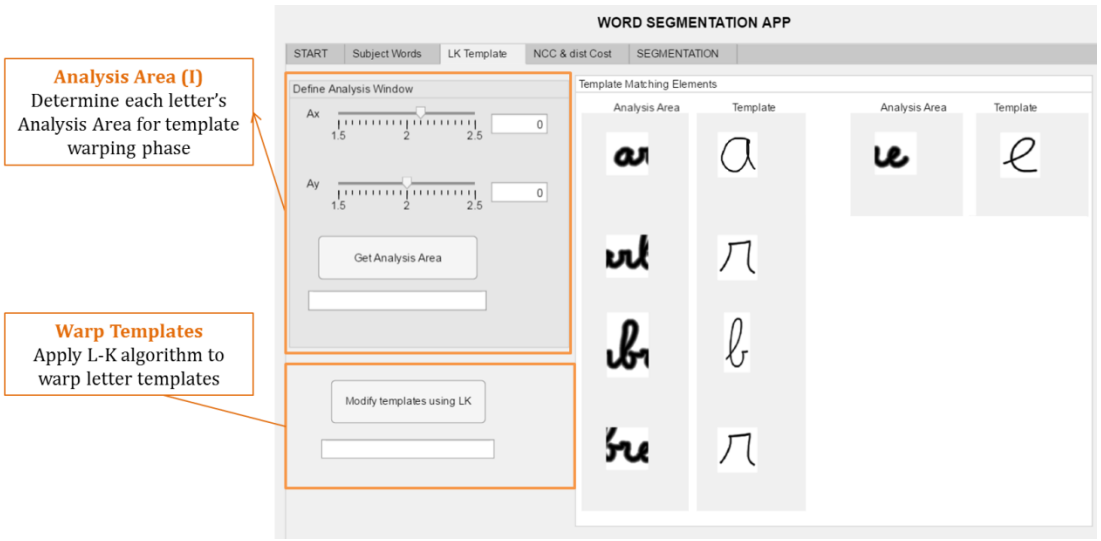


Figure F- 7: LK Template Tab from the Word Segmentation Tab

F.3.4. NCC & dist. Cost Tab

The ***NCC & dist. Cost Tab*** is divided in 3 main areas:

1) Correlation Cost: Execute template matching by local translation of the warped templates with NCC

    a. Get Analysis area NCC: Define one new Analysis Area per letter based on the previous LK results for template size and best location. Analysis Areas in this step are tighter to the template's contour.

    b. Compute correlation: Apply template matching with NCC for each warped template in its Analysis Area in Brute Force mode, only for translational DOFs. Store all results.

2) Distance cost: Compute the inter-letters distance per all consecutive letters.

    a. Find letters' reference points: Find the reference points on the warped templates (apply the same warping to initial coordinates for reference points).

    b. Compute distance between letters: Use the reference points coordinates to compute the distance between the end and the beginning of consecutive letters. Store all results.

3) Graph-based representation: Combine the template matching and the inter-letters distance results in a single graph-representation of the problem. Graph nodes stand for translation possibilities of the warped templates on the word images and graph edges account for the cost associated to (i) template matching results and (ii) distance between letters. Both costs are computed per all possible templates displacement.

a. Create Matrix Blocks: Evaluate the output of (1) and (2) respectively and combine the results per each letter. User can decide the weight to each cost contribution with the *Cost* slide.

b. Create Whole matrix: Rearrange letter's costs in an appropriate graph-based structure.

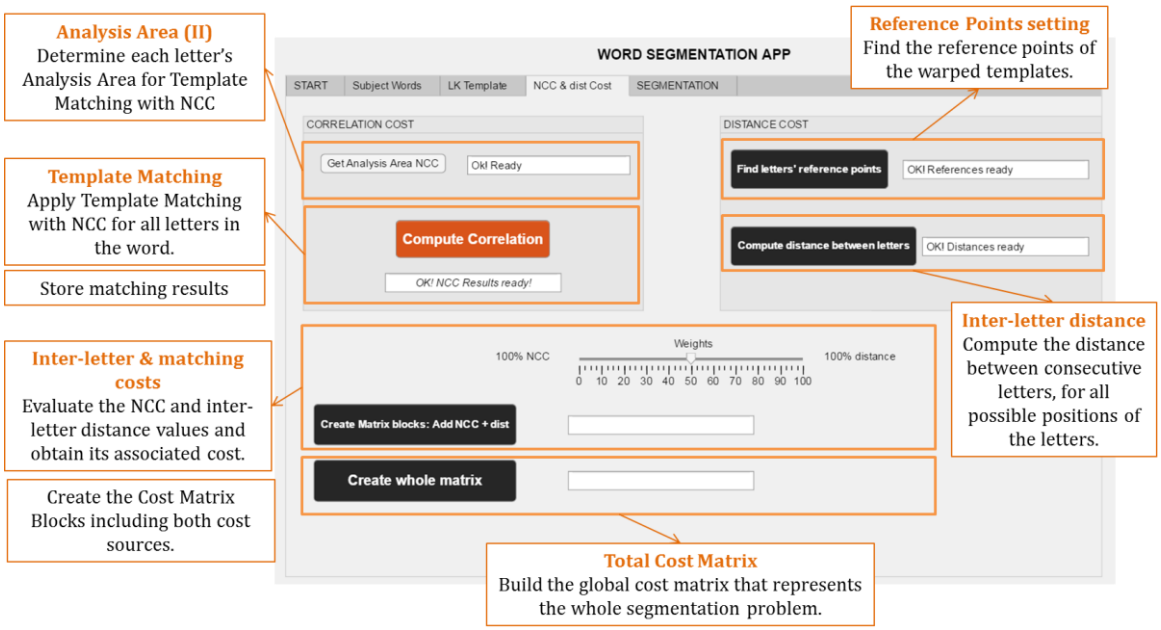


Figure F- 8: NCC & dist. Cost Tab from the Word Segmentation App

F.3.4. SEGMENTATION Tab

The ***NCC & dist. Cost Tab*** is divided in 3 main areas:

1) Word Segmentation: Compute graph-based segmentation results as start and end points of each letter in the word.

   a. Compute shortest path: Obtain graph-segmentation result by means of Dijkstra shortest path algorithm. The solution is the set of the image locations where the warped templates must be placed for optimal matching, given as coordinates in each AA.

   b. Find Segmentation points: Convert the shortest path results' coordinates into global coordinates and obtain the start and end reference points of the templates on the image.

   c. Plot Segmentation: Find the closest point on the word contour for each template start and end point. Plot solution according to desired visualization (dots, lines, bounding box…)

2) Results area: Visualize the results according to the choice in (1.c). Also save the results in the folder create during initialization as (.tif) image. The results are available after the App is closed as well (also the binary images of the words).

3) Restart segmentation: Got to ***Subject Words Tab*** and select a new word to start again.

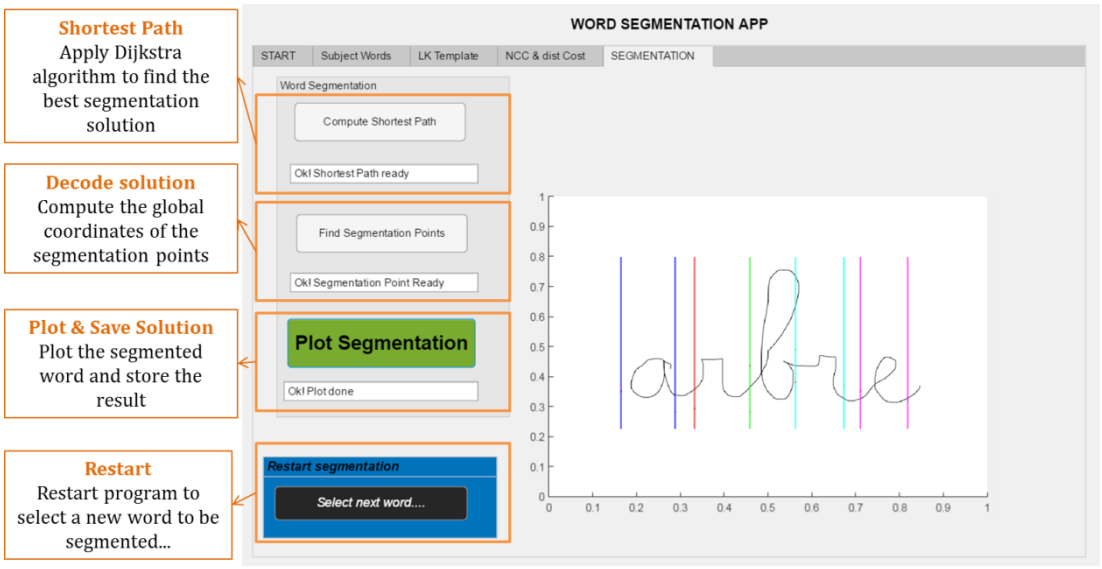


Figure F- 9: SEGMENTATION Tab from the Word Segmentation Tab

To select a new CSV with WACOM samples that belong to a new Subject's handwriting, the user must go back to the START Tab.

9 BIBLIOGRAPHY

[1] Van Reybroeck, M., De Vleeschouwer, C., Gosse, C. (2017). Do children with dyslexia present a handwriting deficit? A group comparison experiment. *UCL docs.*

[2] Gosse, C., Carbonnelle, S., Van Reybroeck, M., De Vleeschouwer. (2016). Graphic Characteristics of Words That Influence Children's Handwriting. *Psychological Sciences Research Institute; Université Catholique de Louvain.*

[3] Perera, J., Aparici, M., Rosado, E. & Salas, N. (Eds.) (2016). Written and Spoken Language Development across the Lifespan: Essays in honour of Liliana Tolchinsky. *Cham, Switzerland: Springer International Publishing | ISBN-10: 3319211358.*

[4] Kandel, S., & Perret, C. (2015). How does the interaction between spelling and motor processes build up during writing acquisition? *Cognition, 136, 325-336. doi:10.1016/j.cognition.2014.11.014.*

[5] Sébastien Roux, Thomas J. McKeeff, Géraldine Grosjacques, Olivia Afonso, Sonia Kandel (2012). The interaction between central and peripheral processes in handwriting production. *Brief article, Cognition, 127 (2013) 235–241. doi.org/10.1016/j.cognition.2012.12.009.*

[6] Mellissa Prunty & Anna L. Barnett (2017). Understanding handwriting difficulties: A comparison of children with and without motor impairment. *Cognitive Neuropsychology, doi: 10.1080/02643294.2017.1376630.*

[7] Connelly, V., Campbell, S., MacLean, M., & Barnes, J. (2010). Contribution of Lower Order Skills to the Written Composition of College Students With and Without Dyslexia. *Developmental Neuropsychology, 29:1, 175-196, DOI: 10.1207/s15326942dn2901_9.*

[8] Sumner, E., Connelly, V., & Barnett, A. L. (2014). The influence of spelling ability on handwriting production: Children with and without dyslexia. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 40(5), 1441-1447. doi.org/10.1037/a0035785.*

[9] Van Galen, G. P. (1991). Handwriting: Issues for a psychomotor theory. *Human Movement Science, 10, 165–191.*

[10] Roux, J.-S., McKeeff, T. J., Grosjacques, G., Afonso, O., & Kandel, S. (2013). The interaction between central and peripheral processes in handwriting production. *Cognition, 127, 235–241.*

[11] Damian, M. F., & Stadthagen-Gonzalez, H. (2009). Advance planning of form properties in the written production of single and multiple words. *Language and Cognitive Processes, 24, 555–579.*

[12] Delattre, M., Bonin, P., & Barry, C. (2006). Written spelling to dictation: Do irregularity effects persist on writing durations? *Journal of Experimental Psychology: Learning, Memory, and Cognition, 32, 1330–1340.*

[13] Kandel, S., Peereman, R., & Ghimenton, A. (2013). Further evidence for the interaction between central and peripheral processes: The impact of double in writing English words. *Frontiers in Psychology, 4, 729. http://dx.doi.org/10.3389/fpsyg.2011.00729*

[14] Lambert, E., Alamargot, D., Larocque, D., & Caporossi, G. (2011). Dynamics of the spelling process during a copying task: Effects of regularity and frequency. *Canadian Journal of Experimental Psychology, 65(3), 141–150.*

[15] Gosse, C. Van Reybroeck, M., Carbonnelle, S., De Vleeschouwer, C. (2018). Specifying the graphic characteristics of words that influence children's handwriting, co *Vol. 1, no.1, p. 1-27 (2018). doi:10.1007/s11145-018-9834-9.*

[16] Goswami, R., Sharma, O.p. (2013). A Review on Character Recognition Techniques. International Journal of Computer Applications (0975 – 8887). Volume 83 – No 7, December 201.

[17] Shankar Rao, P., Aditya, J. (2005). Handwriting Recognition: "Offline" Approach. *Modern Trends in Intelligent Computing Systems (Nasmotics).*

[18] Laxmi Sahu, V., Kubde, B. (2013). Offline Handwritten Character Recognition Techniques using Neural Network: A Review. *International Journal of Science and Research (IJSR), India Online ISSN: 2319-706.*

[19] Khandelwal, H., Gupta, S., Kumar Jain, A. (2017). Review of Offline Handwriting Recognition Techniques in the fields of HCR and OCR. *International Journal of Computer Trends and Technology (IJCTT) V47 (3):161-164, May 2017. ISSN:2231-2803. www.ijcttjournal.org.*

[20] Kala, R., Vazirani, H., Shukla, A., Tiwari, R. (2010). Offline Handwriting Recognition using Genetic Algorithm. *International Journal of Computer Science Issues (IJCSI), Vol. 7, Issue 2, No 1. ISSN (Print): 1694-0814.*

[21] Weisinger, D. (2017). Handwriting Recognition: Digitizing Text from Handwritten Paper Documents. *Formtek Management Blog – Document Management. http://formtek.com/blog/handwriting-recognition-digitizing-text-from-handwritten-paper-documents/*

[22] Nield, D. (2017). How to digitize your handwritten notes. Make your scribbles searchable. *Popular Science, DIY, Bonnier Corporation. https://www.popsci.com/digitize-handwritten-notes*

[23] Livescribe official website: *https://www.livescribe.com/es/smartpen/ls3/*

[24] Moleskine official website. *https://us.moleskine.com/en/*

[25] Neo SmartPen official website: *https://www.neosmartpen.com/en/*

[26] WACOM official website*https://www.wacom.com/en-us*

[27] *Myscript* official website: *https://www.myscript.com/*

[28] CSV splitter official website: *https://download.cnet.com/CSV-Splitter/3000-2074_4-75910188.html*

[29] Ledda, A. (2007). Mathematical Morphology in Image Processing. *Faculteit Ingenieurswetenschappen, Universiteit Gent. Wettelijk depot: D/2007/10.500/1*.

[30] Bebis, G. (2001). 2D/3D Geometric Transformations. *CS485/685 Computer Vision.*

[31] Coste, A. (2012). Affine Transformation, Landmarks registration, Nonlinear Warping. CS*6640: Image Processing (Project 3). Scientific Computing And Imaging Institute.*

[32] Rhody, H. (2005). Geometric Image Transformations. *Chester F. Carlson Centre for Imaging Science Rochester Institute of Technology.*

[33] Collins, R. (2006). Template matching, warps, and Lucas-Kanade. *CSE Department, Penn State University. CSE598G Vision-Based Tracking.*

[34] Collins, R. (2006). More on Lucas-Kanade. Background alignment. *CSE Department, Penn State University. CSE598G Vision-Based Tracking.*

[35] Lucas, B.D., Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI). *Conference Paper, Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81), pp. 674-679.*

[36] Baker, S., Gross, R., Matthews, I. (2003). Lucas-Kanade 20 years on: A unifying framework. *Carnegie Mellon University: Research Showcase. CMU-RI-TR-03-35*.

[37] Swaroop, P., Sharma, N. (2016). An Overview of Various Template Matching Methodologies in Image Processing. *International Journal of Computer Applications (0975 – 8887). Volume 153 – Number 10.*

[38] Garg, P. (2017). Programming Tutorials and Practice Problems: Graph Representation. *HackerEarth. https://www.hackerearth.com/*

[39] Northeastern University, College of Computer and Informatic Science. Fundamentals II: Introduction to Class-based Program Design (2016) *https://course.ccs.neu.edu/cs2510h/lecture30.html*

[40] Briechle, K., D. Hanebeck, U. (2001). Template Matching using Fast Normalized Cross Correlation. *Proc. SPIE 4387, Optical Pattern Recognition XII doi: 10.1117/12.421129; https://doi.org/10.1117/12.421129.*

[41] Luo, J., Konofagou, E.E. (2010). A fast normalized cross-correlation calculation method for motion estimation. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 57, no. 6, pp. 1347-1357, doi: 10.1109/TUFFC.2010.1554*

[42] Lewis, J.P. (2001). Fast Normalized Cross-Correlation. *Ind. Light Magic. 10. Vision Interface 95, Canadian Image Processing and Pattern Recognition Society, p. 120-123.*

[43] E. Usoro, A. (2015). Some basic properties of cross-correlation functions of n-dimensional vector time series. *Journal of Statistical and Econometric Methods, vol.4, no.1, 2015, 63-71 ISSN: 2241-0384), Scienpress Ltd.*

[44] Elliott, D.F., and Rao, K.R. (1982). Fast Transforms: Algorithms, Analyses, Applications. *New York: Academic Press.*

[45] Brigham, E.O. (1974). The Fast Fourier Transform. *Englewood Cliffs, NJ: Prentice-Hall, Chapter 13.*

[46] C. Seiler M., A. Seiler F. (1989). Numerical Recipes in c: the art of scientific computing. *Volume 9, Issue 3. pp. 415-416 https://doi.org/10.1111/j.1539-6924.1989.tb01007.x*

[47] Eaton, D. (2018). How to do Normalized Cross-Correlation Fast. *Department of Computer Science, University of British Columbia. 604-822-662, deaton@cs.ubc.ca*

[48] Zhao, F., Huang, Q. Gao, W. (2006). Image Matching by Normalized Cross-Correlation," *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, Toulouse, 2006, pp. II-II. doi: 10.1109/ICASSP.2006.1660446*

[49] *Mohamad, B., Yaakob, S., A. Raof, R.A., Nazren, A., Nasrudin, M. W. (2015).* Template Matching using Sum of Squared Difference and Normalized Cross Correlation. *100-104. 10.1109/SCORED.2015.7449303.*

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgium **www.uclouvain.be/epl**