Contributions to Pen & Touch Human-Computer Interaction

PhD Thesis

Daniel Martín-Albo

Supervised by Enrique Vidal Ruiz
and Verónica Romero Gómez

May, 2016

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Work partially supported by the Spanish MEC under FPU scholarship (AP2010-0575), the EU's 7th Framework Programme under tranScriptorium grant agreement (FP7/2007-2013/600707), and the Spanish MINECO and the European RDF under the STraDA grant (TIN2012-37475-C02-01).

No man was more foolish when he had not a pen in his hand, or more wise when he had.

— Samuel Johnson, *Of Goldsmith*

Abstract

Computers are now present everywhere, but their potential is not fully exploited due to some lack of acceptance. In this thesis, the *pen computer* paradigm is adopted, whose main idea is to replace all input devices by a pen and/or the fingers, given that the origin of the rejection comes from using unfriendly interaction devices that must be replaced by something easier for the user.

This paradigm, that was was proposed several years ago, has been only recently fully implemented in products, such as the smartphones. But computers are actual illiterates that do not understand gestures or handwriting, thus a recognition step is required to "translate" the meaning of these interactions to computer-understandable language. And for this input modality to be actually usable, its recognition accuracy must be high enough.

In order to realistically think about the broader deployment of pen computing, it is necessary to improve the accuracy of handwriting and gesture recognizers. This thesis is devoted to study different approaches to improve the recognition accuracy of those systems.

First, we will investigate how to take advantage of interaction-derived information to improve the accuracy of the recognizer. In particular, we will focus on interactive transcription of text images. Here the system initially proposes an automatic transcript. If necessary, the user can make some corrections, implicitly validating a correct part of the transcript. Then the system must take into account this validated prefix to suggest a suitable new hypothesis. Given that in such application the user is constantly interacting with the system, it makes sense to adapt this interactive application to be used on a pen com-

puter. User corrections will be provided by means of pen-strokes and therefore it is necessary to introduce a recognizer in charge of decoding this king of nondeterministic user feedback. However, this recognizer performance can be boosted by taking advantage of interaction-derived information, such as the user-validated prefix.

Then, this thesis focuses on the study of human movements—in particular, hand movements—from a generation point of view by tapping into the kinematic theory of rapid human movements and the Sigma-Lognormal model. Understanding how the human body generates movements and, particularly understand the origin of the human movement variability, is important in the development of a recognition system. The contribution of this thesis to this topic is important, since a new technique (which improves the previous results) to extract the Sigma-lognormal model parameters is presented.

Closely related to the previous work, this thesis study the benefits of using synthetic data as training. The easiest way to train a recognizer is to provide "infinite" data, representing all possible variations. In general, the more the training data, the smaller the error. But usually it is not possible to infinitely increase the size of a training set. Recruiting participants, data collection, labeling, etc., necessary for achieving this goal can be time-consuming and expensive. One way to overcome this problem is to create and use synthetically generated data that looks like the human. We study how to create these synthetic data and explore different approaches on how to use them, both for handwriting and gesture recognition.

The different contributions of this thesis have obtained good results, producing several publications in international conferences and journals.

Finally, three applications related to the work of this thesis are presented. First, we created Escritorie, a digital desk prototype based on the pen computer paradigm for transcribing handwritten text images. Second, we developed "Gestures à Go Go", a web application for bootstrapping gestures. Finally, we studied another interactive application under the pen computer paradigm. In this case, we study how translation reviewing can be done more ergonomically using a pen.

Hoy en día, los ordenadores están presentes en todas partes pero su potencial no se aprovecha debido al "miedo" que se les tiene. En esta tesis se adopta el paradigma del *pen computer*, cuya idea fundamental es sustituir todos los dispositivos de entrada por un lápiz electrónico o, directamente, por los dedos. El origen del rechazo a los ordenadores proviene del uso de interfaces poco amigables para el humano.

El origen de este paradigma data de hace más de 40 años, pero solo recientemente se ha comenzado a implementar en dispositivos móviles. La lenta y tardía implantación probablemente se deba a que es necesario incluir un reconocedor que "traduzca" los trazos del usuario (texto manuscrito o gestos) a algo entendible por el ordenador.

Para pensar de forma realista en la implantación del pen computer, es necesario mejorar la precisión del reconocimiento de texto y gestos. El objetivo de esta tesis es el estudio de diferentes estrategias para mejorar esta precisión.

En primer lugar, esta tesis investiga como aprovechar información derivada de la interacción para mejorar el reconocimiento, en concreto, en la transcripción interactiva de imágenes con texto manuscrito. En la transcripción interactiva, el sistema y el usuario trabajan "codo con codo" para generar la transcripción. El usuario valida la salida del sistema proporcionando ciertas correcciones, mediante texto manuscrito, que el sistema debe tener en cuenta para proporcionar una mejor transcripción. Este texto manuscrito debe ser reconocido para ser utilizado. En esta tesis se propone aprovechar información contextual—como por ejemplo, el prefijo validado por el usuario—para mejorar la calidad del reconocimiento de la interacción.

Tras esto, la tesis se centra en el estudio del movimiento humano—en particular del movimiento de las manos— utilizando la Teoría Cinemática y su modelo Sigma-Lognormal. Entender como se mueven las manos al escribir, y en particular, entender el origen de la variabilidad de la escritura, es importante para el desarrollo de un sistema de reconocimiento, La contribución de esta tesis a este tópico es im-

portante, dado que se presenta una nueva técnica (que mejora los resultados previos) para extraer el modelo Sigma-Lognormal de trazos manuscritos.

De forma muy relacionada con el trabajo anterior, se estudia el beneficio de utilizar datos sintéticos como entrenamiento. La forma más fácil de entrenar un reconocedor es proporcionar un conjunto de datos "infinito" que representen todas las posibles variaciones. En general, cuanto más datos de entrenamiento, menor será el error del reconocedor. No obstante, muchas veces no es posible proporcionar más datos, o hacerlo es muy caro. Por ello, se ha estudiado como crear y usar datos sintéticos que se parezcan a los reales.

Las diferentes contribuciones de esta tesis han obtenido buenos resultados, produciendo varias publicaciones en conferencias internacionales y revistas.

Finalmente, también se han explorado tres aplicaciones relaciones con el trabajo de esta tesis. En primer lugar, se ha creado Escritorie, un prototipo de mesa digital basada en el paradigma del pen computer para realizar transcripción interactiva de documentos manuscritos. En segundo lugar, se ha desarrollado "Gestures à Go Go", una aplicación web para generar datos sintéticos y empaquetarlos con un reconocedor de forma rápida y sencilla. Por último, se presenta un sistema interactivo real bajo el paradigma del pen computer. En este caso, se estudia como la revisión de traducciones automáticas se puede realizar de forma más ergonómica.

Avui en dia, els ordinadors són presents a tot arreu i es comunament acceptat que la seva utilització proporciona beneficis, tant a individus com a organitzacions. No obstant això, moltes vegades el seu potencial no s'aprofita totalment. Això probablement es degui al rebuig d'algunes persones cap als ordinadors, a causa de la "por" a utilitzar-los, tot i incrementar la seva productivitat.

En aquesta tesi s'adopta el paradigma del *pen computer*, on la idea fonamental és substituir tots els dispositius d'entrada per un llapis

electrònic, o, directament, pels dits. Aquest paradigma postula que l'origen del rebuig als ordinadors prové de l'ús d'interfícies poc amigables per a l'humà, que han de ser substituïdes per alguna cosa més coneguda. Per tant, la interacció amb l'ordinador sota aquest paradigma es realitza per mitjà de text manuscrit i/o gestos.

L'origen d'aquest paradigma data de fa més de 40 anys, però només recentment s'ha començat a implementar en dispositius mòbils. La lenta i tardana implantació probablement es degui al fet que és necessari incloure un reconeixedor que "tradueixi" els traços de l'usuari (text manuscrit o gestos) a alguna cosa comprensible per l'ordinador, i el resultat d'aquest reconeixement, actualment, és lluny de ser òptim.

Per pensar de forma realista en la implantació del pen computer, cal millorar la precisió del reconeixement de text i gestos. L'objectiu d'aquesta tesi és l'estudi de diferents estratègies per millorar aquesta precisió.

En primer lloc, aquesta tesi investiga com aprofitar informació derivada de la interacció per millorar el reconeixement, en concret, en la transcripció interactiva d'imatges amb text manuscrit. En la transcripció interactiva, el sistema i l'usuari treballen "braç a braç" per generar la transcripció. L'usuari valida la sortida del sistema donant certes correccions, que el sistema ha d'usar per millorar la transcripció. En aquesta tesi es proposa utilitzar correccions manuscrites, que el sistema ha de reconèixer primer. La qualitat del reconeixement d'aquesta interacció és millorada, tenint en compte informació contextual, com per exemple, el prefix validat per l'usuari.

Després d'això, la tesi se centra en l'estudi del moviment humà—en particular del moviment de les mans—des del punt de vista generatiu, utilitzant la Teoria Cinemàtica i el model Sigma-Lognormal. Entendre com es mouen les mans en escriure és important per al desenvolupament d'un sistema de reconeixement, en particular, per entendre l'origen de la variabilitat de l'escriptura. La contribució d'aquesta tesi a aquest tòpic és important, atès que es presenta una nova tècnica (que millora els resultats previs) per extreure el model Sigma-Lognormal de traços manuscrits.

De forma molt relacionada amb el treball anterior, s'estudia el benefici d'utilitzar dades sintètiques per a l'entrenament. La forma més fàcil d'entrenar un reconeixedor és proporcionar un conjunt de dades "infinit" que representin totes les possibles variacions. En general, com més dades d'entrenament, menor serà l'error del reconeixedor. No obstant això, moltes vegades no és possible proporcionar més dades, o fer-ho és molt car. Per això, s'ha estudiat com crear i utilitzar dades sintètiques que s'assemblin a les reals.

Les diferents contribucions d'aquesta tesi han obtingut bons resultats, produint diverses publicacions en conferències internacionals i revistes.

Finalment, també s'han explorat tres aplicacions relacionades amb el treball d'aquesta tesi. En primer lloc, s'ha creat Escritorie, un prototip de taula digital basada en el paradigma del pen computer per realitzar transcripció interactiva de documents manuscrits. En segon lloc, s'ha desenvolupat "Gestures à Go Go", una aplicació web per a generar dades sintètiques i empaquetar-les amb un reconeixedor de forma ràpida i senzilla. Finalment, es presenta un altre sistema interactiu sota el paradigma del pen computer. En aquest cas, s'estudia com la revisió de traduccions automàtiques es pot realitzar de forma més ergonòmica.

Acknowledgments

Ahora que empiezo a ver la claridad del final de este túnel (como diría mi padre) me gustaría agradecer el apoyo de todos aquellos, que de una forma u otra, han ayudado a lo largo de este viaje.

En primer lugar, me gustaría dar las gracias a mis directores. A **Enrique Vidal**, que me dio la oportunidad de dedicarme a la investigación y a **Verónica Romero**, que sin su guía y entusiasmo esta tesis no hubiera sido posible.

I would like to thank **Réjean Plamondon** for welcoming me to his group. Despite the weather I had an unforgettable time in Montreal, while I developed what is now a very important part of this thesis. *Merci beaucoup*.

Many thanks to the reviewers. Their comments and suggestions have positively contributed to the final state of this document.

Me gustaría agradecer a mis compañeros y (ex-compañeros) del PRHLT por el extraordinario ambiente de trabajo de estos años. En particular, un conjunto de personas merecen una mención en especial ya que algunos de ellos ha tenido un impacto directo en esta tesis (pido disculpas de antemano por las posibles omisiones): A **Luis Leiva** que ha sido una constante referencia a lo largo de estos años. A **Paco Álvaro** y **Vicente Bosch**, mis pacientes vecinos durante muchos años. A **Joan Puigcerver**, **Nico Serrano**, **Dani Ortiz** y **Jesús González** por sus siempre valiosos consejos.

Fuera del mundo de la investigación, me gustaría agradecer a **Víctor Carrión** por su amistad, por los cafés y New York cheesecakes.

Finalmente, me gustaría dedicar esta tesis a mis **padres**, a mi **hermana** y a mi **hermano**.

A todos vosotros, muchas gracias.

Valencia, Mayo de 2016

Contents

1 Introduction

Computers are now present everywhere and it is widely accepted that they provide benefits to both individuals and organizations, but it is recognized that their potential is not fully exploited due to some lack of acceptance. Some people are afraid to use a computer even if it appears to increase productivity: *"Many workers are suspicious of new technology, even hostile to it"* [1]. According to Igbaria et al. [2], the perceived usefulness of computers is strongly correlated with their perceived ease of use. Given this, the ease at which information is exchanged between a person and a computer becomes a serious topic. In general, user-computer interaction is expected to be efficient and ergonomic. During the time of evolving computer technology, many different means for user input have been invented: keyboards, mice, trackballs, etc., and nowadays, the primary mode of interacting with a computer is the keyboard, which is quite efficient, but not really ergonomic. Keyboard is not really fitted for the human way of communicating thoughts and ideas.

A pen is more convenient than a keyboard: of all the means of exchange, writing is probably the most precise and comfortable, as well as the most flexible. In other words: *"Use your primary and best developed human skill, writing by hand"* [3]. This idea led Alan Kay to coin the *pen computing* paradigm, by building the Dynabook in 1968, which is a prototype that could be used effortlessly by untrained users. Alan Kay considered the keyboard as an expensive and non-ergonomic component to be replaced by a pentip position sensitive surface superimposed on a graphic display that generates electronic ink [4]. In practical terms, a pen computer consists of a flat display which records and displays the traces from a user's moving pen. The

pen replaces a keyboard and any other input devices and is also able to recognize handwritten text—it can convert strokes to printed text [3].

The original pen computing definition focused on a pen or digital stylus as the implement of articulation. However, the influence of Apple Inc.—and its iPhone[1] and iPad— has made the pen computing paradigm to change and to also include finger-based interactions. With finger-based interactions there is no device acquisition time (the time to pull out and grasp the pen) so it is more direct and more convenient to use. On the other hand, the finger obscures more the screen surface and is less dexterous and precise than a pen. It is possible to enable both finger and pen interactions on the same device. For a small amount of imprecise drawing, the hurried user should be able to use its bare finger. In contrast, if the user is going to handwrite, one should be able to switch to a pen, given that using a pen involves more joints of the hand and offers higher precision and dexterity than using a single finger [5]. In other words, gestures are more easily drawn *touching* with a finger, but it is easier to write using a *pen*. This is the reason for including both *pen* and *touch*, instead of just pen, in the title of this thesis. They are different problems and therefore must be tackled using different approaches.

Although the pen computer term[2] was proposed over 40 years ago, only recently it has been fully implemented in consumer products, such as the smartphones and tablets. These devices can be controlled now using handwriting and gestures. Among the reasons that could explain this slow implantation, one probably stands out. Computers are actual illiterates that do not understand the most simple strokes of a gesture, not to mention the complex strokes of handwriting [3]. Thus, unlike keyboard, pen and touch interactions require a recognition step to decode their non-deterministic meaning. And for this input modality to be actually usable, its recognition accuracy must be high enough. Humans seem to carry out recognition effortlessly—achieving low error—, but even though in recent years the technology has progressed this is not the case for comput-

[1] During the iPhone presentation, Steve Jobs jokingly said: *"Who wants a stylus?"*.

[2] In the following, the pen computer term will be used for describing a machine that is primarily or exclusively operated using a pen and/or a finger.

ers. Computers still do not obtain as good results as those obtained by humans and thus there is room for improvement.

In order to realistically think about the broader deployment of pen computing, it is necessary to solve the main reason that causes the user rejection and thus its slow implantation. This thesis is devoted to study different approaches and strategies to improve the pen computer recognition accuracy. In particular, we will focus on improving the accuracy of recognizing handwriting and gestures, which are probably the most common ways of interacting with a pen computer.

First, we will investigate how to take advantage of interaction-derived information to improve the accuracy of the recognizer. In particular, we will focus on interactive transcription of text images. Here the system initially proposes an automatic transcript. If necessary, the user can make some corrections, implicitly validating a correct part of the transcript. Then the system must take into account this validated prefix to suggest a suitable new hypothesis. Given that in such application the user is constantly interacting with the system, it makes sense to adapt this interactive application to be used on a pen computer. User corrections will be provided by means of pen-strokes and therefore it is necessary to introduce a recognizer in charge of decoding this king of nondeterministic user feedback. However, this recognizer performance can be boosted by taking advantage of interaction-derived information, such as the user-validated prefix [6,7].

Then, we will focus on the study of human movements—in particular, hand movements—from a generation point of view. Understanding how the human body generates movements is important in the development of a recognition system, particularly to understand the origin of the human movement variability. In this case, we tap into the kinematic theory of rapid human movements, which provides the best performance in reconstructing human movements [8].

Finally, we investigate the benefits of using synthetic data to enlarge the training data. Although there are some basic forms that everybody learned, every writer has a personal and different writing style. The easiest way of training a recognizer is to provide a huge set of training data, representing all possible variations. Therefore, a recognizer accuracy heavily depends on the size of the training set

used, according to the rule of thumb: the bigger, the better. But usually it is not possible to infinitely increase the size of a training set. Recruiting participants, data collection, labeling, etc., necessary for achieving this goal can be time-consuming and expensive. One way to overcome this problem is to create and use synthetically generated data that *look and feel* like human. We study how to create these synthetic data and explore different approaches on how to use them, both for handwriting and gesture recognition.

This thesis is organized in seven chapters. Chapter 2 is an overview of the handwriting and gestures recognition formal framework. Here the different recognition techniques (as well as the assessment measures used to evaluate their performance) used in this thesis are presented and explained. The overview of these techniques have been gathered in one chapter, given that most of these techniques are used throughout the entire thesis, rather than in an isolated chapter.

Chapters 3 to 5 have been conceived as units that relate to the central topic of this thesis: contributions to improve the recognition accuracy of a pen computer. These chapters enclose the scientific goals previously presented and, to facilitate reading, are structured similarly. First section is an introduction to the topic. Previous related works, as well as their formal framework, are then reviewed. The subsequent sections are focused on developing and testing the different ideas proposed in this thesis.

Chapter 3 is focused on the interactive transcription of handwritten text images. This chapter discusses how interaction-derived information can provide leverage to improve the feedback recognition accuracy. In this chapter, two different approaches are presented and evaluated.

Chapter 4 reviews the kinematic theory of rapid human movements and its associated models. A new Sigma-Lognormal parameters extractor is presented and evaluated.

Chapter 5 studies the origin of human movement variability tapping into the Kinematic Theory. A synthetic handwriting and gestures generation technique is reviewed. After that, different uses of this technique are proposed and evaluated.

Chapter 6 presents three applications of the developments of this thesis. First, a pen computer prototype focused on transcribing handwritten documents is developed. After that, an application for synthesizing and bootstrapping gestures is presented. Finally, we show a real application of an interactive system where interaction-derived information is exploited to achieve better feedback recognition accuracy, in this case, used for translation reviewing.

Chapter 7 summarizes the contributions of this thesis and presents the publications resulted from this work.

2 Overview of Pen & Touch Recognition

In terms of pen computing and in addition to handwriting, a user should be able to interact with non-textual feedback—graphics and drawings. A way of issuing commands, selecting menu items or icons must be provided. However, the meaning of most of these interactions is unclear for the computer. In the past, many approaches have been presented to tackle with their recognition, but all the approaches share the same goal: to give a computer-understandable meaning to the input data.

As previously stated, in this thesis we will focus on 2 types of input data: handwriting and gestures. Handwritten text recognition (HTR) will be outlined in Section 2.1. After that, in Section 2.2 gesture recognition will be reviewed. Finally, this chapter concludes with the description of the different datasets that will be used in the experiments of this thesis.

2.1 Handwritten Text Recognition

HTR is the ability of a computer to receive and interpret intelligible handwritten input from sources such as stylus, touch-screens and paper documents or pictures. The introduction of HTR into commercial products took place in early 1980s, when products such as the Pencept Penpad or the Inforite point-of-sale terminal [9] were presented. These products incorporated HTR as a replacement for keyboard input. Later, advancements in electronics allowed the computing power necessary for HTR to fit into a smaller devices, being used as input method for PDAs. For example, the Apple Newton was one of the first PDAs to provide a handwriting recognition system.

HTR is classified into two different categories: on-line (such as the devices previously highlighted) and off-line. On-line HTR involves the automatic conversion of text as it is handwritten on a special digitizer or tablet, using either an pen or a finger. Here the sensor picks up the pen or finger-tip movements, as well as pen-up/pen-down switching. That is, the trace of the handwriting or line drawing is captured as a sequence of coordinate points with stroke separation indications.

Alternatively, off-line HTR involves the automatic conversion of text in an image into letter codes which are usable within computer and text-processing applications. The data obtained by this form is regarded as a static representation of handwriting. This technique cannot use any dynamic writing information, such as the number of pen-strokes, their order and their direction of creation.

Both on-line and off-line can be solved in a similar manner, by using a recognition system composed of three modules. That is, *preprocessing*, *feature extraction* and *recognition*. Here, the preprocessing module is in charge of reducing spurious noise, whilst the feature extraction transforms a preprocessed text into a sequence of numbers. Finally, the recognition module transforms this sequence of numbers into computer-understandable letter codes. The first 2 modules entails different techniques depending on the nature of the data—on-line or off-line—, however, the last module can be identical.

2.1.1 ON-LINE HTR SYSTEM

On-line HTR refers to dealing with the automatic transcription of a message written using a digitizer or an stylus or pen. The device captures, synchronously or asynchronously, the spatial information (x,y) with respect to a particular temporal instant. This data is accompanied by information about the pressure, or sometimes just whether there is contact with the surface (or not).

As previously stated, the architecture adopted to carry out the on-line HTR is formed by 3 modules: preprocessing, feature extraction and recognition. The following describes these 3 modules.

2.1.1.1 *Preprocessing*

Prior to any recognition, the acquired data is generally preprocessed to reduce irrelevant information and noise. Here, the preprocessing involves two steps: spurious points elimination and noise reduction. Sometimes, repeated points appear in a trajectory because the pen is motionless. These points are trivially removed. In addition, the points with null pressure (or marked as *pen-up*) are also removed. Noise in the captured data can be due to erratic hand motion and/or inaccuracy of the capture device. This noise is reduce by replacing every point by the mean value of its neighbors [10].

2.1.1.2 *Feature Extraction*

For each point $p_t = (x(t), y(t))$, we compute six time-based features [11].

Normalized vertical position The coordinates $x(t)$ and $y(t)$ are normalized and translated such that $y \in [0, 100]$ and the aspect ratio is preserved.

Normalized first derivatives The first derivatives $x'(t)$ and $y'(t)$ are calculated using:

$$\begin{aligned} x'(t) &= \frac{\sum_{i=1}^{r} x(t+r) - x(t-r)}{\|\nabla\|} \\ y'(t) &= \frac{\sum_{i=1}^{r} y(t+r) - y(t-r)}{\|\nabla\|} \end{aligned} \tag{2.1}$$

being

$$\|\nabla\| = \sqrt{\Delta x^2 + \Delta y^2} \tag{2.2}$$

where r defines a sliding window of size $2r + 1$ that determines the neighbors involved in the calculation, Δx and Δy are, for each coordinate, the distance between the smallest and largest points for the given sliding window. Toselli et al. [12] have proved that $r = 2$ is a good value to be used.

Normalized second derivatives The second derivatives are calculated in the same manner as the first derivatives, replacing $x(t)$ by $x'(t)$

and $y(t)$ by $y'(t)$:

$$\begin{aligned} x''(t) &= \frac{\sum_{i=1}^{r} x'(t+r) - x'(t-r)}{\|\nabla\|} \\ y''(t) &= \frac{\sum_{i=1}^{r} y'(t+r) - y'(t-r)}{\|\nabla\|} \end{aligned} \tag{2.3}$$

It should be noted that normalizing by $\|\nabla\|$ entails an implicit writing speed normalization. Toselli et al. [12] proves that this technique leads to better results than using explicit speed preprocessing techniques.

Curvature The inverse of the radius of the curve in each point:

$$k(t) = \frac{x'(t)\, y''(t) - y'(t)\, x''(t)}{(x'(t)^2 + y'(t)^2)^{3/2}} \tag{2.4}$$

Finally, for each point p_t we obtain a feature vector f_t:

$$f_t = [x(t),\, y(t),\, x'(t),\, y'(t),\, x''(t),\, y''(t),\, k(t)] \tag{2.5}$$

2.1.1.3 Recognition

An on-line handwritten word (or sequence of words) is represented by a sequence of any size of feature vectors, $\mathbf{x} = (f_1 f_2 \dots f_M)$. The HTR problem can be formulated as finding the most likely word (or sequence of words) $\mathbf{w}$ given the feature vector sequence $\mathbf{x}$. That is:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \Pr(\mathbf{w} \mid \mathbf{x}) \tag{2.6}$$

using the Bayes' rule

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \frac{\Pr(\mathbf{x} \mid \mathbf{w}) \Pr(\mathbf{w})}{\Pr(\mathbf{x})} \tag{2.7}$$

where $Pr(\mathbf{x})$ can be drop since it is constant for any $\mathbf{w}$:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \Pr(\mathbf{w}) \cdot \Pr(\mathbf{x} \mid \mathbf{w}) \tag{2.8}$$

In this expression, $\Pr(\mathbf{w})$ represents a syntactic knowledge and $\Pr(\mathbf{x} \mid \mathbf{w})$ an *optical* or morphological knowledge.

Different approaches can be used to model the optical knowledge, such as long short-term memory (LSTM) RNN [13] or hidden Markov models (HMM) [14]. In this thesis, each character is modeled by continuous density left-to-right HMM, with a Gaussian mixture per state. This mixture serves as a probabilistic law to the emission of feature vectors on each model state. A HMM is a statistical model such that if we use them to predict the next observation in a sequence, the distribution of predictions will depend only on the value of the immediately preceding observation and will be independent of all earlier observations [15]. Given a set of observations, the parameters of a HMM can be estimated using maximum likelihood by means of the Baum-Welch algorithm [16]. Morphological word models are built from respective lexical entries, modeled by a stochastic finite-state automaton which represents all possible concatenations of individual characters that may compose the word. Finally, by embedding the character HMMs into the edges of the word automaton, a lexical HMM is obtained. These HMMs estimate the word-conditional probabilities $\Pr(\mathbf{x} \mid \mathbf{w})$ of Equation (2.8)

An n-gram language model is employed here to approximate the linguistic knowledge [14]. It predicts the following word by taking into account the previous $n-1$ words. For a word sequence $\mathbf{w}$:

$$\Pr(\mathbf{w}) \approx \prod_{i=1}^{N} P(w_i \mid \mathbf{w}_{i-n+1}^{i-1}) \tag{2.9}$$

All the above refers to recognition at word level, namely, the minimum unit that the recognizer is able to return is a word. The formulation can be easily adapted to recognize characters. This is done by replacing $\mathbf{w}$ with $\mathbf{c}$ in Equation (2.8). Then the problem can be now formulated as finding the most likely character $\mathbf{c}$ (or sequence of characters) given $\mathbf{x}$:

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}} \Pr(\mathbf{c}) \cdot \Pr(\mathbf{x} \mid \mathbf{c}) \tag{2.10}$$

Following the previous approach, $\Pr(\mathbf{c})$ is approximated using n-grams, whilst $\Pr(\mathbf{x} \mid \mathbf{c})$ is approximated using HMMs. In this case, the n-gram is also responsible for modeling the way the characters are con-

catenated to form words. Then, the HMMs here will be only in charge of describing the optical shape or morphology of each character.

2.1.2 OFF-LINE HTR SYSTEM

Off-line HTR involves the automatic conversion of handwritten text contained in an image into letter codes usable by a computer. The following describe the modules that are part of the off-line HTR system.

2.1.2.1 Preprocessing

A required step prior to recognize off-line text is to perform some common operations aimed at attenuate or remove degradation problems that are quite common in scanned documents. Notable among these are: smear, large background color variations, geometric distortions, and bleed-through. There is not a perfect set of operations to fix these problems, given that many of them must be applied depending of the document preservation condition. Here, we use the following common operations: thresholding and noise reduction, slant and slope correction and size normalization [17].

Thresholding and Noise Reduction The task of thresholding is to extract the text in the foreground from the background. This is achieved by applying a two-dimensional median filter on the entire image and subtracting the result from the original [18]. Then, a gray-level equalization is applied to increase the image contrast.

Slope Correction The slope is a distortion understood as the angle between the direction of each line with respect to the scanned document horizontal direction. The slope is corrected by calculating the horizontal histogram profile and applying the angle rotation that maximizes the variance of the horizontal projection profile [17].

Slant Correction The slant is the predominant angle in the upward and downward strokes in handwriting. Slant correcting calculates the slant angle by using a method based on vertical histogram profile. This technique seeks for the angle that maximizes the variability between peaks and valleys on the histogram, as occurs in unslanted text [11].

Size Normalization Size normalization aims at minimizing the parts of text with little information. Three different parts can be found in a text line: ascenders, descenders and body. The higher height of ascenders and descenders makes that uninformative areas, which are mainly background, are included in the text frame to consider. The height of the body of a text line can be obtained using a vertical histogram profile [19]. After that, ascenders and descenders are linearly warped in height so that they take up less area (30% and 15% of the total height of the image, respectively), leaving the rest for the body [20].

2.1.2.2 Feature Extraction

Each preprocessed text line image is represented as a sequence of feature vector. First, the image is divided into a grid of N×M cells. Then for each cell three values are calculated: normalized gray level, horizontal gray-level derivative and vertical gray-level derivative. These values are weighted to enhance the role of the central cells using a 5×5 window centered at the current cell to obtain smoothed values. A two-dimensional Gaussian function was used for the normalized gray level and a one-dimensional Gaussian function for the horizontal and vertical gray-level derivatives. The derivatives are computed by least squares, fitting a linear function. Finally, the three images that were obtained after computing these values for every cell are vertically joined, resulting in a (3×N)-dimensional vector for every column of the grid (M) [17]. Usually, N is set to 20 and M is set in such a way that the grid keeps the original image aspect ratio.

2.1.2.3 Recognition

The off-line recognition module follows the same scheme to the one proposed in Section 2.1.1.3 for on-line HTR.

2.2 Gesture Recognition

Gesture recognition have existed in the industry for decades. Early examples of commercial products that successfully incorporated gestures are, e.g., PDAs like the Palm Pilot or the Apple Newton, and

the Windows Tablet. These devices featured the Graffiti [21] and Unistroke shorthand writing systems, which used a single stroke Roman letter-like gesture vocabulary. Gesture recognition has its own roots in HTR. Similar approaches have been used to solve this problem [5]: linear discriminant analysis [22], template matching [23], decision trees [24], neural networks [25], HMMs [26], parsing grammars [27], support vector machines [28], principal component analysis [29], or *ad-hoc* recognizers [30].

2.2.1 RECOGNIZERS

The gesture recognizers used in this thesis are based on the template matching (or instance-based) approach. That is, a query gesture is geometrically compared against a number of stored templates, using 1 nearest-neighbor for classification and either Euclidean distance or a Mean Square Error (MSE) score as dissimilarity measures. Template matchers are a very viable and a relatively simple solution for recognizing gestures, and can be adapted to personalized user gestures. Popular examples of these instance-based recognizers are the so-called *$ family*[1]: $1 [31], $N [32], and their newer versions Protractor [33] and $N-Protractor [34], respectively. Below, we present the different gesture recognizers used in this thesis.

2.2.1.1 $1 Recognizer

The $1 Unistroke Recognizer is a two-dimensional single-stroke recognizer designed for rapid prototyping of gesture-based user interfaces. For each gesture, either test or training sample, $1 preprocesses it into a fixed-dimension vector. It employs a nearest-neighbor approach for classification, given an unknown gesture, it searches for similar gesture templates by calculating an optimal angular distance between the unknown gesture and each of the stored templates. Here, an optional speed-enhancement called Protractor has been used. This recognizer is made up of different steps that are described below.

[1]The name of this family of recognizers comes from their low cost and ease of use, according to the authors: *"[...] we present a $1 recognizer that is easy, cheap, and usable almost anywhere in about 100 lines of code."*

Preprocessing This step is intended to remove irrelevant factors, such as different drawing speeds, different gesture locations on the screen, and noise in gesture orientation. The preprocessing transforms the two-dimensional trajectory of a gesture into a uniform vector representation. Every gesture is resampled into a fixed number N of equidistantly-spaced points [31]. Then points are translated so that the centroid of these points becomes the origin. These steps remove the variations in drawing speeds and locations on the screen. After that, every gesture is rotated so that its indicative angle is aligned with respect the closest direction of the eight major orientations. The indicative angle is defined as the direction from the centroid to the first point of the gesture [31]. After this preprocessing, an equal-length vector is available for each gesture:

$$\mathbf{g} = [g_1, g_2, \ldots, g_{N-1}, g_N] \tag{2.11}$$

where $g_i = (x_i, y_i)$ and $N = 32$.

Recognition $1 finds the most similar template gesture **e** to a given gesture **g**, that minimizes a distance between them, defined as:

$$d(\mathbf{e}, \mathbf{g}) = \min_{-\frac{\pi}{4} \leq \theta \leq \frac{\pi}{4}} \frac{1}{N} \sum_{i=1}^{N} \|g_i - R(e_i, \theta)\| \tag{2.12}$$

where $R(e_i, \theta)$ is a rotation applied to the point e_i around the centroid of the gesture, being θ the angle rotation.

2.2.1.2 $N Recognizer

$N is the extension of $1 to multistroke gestures, where all possible stroke orders and directions are considered for each stored template.

2.2.1.3 $P Recognizer

The $1 and $N recognizers present certain limitations. For example, $1 only handles unistroke gestures and although $N focuses on adding support for multistrokes, the solution involves a really high memory and temporal cost. Most limitations of the $N recognizer come from the chronological order of drawn points, which enforces a predefined order for strokes and points within each stroke. Thus if the

temporal information is dropped, aspects such as the stroke number, ordering or direction become irrelevant. Discarding the time information makes a gesture an unordered set $\mathbf{g} = \{g_1, g_2, \ldots, g_{N-1}, g_N\}$, where point g_i does not necessarily follow g_{i-1} and point g_1 does not mark the starting point of the gesture.

$P finds the most similar template $\mathbf{e}$, given a gesture $\mathbf{g}$, by calculating the minimum distance between two clouds of points.

2.2.1.4 Dynamic Time Warping

Finally, in addition to the $ family recognizers, we will use a nearest-neighbor classifier with Dynamic Time Warping (DTW) as a distance measure. Unlike $ family recognizers, DTW is an algorithm for measuring similarity between two temporal sequences which may vary in length.

Given a template gesture $\mathbf{e} = [(x_i, y_i, t_i) \mid i = 1, 2, \ldots, E]$ and a gesture $\mathbf{g} = [(x_j, y_j, t_j) \mid j = 1, 2, \ldots, G]$, DTW finds an *alignment* between the points of $\mathbf{e}$ and $\mathbf{g}$, having a minimal overall distance. A dynamic programming approach can be used to find this optimal aligment. Finally, a nearest neighbor classifier assigns the label of the most similar template gesture $\mathbf{e}$ to the given gesture $\mathbf{g}$:

$$\hat{\mathbf{e}} = \arg\min_{e} \mathrm{DTW}(\mathbf{e}, \mathbf{g}) \tag{2.13}$$

2.3 Assessing the Recognition Performance

In the last few decades, the assessment paradigm based on labeled training and testing datasets has been adopted in the pattern recognition field. In this paradigm, different approaches can be easily, objectively and automatically tested and compared, without requiring human intervention in the process. Following this paradigm, different objective measures have been adopted in this thesis.

Two measures are typically used to assess the quality of the transcription process: the word error rate (WER) and the character error rate (CER). The WER counts the number of differences between the

transcript proposed by the system and the reference transcript:

$$\text{WER} = 100 \cdot \frac{N_i + N_s + N_d}{N_s + N_d + N_c} \tag{2.14}$$

where N_i is the number of insertions; N_s is the number of substitutions; N_d is the number of deletions and N_c is the number of correct words. However, the computation of this measure is not trivial, given that the hypothesis length can be different from the reference length. Therefore, this metric is calculated by first aligning the recognized word sequence with the reference word sequence using dynamic string alignment, where 4 different situations can occur:

correct word: the reference word is the same as the aligned recognized word.

substitution: the reference word and the recognized word are different.

insertion: a recognized word can not be aligned with any from the reference.

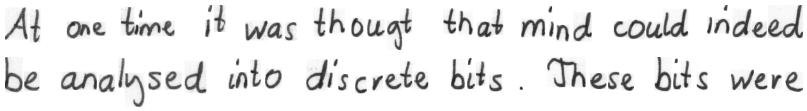
deletion: a reference word does not appear in the recognized.

The optimal alignment is defined as the alignment that minimizes the Levenshtein distance [35]; i.e., the minimum number of insertions, deletions and substitutions between both word sequences. This value can be obtained using dynamic programing. On the other hand, the calculation of the CER is analogous to the WER, but substituting words by characters.

With respect to gesture recognition, we use the error rate (ER). This measure is defined as the number of incorrectly recognized gestures over the total of recognized gestures, expressed as a percentage.

2.3.1 DATASETS

As we mentioned before, in order to assess and compare the performance of the different approaches and methodologies that will be presented in this thesis, public datasets are required. In this section,



(a) IAM Handwriting Database

(b) UNIPEN and UNIPEN-ICROW-2003 datasets

(c) IBM-UB Dataset

(d) $-GDS Dataset

(e) MMG Dataset

(f) Chars74k Dataset

Figure 2.1. Examples from the different datasets.

first handwritten text datasets—off-line and on-line—are briefly described. After that, gesture datasets are reviewed. Figure 2.1 shows some examples from the different datasets. These datasets were built using a variety of different input equipments.

2.3.1.1 IAM Handwriting Database

The IAM Handwriting database (IAMDB) [36] contains off-line forms of handwritten English text. It is composed of 1,539 scanned text pages, handwritten by 657 different writers. The database is provided at different segmentation levels, here sentence-segmented images has been used.

2.3.1.2 UNIPEN database

The UNIPEN database [37][2] is a publicly available on-line dataset. It comes organized into several categories: lower and uppercase letters, digits, symbols, isolated words and full sentences.

2.3.1.3 UNIPEN-ICROW-2003 Benchmark Set

The UNIPEN-ICROW-2003 benchmark set[3] was released during the ICDAR 2003 Informal Competition for the Recognition of On-line Words. The benchmark set is composed of 13,119 isolated on-line free-style words written by 72 different persons. This database contains 884 unique lexical word entries in Dutch, English and Italian.

2.3.1.4 IBM-UB Dataset

The IBM-UB Data Set [38] is a bi-modal—on-line and off-line— and multilingual dataset of handwritten documents. This dataset was collected on a CrossPad device. The CrossPad is a portable digital notepad that used an electronic pen that produced real ink on paper while simultaneously capturing the on-line pen trajectories. Thus, the handwriting sample was available both as a hardcopy paper document as well as on-line trajectory data. It contains pages of forms, spontaneously written letters, tables of words, isolated characters, symbols, etc.

The current release of data comprises two sections. IBM_UB_1 contains free form cursive handwritten pages in English. It contains 6,654 pages of on-line data collected from 43 writers (4,138 summary

[2] http://www.ai.rug.nl/~lambert/overslag/unipen-CDROM-train_r01_v07.tgz

[3] http://www.ai.rug.nl/~lambert/unipen/icdar-03-competition/

pages and 2,516 query pages). It also contains 5,934 page of off-line data collected from 41 writers (3,714 summary pages and 2,220 query pages). IBM_UB_2 contains handwritten pages in French collected from 200 authors. The pages are in the form of booklets each of which has several typed lines that the author reproduces by hand. These lines contain short cursive sentences, or discrete characters, symbols and/or digits.

2.3.1.5 $1-Gesture Dataset

The $1-Gesture dataset ($1-GDS) dataset[4] is a reference dataset in HCI to test unistroke-based gesture recognizers. It contains 16 unistroke gesture classes, 5,280 samples in total. 10 users (plus 1 pilot user) provided 10 samples per class at three articulation speeds (slow, medium, fast) using an iPAQ Pocket PC (stylus as input device). For slow speed, users were asked to *be as accurate as possible*; for medium speed, users were asked to *balance speed and accuracy*; for fast speed, users were asked to *go as fast as you can* [31].

2.3.1.6 Mixed Multistroke Gestures Dataset

The Mixed Multistroke Gestures (MMG) dataset[5] is a reference dataset in HCI to test multistroke-based gesture recognizers. It comprises 16 multistroke gesture classes, 9,600 samples in total. 20 users provided 10 samples per class at three articulation speeds (slow, medium, fast) using either finger (half of the users) or stylus as input devices on Tablet PCs. Speed definitions are the same as in the $1-GDS dataset [34].

2.3.1.7 Chars74k Dataset

The Chars7k4 dataset[6] comprises 62 handwritten classes (0-9, A-Z, a-z), unistrokes and multistrokes, 3,410 samples in total. 55 users provided one sample per class using a Tablet PC at a constant sampling rate of 100 Hz.

[4] `https://depts.washington.edu/aimgroup/proj/dollar/xml.zip`

[5] `https://depts.washington.edu/aimgroup/proj/dollar/mmg.zip`

[6] `http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/EnglishHnd.tgz`

2.4 Conclusion

In this chapter the recognition systems that are going to be used in this thesis have been presented. On the one hand, both off-line and on-line HTR systems are based on HMMs and n-grams. On the other hand, gesture recognition is performed used the $ family of template recognizers. Finally, the datasets that are going to be used to evaluate the techniques presented have been briefly overviewed.

3 *Improving Interactive Transcription of Text Images*

In today's digital age, almost every historical document is still in handwritten *analog* form. In fact, despite the intensive use of computers nowadays, there are still more handwritten documents than printed [39]. There is a growing number of libraries whose mission is to scan and publish those handwritten documents in digital format. But most of these scanned documents remain waiting to be transcribed into an electronic format. The advantages of having an electronic transcript are countless, for example: new ways of indexing, consulting or querying these documents.

The small number of transcribed documents is mainly due to the laborious process involved. Given that these documents are characterized by different calligraphy or print styles from diverse places and time periods, the transcription process must be carried out by a *palaeographers*, who are people specialized in reading and transcribing ancient scripts. How long it takes an expert to transcribe one of these documents depends on many different factors: skill and expertise, size and state of conservation of the document, etc. To give an idea of how complex this task is, an expert could take hours to transcribe just one *difficult* page.

Currently, off-line HTR systems yield good results in applications involving form-constrained handwriting, such as postal addresses or bank check legal amounts [40, 41]. But results are far from being usable in unconstrained handwritten documents—such as historical manuscripts. Thus heavy human intervention is required in order to find and correct the mistakes made by the system. Sometimes, given the high error rates involved, such a *post-editing* solution is quite in-

efficient for the human, being easier for him to discard the system output and start from scratch.

In the last years, having assumed that achieving automatically the perfect transcript is barely impossible, different approaches have been created in which the user and the system collaborate to perform the transcript. The objective is to achieve a synergy, where the accuracy proportioned by the human transcriber is combined with the efficiency of the automatic system to achieve the final correct transcript.

For example, Toselli et al. [42–44] introduced an approach called *Computer Assisted Transcription of Text Images* (CATTI), where the human is directly involved in the transcription process as he is responsible of correcting or validating the outputs. This interactive system is based on previous works on interactive automatic translation [45,46] and interactive speech recognition [6]. The key idea is "*to shift from full automatism towards systems where the output is affected by human feedback*" [7,45]. To start, the system proposes its best hypothesis. If the user finds that it is correct, the answer is accepted and the process goes on with successive input data. Otherwise, the user introduces some amendments that the system must take into account in order to improve the new proposed output. Previous studies have proven that interactive approaches can save significant amounts of overall human effort [47].

Given that the interactive paradigm implies that the user is continuously interacting with the system, it is necessary to make this process as comfortable as possible. Therefore it makes sense to use these applications within the paradigm of pen computing. Following this idea, Toselli et al. [48,49] presented a CATTI system focused on using a pen as a way to interact with the system. However, as we previously commented, the use of a pen comes at the cost of the introduction of an on-line HTR system in charge of decoding its *nondeterministic* pen-strokes.

The results of these previous works suggest that pen interactions, even though the loss of the deterministic accuracy of traditional peripherals, can save significant amounts of user effort with respect to fully manual transcription, as well as to non-interactive post-editing correction. However, despite the encouraging results, these works

fail to provide a sufficiently comfortable approach, since interaction was limited to words. In this thesis we continue the work initiated by Toselli et al., and present 2 new pen-stroke interaction approaches. One focused on interactions at character level and another where interactions are provided without constraining on how to interact with the system, that is, the user can correct any subset of incorrect characters.

This chapter is structured as follows. Section 3.1 overviews CATTI using key-stroke interactions (Section 3.1.2) and pen-stroke interactions at word level (Section 3.1.3). Then subsequent sections present the contributions of this thesis to the topic: Section 3.2 presents an interactive system based on character-level pen-strokes, whereas Section 3.3 presents an approach where any subset of incorrect characters can be corrected. Each section contains different experiments to validate the performance of these new approaches. Finally, conclusions are drawn in Section 3.5.

3.1 Overview of Interactive Transcription of Text Images

This section overviews the Computer Assisted Transcription of Text Images (CATTI) system, introduced by Toselli et al. [42]. In CATTI, the transcription process begins with the system providing a full transcript (s) of a feature vector sequence ($\mathbf{x}$), extracted from a scanned image. The user validates an initial part (p'), which is error-free, and introduces a correct word (w), thereby producing a *validated* prefix ($p = p'w$). Then the system must take into account this new information to suggest a suitable *suffix* (s). This process, shown in Figure 3.1, is repeated until the user accepts the transcript as correct [48].

3.1.1 Formal Framework

More formally speaking, the system must search for the most likely suffix ($\hat{s}$), taking into account $\mathbf{x}$ and p:

$$\hat{s} = \arg\max_{s} P(s \mid x, p) = \arg\max_{s} P(x \mid p, s) \cdot P(s \mid p) \qquad (3.1)$$

Given that the concatenation of p and s constitutes a full transcript hypothesis, Equation (3.1) is equivalent to Equation (2.8). Thus $P(x \mid p, s)$

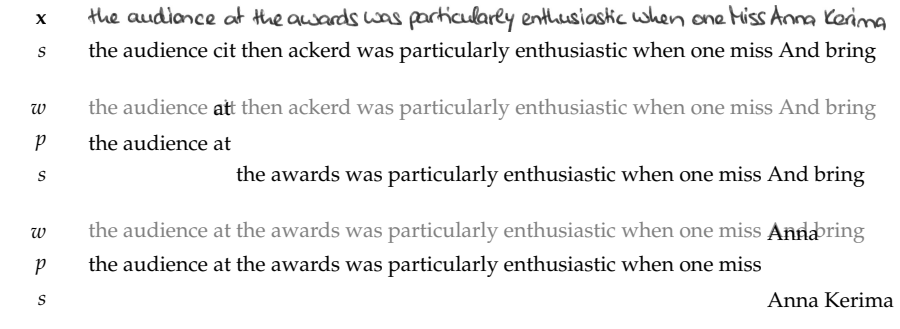


Figure 3.1. Example of interactive transcription. The sentence to recognize is *"the audience at the awards was particularly enthusiastic when one miss Anna Kerima"*. First, the user validates an initial part (p'), which is error-free, introducing a correction (w), producing a *validated* prefix ($p = p'w$). Then, taking into account the validated prefix (p), the system proposes the most probable suffix (s). The process ends when the user accepts the suffix as a full correct transcript.

can be approximated using concatenated character HMMs as shown in Section 2.1.1.3. On the other hand, $P(s \mid p)$ is approximated using an n-gram dynamically modified in order to cope with the increasingly long consolidated prefixes [48]:

$$P(s \mid p) \simeq \prod_{j=1}^{n-1} P(s_j \mid p_{k-n+1+j}^{k}, s_1^{j-1}) \cdot \prod_{j=n}^{m} P(s_j \mid s_{j-n+1}^{j-1}) \tag{3.2}$$

where $k + m$ is the length of the sentence, $p = p_1^k$ is the validated prefix and $s = s_1^m$ a possible suffix.

We can explicitly rely on Equations (3.1) and (3.2) to implement a one-step decoding process as in conventional handwritten text recognition systems. Here the decoder should be forced to cope with p and then continue searching for $\hat{s}$ according to the constraints from Equation (3.2). This can be achieved by building a language model which can be seen as the *concatenation* of a linear language model which strictly accounts for the successive words in p and the *suffix language model* of Equation (3.2).

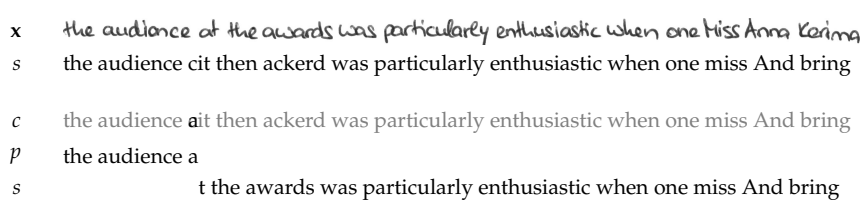


Figure 3.2. Example of a step of the CATTI at character level. The sentence to transcribe is *"the audience at the awards was particularly enthusiastic when one miss Anna Kerima"*. First, the user validates *"the audience a"* (p'), which is error-free, introducing an `'a'` (c), producing a *validated* prefix ($p = p'c$), where the fragment of the prefix formed by complete words (p'') is *"the audience"* and the last incomplete word of the prefix (u) is `'a'`. Then, taking into account the validated prefix (p), the system proposes the most probable suffix *"t the awards was particularly enthusiastic when one miss And bring"* (s), where the system output (v) is `'t'` and the rest of the suffix (s'') is *"the awards was particularly enthusiastic when one miss And bring"*.

3.1.2 INTERACTION USING ISOLATED TYPED CHARACTERS

Following the previous work, Romero et al. [43] presented a version of the CATTI approach using interactions at character level. Here, as soon as the user types a new key-stroke (c), the system proposes a suitable continuation following the same process described in Section 3.1.

As the user operates now at the character level, the last word of the prefix may be incomplete. In order to *autocomplete* this last word, it is assumed that p is divided into two parts: the fragment of the prefix formed by complete words (p'') and the last incomplete word of the prefix (u) (see Figure 3.2). In this case the system has to take into account $\mathbf{x}$, p'' and u, in order to find $\hat{s}$, whose first part is the continuation of u:

$$\hat{s} = \arg\max_{s} P(x \mid p'', u, s) \cdot P(s \mid p'', u) \tag{3.3}$$

Again, Equation (3.3) is equivalent to Equation (2.8), given that the concatenation of p'', u and s constitutes a full transcript hypothesis. Thus $P(x \mid p'', u, s)$ can be modeled with HMMs as previously shown. On the other hand, to model $P(s \mid p'', u)$, it is assumed that the suffix s is divided into two fragments: the first part of the suffix that corresponds with the final part of the incomplete word of the prefix (v)

and the rest of the suffix (s'') (see Figure 3.2). Then the second term of Equation (3.3) can also be written as $P(v, s'' \mid p'', u)$ and can be decomposed into:

$$P(v, s'' \mid p'', u) = P(s'' \mid p'', u, v) \cdot P(v \mid p'', u) \tag{3.4}$$

$P(s'' \mid p'', u, v)$ accounts for the probability of all the whole-words in the suffix and must ensure that $uv = w$ is an existing word in the task vocabulary (V). This probability can be modeled using Equation (3.2). $P(v \mid p'', u)$ should ensure that the first part of the suffix v—usually a word ending part—will be a possible suffix of the incomplete word u, and can be stated as:

$$P(v \mid p'', u) = \begin{cases} \frac{P(u,v|p'')}{\sum_{v':uv' \in V} P(u,v'|p'')} & \text{if } uv \in V \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

3.1.3 Interaction Using Isolated Handwritten Words

Toselli et al. presented a version of the CATTI system which focused on the use of handwritten words as feedback [48,49]. The underlying idea of this work is that the use of more ergonomic interfaces should result in an easier and more comfortable human-machine interaction.

However, it is worth noting that the use of pen-stroke feedback comes at the cost of the introduction of an on-line HTR system in charge of decoding the nondeterministic feedback. This on-line recognition module can be adapted to boost its performance by taking advantage of interaction-derived information (see Figure 3.3). Thus a synergy arises between the main off-line HTR system and the feedback on-line system, where each system *helps* the other to optimize the overall performance.

Continuing with the previous notation, $\mathbf{x}$ is the feature vector representing the input image and t are the on-line pen-strokes provided by the user intended to correct a word in the previous suffix (s'). The position of these corrections with respect to the phrase allows us to define the user-validated-error-free prefix (p'). Then the system has to find a new suffix ($\hat{s}$), as a valid continuation of the prefix (p'), considering all possible decodings (d) of the on-line data (t) and some

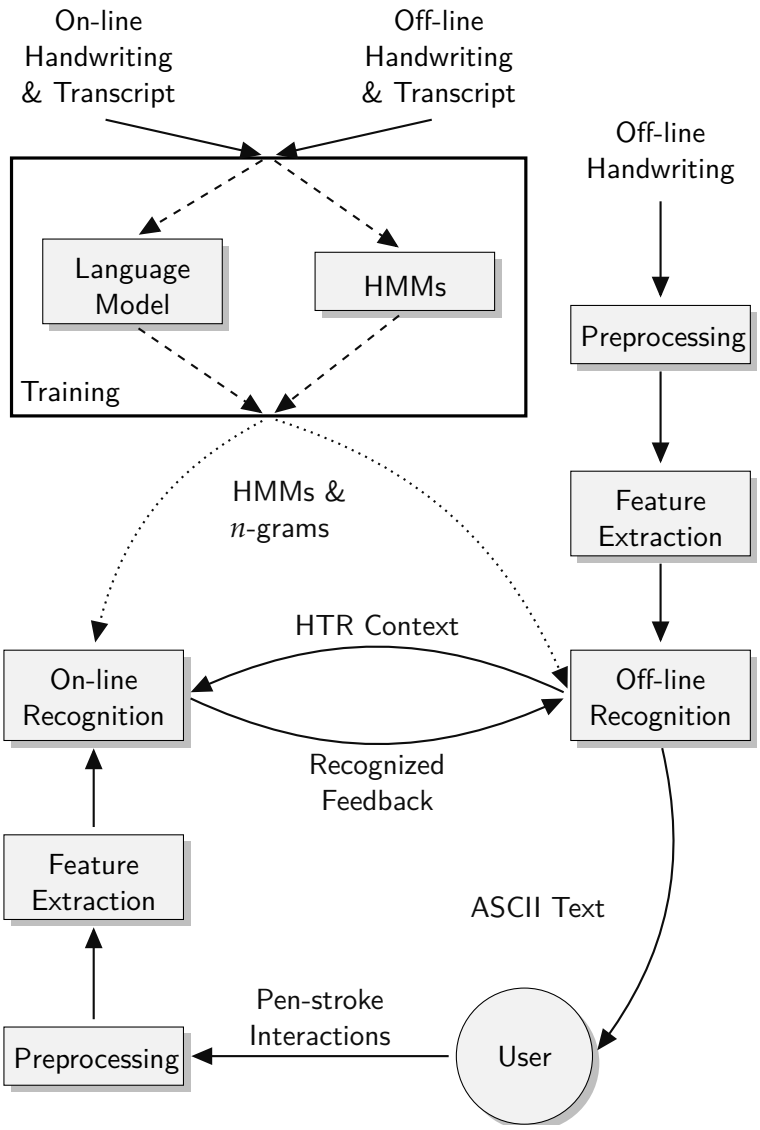


Figure 3.3. Overview of the Computer Assisted Transcription of Text Images (CATTI) system. Once the system has proposed a transcript, the user can provide certain corrections in the form of pen-strokes to fix the first mistake of the system output. This on-line corrections must be decoded first and the accuracy of this decoding can be boosted by using contextual information.

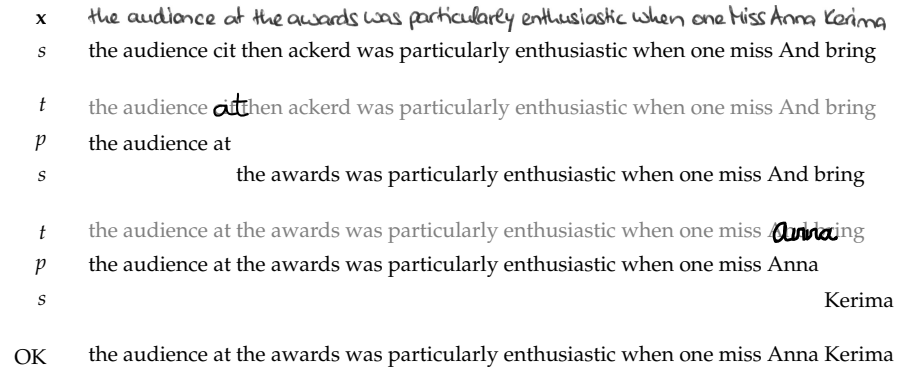


Figure 3.4. Example of CATTI using pen-strokes as feedback to transcribe an image containing the sentence *"the audience at the awards was particularly enthusiastic when one miss Anna Kerima"*. Each interaction consists of two steps. In the first step, the user writes some pen-strokes (t) to amend the suffix (s) proposed in the previous step. This defines a correct prefix p', which can be used by the on-line HTR system to obtain a more accurate decoding of t. In the second step, a new prefix (p) is built from the previous correct prefix p' concatenated with the decoded pen-strokes $\hat{d}$. Using this information, the system proposes the most probable suffix. The process ends when the user accepts the suffix as a full correct transcript.

information from s'—the erroneous word that the user wants to correct [48] (see Figure 3.4):

$$\begin{aligned}\hat{s} &= \arg\max_{s} P(s \mid x, s', p', t) \\ &= \arg\max_{s} \sum_{d} P(s, d \mid x, p', s', t) \\ &\approx \arg\max_{s} \max_{d} P(t \mid d) \cdot P(d \mid p', s') \cdot P(x \mid s, p', d) \cdot P(s \mid p', d)\end{aligned} \tag{3.6}$$

Equation (3.6) can be approximately solved using a two-step approach [48]. First, the on-line system must decode the pen-strokes (t) into the most probable word ($\hat{d}$), knowing that this decoding must be a valid continuation of the prefix (p'):

$$\hat{d} = \arg\max_{d} P(t \mid d) \cdot P(d \mid p', s') \tag{3.7}$$

Once $\hat{d}$ has been recognized, a new consolidated prefix p is produced, joining the previous prefix (p') and the most probable decoding ($\hat{d}$).

Then, the second step searches, in a similarly to Equation (3.1), for the most probable suffix using the new consolidated prefix. These two steps are repeated until p is accepted by the user as a full correct transcript of x. As in Equation (2.10), $P(t \mid d)$ can be approximated using HMMs and $P(d \mid p', s')$ can be modeled using a language model dynamically constrained by information derived from the validated prefix and the previous suffix s'.

3.1.4 Assessing the Performance of Interactive Systems

Given that in this interactive framework the user is "embedded" in the loop, the assessment measures proposed in Section 2.3 are not enough. It is clear that WER and/or CER are no longer enough to evaluate the system quality, given that we must also take into account how much human effort is required to achieve the correct transcript. In fact, we no longer focus only on errors, given that user will ensure the required level of accuracy. Apparently, evaluating interactive systems performance would obviously require human intervention, but this is rather expensive, slow and tedious. By carefully specifying precise goals and ground-truth, the corpus-based assessment paradigm is still applicable in this interactive task. For example, the number of interaction steps needed to produce a fully correct transcript can be used to assess the user effort. This will allow us to obtain adequate estimates of the amount of user interaction effort that would be needed to perform the transcription.

Different objective measures based on labeled datasets have been adopted in this thesis to assess CATTI. Typically, the quality of the transcription at character level for a fully-automatic system is measured by means of the CER. However, CER can not be used as a character-level post-editing effort estimate, given that this measure ignores the possibility of using a word processor, which may provide word "auto-completing" capabilities. Therefore we use an alternative definition of CER which better estimates the effort needed to post-edit a transcription with the help of word auto-completing. This mea-

sure is called post-editing key-stroke ratio (PKSR) [48]. After using an automatic HTR system, the longest common prefix between the hypothesis and the reference is obtained and the first mismatching character from the hypothesis is replaced by the corresponding reference character, the system will complete the word with the most probable word in the task vocabulary. Obviously, the rest of the sentence is not changed. After this, we can define the PKSR as the sum of these erroneous characters divided by the total number of reference characters.

The effort needed by a human transcriber to produce correct transcriptions using the CATTI system with interactions at character level is estimated by the *Key-Stroke Ratio* (KSR), which can be also computed using the reference transcription. In this case, replacing the automatic HTR system for a CATTI system and taking into account that after each user correction the system generates a new suffix. This process is iterated until a full match with the reference is achieved. Thus, KSR and PKSR are defined in an analogous manner, making them comparable in a fair way.

On the other hand, the effort needed by a user to produce correct transcriptions using CATTI with interactions at word level is estimated using the *word stroke ratio* (WSR), which can be computed in a similar manner to KSR. After each system hypothesis, the longest common prefix between the hypothesis and the reference is obtained and the first unmatching word from the hypothesis is replaced by the corresponding reference word. This process is iterated until a full match with the reference is achieved. Therefore, the WSR can be defined as the number of word level user interactions that are necessary to achieve the reference transcription of the text image considered, divided by the total number of reference words. This definition makes WER and WSR comparable.

It should be noted that measures such as the KSR and WSR ignore the user cognitive supervision process effort; that is, only corrective interaction steps are considered relevant in order to estimate system performance. In our case, this is an adequate assumption, a perfect transcript is expected and the human transcriber has to review the whole system transcript hypothesis to guarantee its quality. Never-

theless, in general, performance measures should take into account, probably with different weight, the cost of both supervision and corrective steps.

Finally, the relative difference between PKSR and KSR, as well as between WER and WSR, gives a good estimate of the reduction in human effort that can be achieved by using CATTI with respect to using a conventional HTR system followed by human post-editing with auto-completing. This measure will be denoted as effort reduction (EFR).

We would like to remark that testing of final and mature interactive systems should always be based on performance evaluations with human beings. We must not forget that our objective is to increase user comfort, instead of maximizing (or minimizing) a numerical value.

3.2 Interaction Using Isolated Handwritten Characters

In previous works, user interactions were provided typing characters on a keyboard or writing complete words using pen-strokes. Due to the good results of these works, the idea here was to combine both issues. Here we present a new way of interaction, called from now on CATTI_{c}, using pen-stroke interactions at character level.

3.2.1 Formal Framework

This work focuses on the first part of Equation (3.6); i.e., the restricted decoding of the user interactions, which can be isolated in a single equation:

$$\hat{d} = \arg\max_{d} P(t \mid d) \cdot P(d \mid p'', u', s') \tag{3.8}$$

where $P(t \mid d)$ is provided by a morphological model—HMMs are used—of the character d. On the other hand, $P(d \mid p'', u', s')$ can be modeled using a language model dynamically constrained by information derived from the validated prefix—where p'' are the complete words of the prefix and u' is the last incomplete word—and the previous suffix s' (see Figure 3.5).

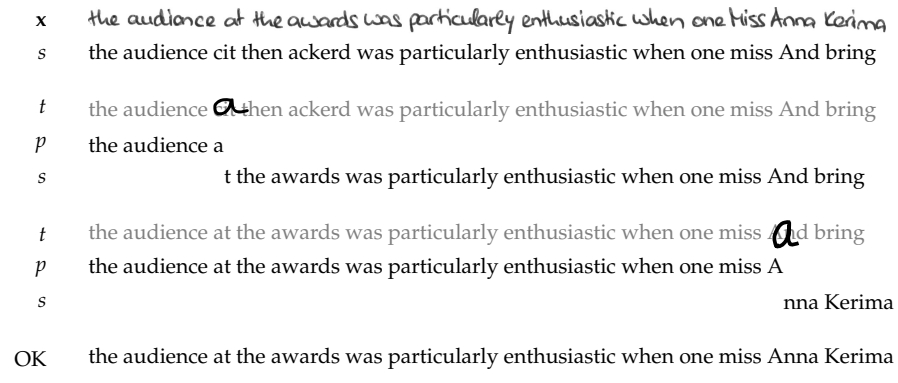


Figure 3.5. Example of interactive transcription using CATTI_c to transcribe an image containing the sentence *"the audience at the awards was particularly enthusiastic when one miss Anna Kerima"*. Each interaction consists of two steps. In the first step, the user writes some pen-strokes (t) to amend the suffix (s) proposed in the previous step. This defines a correct prefix p', which can be used by the on-line HTR system to obtain a more accurate decoding of t. In the second step, a new prefix (p) is built from the previous correct prefix p' concatenated with the decoded pen-strokes $\hat{d}$. Using this information, the system proposes the most probable suffix. The process ends when the user accepts the suffix as a full correct transcript.

Equation (3.8) may lead to several scenarios, depending on the assumptions and constraints adopted for $P(d \mid p'', u', s')$. The first and simplest scenario corresponds to a naive approach where any kind of interaction-derived information is ignored; that is, $P(d \mid p'', u', s') \equiv P(d)$. This scenario will be used to provide a *baseline* result.

In a slightly more restricted scenario, we take into account just the information from the previous off-line HTR prediction s'. The user interacts providing t in order to correct the first wrong character e of s', that follows the validated prefix p'. Clearly, the erroneous character e should be prevented to be decoded again. This *error-conditioned model* can be written as $P(d \mid p'', u', s') \equiv P(d \mid e)$.

Another scenario arises if the portion of word already validated u' is taken into account. That is $P(d \mid p'', u', s') \equiv P(d \mid u', e)$. In this case, the decoding will be clearly more accurate, since we know beforehand that the character to be recognized should be a valid continuation of the part of word accepted so far.

The last scenario arises if the set of complete words p'' are also taken into account. In this case, the possible decodings are constrained to be a suitable continuation of the prefix accepted so far. This scenario can be written as $P(d \mid p'', u', s') \equiv P(d \mid p'', u', e)$.

3.2.2 DYNAMIC LANGUAGE MODELING

Language model restrictions are implemented on the base of n-grams. Given that only one character must be recognized, language models can be easily modified to account for this restriction. That is, after consuming a state of the language model during recognition, all the outputs from that state are redirected to the language model *end symbol* state.

The implementation of the different scenarios is performed as follows. Given that the *baseline* scenario $P(d)$ does not take into account any contextual information, only a character 1-gram is used.

The second scenario $P(d \mid e)$ only considers the first wrong character. The language model probability uses an 1-gram model like the previous one. In this case, it avoids to recognize the wrong character by modifying its probability:

$$P(d \mid e) = \begin{cases} \frac{P(d)}{1-P(e)} & \text{if } d \neq e \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

In the next scenario, given by $P(d \mid u', e)$, the on-line HTR subsystem counts, not only on the first wrong character, but also on the last incomplete word of the validated prefix u'. This scenario can be approached in two different ways: using a character language model or a word language model. The first one uses a modified character n-gram model conditioned by the chunk of word (u') and the erroneous character (e):

$$P(d \mid u', e) = \begin{cases} \frac{P(d|u'^{k}_{k-n+2})}{1-P(e|u'^{k}_{k-n+2})} & \text{if } d \neq e \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where k is the length of the incomplete word u'. In the second one, we use a word language model to generate a more refined character

language model. This can be written as:

$$P(d \mid u', e) = \begin{cases} \frac{P(d|u')}{1-P(e|u')} & \text{if } d \neq e \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

where

$$P(d \mid u') = \frac{P(u', d)}{\sum_{d'} P(u', d')} = \frac{\sum_{v:u'dv \in \mathrm{V}} P(u', d, v)}{\sum_{d'} \sum_{v':u'd'v' \in \mathrm{V}} P(u', d', v)} \tag{3.12}$$

where the concatenation of $u'dv$ is an existing word in the task vocabulary (V).

Finally, the last scenario uses all the available information; i.e., the erroneous character (e), the last incomplete word of the prefix (u') and the rest of the prefix (p''). This can be written as:

$$P(d \mid p'', u', e) = \begin{cases} 0 & \text{if } d = e \\ \frac{P(d|u', p'')}{1-P(e|u', p'')} & \text{if } d \neq e \end{cases} \tag{3.13}$$

Using a word n-gram model we can generate a more refined character language model as in the previous case, so:

$$P(d \mid u', p'') = \frac{P(d, u' \mid p''^{k}_{k-n+2})}{\sum_{d'} P(d', u' \mid p''^{k}_{k-n+2})} = \frac{\sum_{v:u'dv \in \mathrm{V}} P(u', d, v \mid p''^{k}_{k-n+2})}{\sum_{d'} \sum_{v':u'd'v' \in \mathrm{V}} P(u', d', v' \mid p''^{k}_{k-n+2})} \tag{3.14}$$

3.2.3 EVALUATION

In order to validate this approach, we will simulate the process of a user transcribing a text image using CATTI$_{\mathrm{c}}$. This system is composed of two recognition modules: one (off-line) in charge of transcribing the text images and another (on-line) in charge of decoding the user feedback. For each sentence, the CATTI$_{\mathrm{c}}$ proposes a new potential transcript. If the answer contains any mistakes, the user must provide some feedback in the form of an isolated character to correct

the first erroneous character of the transcript. This feedback will be decoded by the on-line HTR system. Once the error is corrected and having a new consolidated prefix, the system generates a new potential suffix. This process runs until the transcription is equal to the reference.

The performance of this approach will be compared against a full post-editing approach and a CATTI system using key-strokes, as the one presented by Romero et al. [43]. The introduction of pen-strokes as feedback leads to a more ergonomic and easier way of working, although at the cost of having to handle the nondeterministic nature of the signal. So the aim of the evaluation is to answer the following questions:

- which is the lowest error we can achieve from the feedback recognition?
- how much the effort is increased by using CATTI_c instead of using a CATTI system with a deterministic device?

To answer the first question we will evaluate the impact of decoding the user feedback making use of the different language models presented in Section 3.2.2, each of which provides different degrees of contextual information. The performance of these scenarios will be assessed using the CER, labeled in the experiments as error rate (ER).

To answer the second question, we will tap into effort assessment measures: the PKSR, KSR and EFR—which are defined in Section 3.1.4. The experimental details are described below.

3.2.3.1 Experimental Details

Both the interaction recognizer and the transcription system used here follow the same architecture as the one presented in Section 2.1. The former is an on-line HTR system and the latter is an off-line HTR system.

Two datasets have been used in the experiments (see Section 2.3.1). We employed IAMDB, an off-line handwritten text dataset, as a document to be transcribed. To better focus on the essential issues of the

considered problems, punctuation marks (excluding the exclamation and interrogation marks), diacritics, or different word capitalizations were removed from the IAMDB dataset transcripts. From the 2,324 sentences that conform IAMDB, 200 were chosen as test sentences, leaving the rest to train the *off-line* HTR system. For each test sentence, the system proposes a new potential transcript. If this hypothesis contains any mistake, some corrections must be introduced by the user. This process is iterated until a full match with the reference is obtained. The IAMDB consists of hand-copied sentences from the much larger electronic text LOB dataset [50], which contains about one million running words. This dataset was used, after removing the test sentences, to train the n-grams.

On-line isolated character samples, extracted from the UNIPEN dataset, are used to simulate user corrections. A random on-line sample of that character is chosen for each test writer from the UNIPEN dataset. Three arbitrary writers were chosen to simulate user interaction and 17 were used as on-line HTR training data.

As a summary, the set of mistranscribed characters is formed by 1,581 lowercase letters, 41 numerical digits, and 5 symbols. Therefore, a total of 4,881 characters—1,627 for each test writer—were chosen as on-line test set from the UNIPEN dataset.

3.2.4 RESULTS

Table 3.1 shows the writer-averaged feedback decoding ER for the 5 different language models that embody the restrictions discussed in Section 3.2.2. The first one (CU) corresponds to a 1-gram estimation of $P(d)$, which is used here as a *baseline*. The second scenario (CU_e) is a error-conditioned character 1-gram estimate of $P(d \mid e)$ defined in Equation (3.9). The third one (CB_e) corresponds to a prefix-and-error-conditioned character 2-gram $P(d \mid u', e)$ defined in Equation (3.10)). The fourth (WU_e) is a prefix-and-error-conditioned word 1-gram (Equation (3.11)). Finally, the last one (WB_e) is a whole-prefix-and-error conditioned word 2-gram $P(d \mid p'', u', e)$ (Equation (3.13)). The best result, an ER of 4.3%, is achieved by the WB_e scenario. This represents a relative accuracy improvement of 38.6% with respect to

Error Rate					Rel. Improv.	
CU	CU_e	CB_e	WU_e	WB_e	WU_e	WB_e
7.0	6.9	6.4	5.0	4.3	28.6	38.6

Table 3.1. On-line feedback recognition system error rates for the different language models. From left to right: character 1-gram (CU), character error-conditioned 1-gram (CU_e), prefix-and-error-conditioned character 2-gram (CB_e), prefix-and-error-conditioned word 2-gram (WU_e), whole-prefix-and-error conditioned word bi-gram (WB_e). The relative accuracy improvements for WU_e and WB_e, with respect to the baseline, are shown in the last two columns. All results are percentages and averaged for the 3 writers.

the baseline scenario. As expected, the more contextual information used, the highest feedback decoding accuracy.

In the case that a character is recognized incorrectly—which occurs more or less 4 times out of 100—the user can try to introduce again that character. We simulated this process by choosing another instance of that character of the same writer from the test set. In order to avoid recognizing the same character, the probability of that character is set to zero in the language model. Results are shown in Figure 3.6. As we can see, after just one more attempt the ER reduction is significant while KSR increment is minimal. Moreover, it is clear that the cognitive cost of making this kind of correction is rather small, since the main difficulty lies in finding the wrong character inside the phrase and not in the correction process itself.

Table 3.2 compares the effort-related results of using CATTI_c with respect to using a CATTI system with deterministic interactions, such as using a keyboard. To make the results comparable, we assume that the cost of pressing a key is similar to performing a key-stroke. According to these results, both—keyboard and pen—interactive systems involves less effort than using a post-editing approach system. The expected user effort for the more ergonomic CATTI_c, is only slight higher than that of using a keyboard. Clearly, this is thanks to the improved on-line feedback decoding accuracy achieved by means of contextual-interaction-derived constraints.

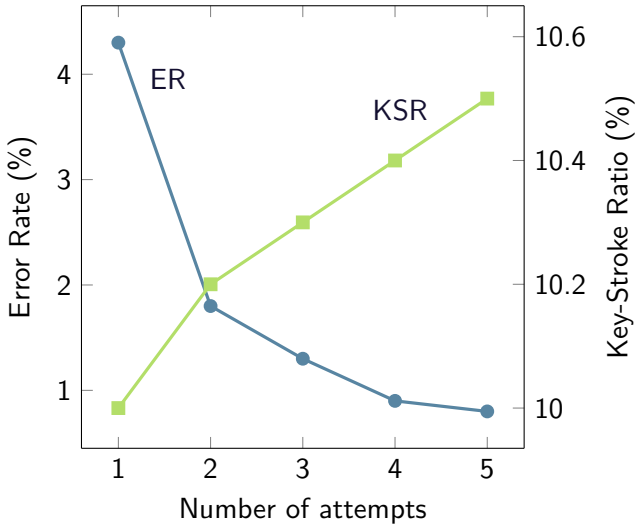


Figure 3.6. Error Rate and Key-Stroke Ratio as a function of the number of times that the user tries to make the correction using pen-strokes.

	KSR		EFR	
PKSR	Keyboard	Pen	Keyboard	Pen
13.5	9.6	10.0	28.9	25.9

Table 3.2. From left to right, estimated effort comparison. PKSR obtained with the *post-editing auto-completing* approach (first column), KSR achieved with a CATTI system using a keyboard (second column) and CATTI_c (third column). Overall EFR of a CATTI system using a keyboard (fourth column) and CATTI_c (fifth column) with respect to a post-editing auto-completing approach. The value of the third column is calculated under the assumption that if the system fails to recognize a character the user proceeds to enter it again with the keyboard, thereby combining two corrections. All results are percentages.

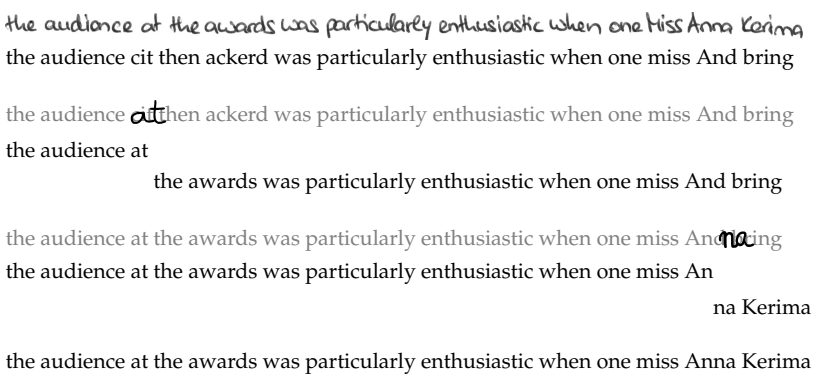


Figure 3.7. Example of interactive transcription using CATTI$_{c*}$ to transcribe an image containing the sentence *"the audience at the awards was particularly enthusiastic when one miss Anna Kerima"*. Each interaction consists of two steps. In the first step, the user writes some pen-strokes (t) to amend the suffix ($\hat{s}$) proposed in the previous step. This defines a correct prefix p', which can be used by the on-line HTR subsystem to obtain a more accurate decoding of t. In the second step, a new prefix (p) is built from the previous correct prefix p' concatenated with the decoded on-line handwritten text $\hat{d}$. Using this information, the system proposes the most probable suffix. The process ends when the user accepts the suffix as a full correct transcription.

3.3 Interaction Using A Sequence of Handwritten Characters

Previous works [43, 48] and CATTI$_c$ results suggest that pen-stroke based interactions are more efficient and more ergonomic than fully-manual transcription, as well as to post-editing correction, even taking into account the loss of the deterministic accuracy of traditional peripherals. However, despite the good results, these previous works fail to provide a sufficiently comfortable approach since they restrict user corrections—whole words or isolated characters. A better interactive system would be the one that allows the user to correct any subset of characters, starting from the first mistranscribed word, leaving the system to complete the remaining part, if needed.

We present here CATTI$_{c*}$, our second approach on interaction using pen-stroke interactions, but in this case using a sequence of characters. As seen in Figure 3.7, the user can produce corrections to fix a set of characters of the first incorrect word in the transcription. These

pen-strokes interactions may represent an isolated character, a word substring or an entire word.

However, this new, hopefully, more ergonomic and friendlier interaction level entails important difficulties and challenges which are addressed in the following sections. In particular, to cope with word substring feedback, different language models have been studied. We also have studied how to take advantage of the contextual information derived from the interaction process to improve the accuracy of the feedback decoding system.

3.3.1 Formal Framework

Again, as in Section 3.2.1 we focus on the first step of Equation (3.6) i.e., the restricted decoding of the user interactions:

$$\hat{d} = \arg\max_{d} P(t \mid d) \cdot P(d \mid p'', u', s') \tag{3.15}$$

Equation (3.15) is equivalent to Equation (3.8), the difference lies in that now t consists of a set of strokes that may comprise from one character to a whole word, going through all the intermediate substring sizes. Again, $P(t \mid d)$ is provided by HMMs and $P(d \mid p'', u', s')$ can be approached by a language model dynamically constrained by information derived from the interaction process. As in Section 3.2.2, different scenarios arise depending on the assumptions and constraints adopted for $P(d \mid p'', u', s')$.

The first and simplest scenario corresponds to a naive approach where any kind of interaction-derived information is ignored; that is, $P(d \mid p'', u', s') \equiv P(d)$. This scenario will be used as a *baseline* result in the later experiments.

A more restrictive scenario arises when we regard the portion of word already validated (u'). In this case the decoding can be more accurate, since we know beforehand that the sequence of pen-strokes to decode must be a valid continuation of the part of word accepted so far. This scenario can be written as $P(d \mid p'', u', s') \equiv P(d \mid u')$.

The last scenario emerges if we add, to the previous one, the set of complete words p''. In this case, the possible decodings are con-

strained to be a suitable continuation of the whole prefix accepted so far. This scenario can be written as $P(d \mid p'', u', s') \equiv P(d \mid p'', u')$.

3.3.2 DYNAMIC LANGUAGE MODELING

Language model restrictions are implemented on the base of n-grams. As we mentioned earlier, the *baseline* scenario, given by $P(d)$, (being $d = \{c_1, c_2, ..., c_l\}$) does not take into account any information derived from the interaction. Here, character n-grams have been used for modeling $P(d)$.

The next scenario, given by $P(d \mid u')$, is approached also using a character n-gram language model, but it is conditioned by the fragment of word (u'). This can be written as [48]:

$$P(d \mid u') = \prod_{i=1}^{n-1} p(c_i \mid c_1^{i-1}, u'^{k}_{k-n+i+1}) \cdot \prod_{i=n}^{l} p(c_i \mid c_{i-n+1}^{i-1}) \tag{3.16}$$

where $u' = \{u'_1, u'_2, \ldots, u'_k\}$. The first term of (3.16) accounts for the probability of the $n-1$ characters of the suffix, whose probability is conditioned by known characters from the validated prefix, and the second one is the usual n-gram probability for the rest of the unknown characters.

The scenario defined by $P(d \mid p'', u')$ uses the last incomplete word of the prefix (u') and the complete words of the prefix (p''). This scenario has been approached using a character n-gram model conditioned by $p' = p''\,␣\,u'$ (where '␣' is a white space). We can restrict this model in a similar manner to (3.16):

$$P(d \mid p') = \prod_{i=1}^{n-1} p(c_i \mid c_1^{i-1}, p'^{z}_{z-n+i+1}) \cdot \prod_{i=n}^{l} p(c_i \mid c_{i-n+1}^{i-1}) \tag{3.17}$$

where $p = \{p_1, p_2, \ldots, p_z\}$.

3.3.3 EVALUATION

In order to validate CATTI$_{c*}$, we will simulate the process of a user transcribing a text image using this system. This system is composed of two recognition modules: one (off-line) in charge of transcribing

the text images and another (on-line) in charge of decoding the user feedback. For each sentence, CATTI$_{c*}$ proposes a new potential transcript. If the answer contains any mistakes, the user must provide some rectifications. In order to increase the reproducibility and simplify the experiments, we assumed that the user rectifications correct from the beginning of the erroneous fragment until the end of the first mistranscribed word; i.e., if the system recognizes the word `january`, but the reference is `janitor`, the sample used as correction is `itor`. Although the system will accept all the possible subsequences: `i`, `it`, `ito` and `itor`. On the other hand, if the first character of a word is incorrect, we simulate that the user writes the whole word. Once the error is corrected and having a new consolidated prefix, the system generates a new potential suffix. This process runs until the transcription is equal to the reference.

CATTI$_{c*}$ will be compared against a full post-editing approach and a CATTI system providing corrections using a keyboard, as the one presented by Toselli et al. [42]. As mentioned earlier, the introduction of pen-strokes as feedback leads to a more ergonomic and easier way of working, although at the cost of having to handle the non-deterministic nature of the signal. However, high error rates causes user rejection, so the aim of the evaluation is to answer the following questions:

- which is the lowest error we can achieve from the feedback recognition?
- how much the effort is increased by using CATTI$_{c*}$ instead of using CATTI with a deterministic device?

To answer the first question we will evaluate the impact of decoding the user feedback making use of the different language models presented in Section 3.3.2, each of which provides different degrees of contextual information. The performance of these scenarios will be assessed using the more pessimistic WER, labeled in the results as error rate (ER), instead of the CER. That is, if just a character of the user feedback is recognized incorrectly, the whole answer is considered as erroneous. A priori, it would seem more logical to make use of the

CER, given that a sequence of characters is expected as output. However, the reason behind of using the WER is that the user is obliged to perform again the whole amendment even if only one character is incorrectly recognized.

To answer the second question, we will tap into effort assessment measures, which are defined in Section 3.1.4. The following section will provide more details about the experimental setup.

3.3.3.1 Experimental Setup

Both the interaction recognizer and the transcription system used here follow the same architecture as the one presented in Section 2.1. The former is an on-line HTR system and the latter is an off-line HTR system.

Two datasets have been used in the experiments (see Section 2.3.1). We employed IAMDB, an off-line handwritten text dataset, as a document to be transcribed. The same partitions and preprocessing as in Section 3.2.3.1 have been used here, that is: 200 sentences as test set, leaving the remaining 2,124 sentences to train the off-line HTR system. After simulating the interactive transcription process of those 200 test sentences, the set of mistranscribed character sequences is formed by 2,262 sequences, being 1,848 of them complete words.

The LOB dataset [50] has been used again, after removing the test sentences, to train the language models based on n-grams. Character n-grams with $n = 9$ have been trained. This value has been chosen taking into account the histogram of average lengths of the LOB dataset words.

The on-line UNIPEN dataset has been employed again to simulate user interactions. The same configuration as in Section 3.2.3.1 has been used, that is 3 test writers and 17 as training data. Unfortunately, UNIPEN dataset does not contain any of the required samples that should be written by the user to interact with the system. Therefore, these words had to be generated by concatenating the different characters of the word. Here a non-connection concatenation technique was used, where random characters from each test writer were put together to form the sequence of characters. Each character was

aligned along a word baseline, except if it had a descender, in this case, the character baseline was raised 1/3 of its height. Moreover, a small noise was introduced for each character in its vertical and horizontal position. More details about the technique can be found in [49]. As as summary, a total of 2,262 test tokens (414 word fragments and 1,848 complete words) were generated—754 tokens for each test writer. Training data was produced in a similar fashion, using the 17 training writers. For each of these writers, the 1,000 most frequent Spanish and English words were synthesized using the previous technique. In addition, a random-selected sample of digits and symbols were chosen. Altogether, 34,714 training tokens were available for training the on-line recognizer—2,042 tokens for each training writer.

3.3.4 RESULTS

Table 3.3 reports the average feedback decoding error considering the different scenarios described in Section 3.3.2. The first one, called here CN, corresponds to the baseline given by $P(d)$. We used a 9-gram character model constructed using the characters of isolated words, since in this scenario there is no need of context between words. The second scenario ($P(d \mid u')$), called here CN_p, uses the same character 9-gram language model as above, but in this case is prefix-conditioned. The third one, called W-CN_p, is a a whole-prefix-conditioned character 9-gram ($P(d \mid p'', u')$). This language model has been constructed using separated characters of words grouped in pairs, thus this language model contains information about how words are connected. Finally, the last column represents our best system, created by combining the best results—CN_p for word fragments and M-CN_p for complete words. This approach can be easily implemented by taking into account the position of the pen when the user is writing. If the pen is at the beginning of a word, M-CN_p would be used to recognize. If the pen is in any other position, CN_p would be use. As expected, the more information available, the highest feedback decoding accuracy. The excellent result achieved by CN_p recognizing word fragments is probably due to the way that language model is created, since it only uses isolated words.

	CN	CN_p	W-CN_p	Comb.
Word Fragments	11.4	3.2	8.5	3.2
Complete Words	12.7	12.7	10.9	10.9
Average	12.5	11.0	10.5	9.5

Table 3.3. On-line feedback recognition system error rates for the different language models. From left to right: character 9-gram (CN), prefix-conditioned character 9-gram (CN_p), whole-prefix-conditioned character 9-gram (W-CN_p) and the combination using the best language model for word fragments and the best scenario for complete words (Comb.). All results are percentages and averaged for the three test writers.

	WSR		EFR	
WER	Keyboard	Pen	Keyboard	Pen
25.1	21.5	23.1	14.3	8.0

Table 3.4. From left to right, estimated effort comparison. A post-editing auto-completing approach (first column), a CATTI system using a keyboard (second column) and $CATTI_{c*}$ (third column). Last two columns show the overall effort reduction comparison of CATTI using a keyboard and $CATTI_{c*}$ with respect to the post-editing auto-completing approach. All results are percentages.

As a final overview, Table 3.4 compares the estimated user effort results. The first column shows the WER achieved using post-editing corrections. The second one, shows the WSR achieved by a CATTI system using a deterministic interface, such as the keyboard. The third column shows the $CATTI_{c*}$ WSR using the best feedback decoding approach (last column in Table 3.3). This value is calculated under the simplification that if the system fails to recognize a sequence of characters, the user proceeds to enter it again with the keyboard, thereby counting two corrections. Finally, the last two columns show the EFR for the CATTI system using a keyboard and $CATTI_{c*}$ with respect to post-edition with auto-completing. According to these results, the expected $CATTI_{c*}$ user effort is only slightly higher than that of using a deterministic interaction system and still better than a post-editing approach.

3.4 Discussion

Experiments presented in this chapter support the benefits of employing interactive transcription systems, such as CATTI$_c$ and CATTI$_{c*}$, using pen-strokes rather than transcription followed by human post-editing. From the results, we observe that the use of the more ergonomic feedback modality comes at the cost of only a reasonably small number of additional interaction steps needed to correct the few feedback decoding errors. The number of these extra steps is kept very small thanks to the system ability to use interaction-derived constraints to considerably improve the on-line HTR feedback decoding accuracy. Clearly, this would not have been possible if a conventional off-the-shelf on-line HTR decoder were trivially used for the correction step.

With respect to feedback decoding error rates, results in Tables 3.1 and 3.3 are not directly comparable as they are. As we mentioned before, the results of CATTI$_c$ are provided using the CER, whereas the CATTI$_{c*}$ results are provided using WER as a measure. This is because we believe that it is unfair to evaluate the second approach using the CER. However, to make a comparison between both approaches, we can roughly estimate the CER of the second system. In this approach, there is a total of 754 user interactions and their mean length is 6 characters. Thus, on average, there are 4,524 characters in this test set. If we assume that every misrecognized token was due to, more or less, 1 or 2 misrecognized characters (for example, the reference is "liver", but the system output was "lover" or "lovers") that makes a total of 144 misrecognized characters, resulting in a CER of 2%. Therefore, according to these CER values, CATTI$_{c*}$ is better than the CATTI$_c$ from a recognition accuracy viewpoint.

Moreover, respecting user effort, the number of interactions using CATTI$_c$ is higher than if CATTI$_{c*}$ is used—1627 and 754 interactions per user, respectively. Even though CATTI$_{c*}$ corrections are longer, we believe that it is always preferable the system that minimizes the number of user interactions. Probably the most expensive process, according to user effort, is to find the errors in transcription. However, our user effort estimation ignores the associated cognitive cost of finding the mistakes in the transcript and only takes into account

the effort of making the correction. This is because we can not quantify this cost by using the corpus-based paradigm. We will retrieve this topic later on in Section 6.3, where experiments with real users are conducted in a similar interactive scenario.

3.5 Conclusion

In this chapter, a computer assisted approach to transcribe text images has been reviewed. This approach combines the efficiency of automatic HTR with the accuracy of the user. The user corrections become part of an increasingly longer validated recognition. This validated contextual information is exploited by the system to suggest better answers. Different types of interaction have been also reviewed: interaction using a keyboard and pen-strokes.

Two different new ways of interaction, called CATTI$_c$ and CATTI$_{c*}$, have been presented. Experiments have been performed using different datasets to prove the validity of these new approaches.

4 *Improving Sigma-Lognormal Parameters Extraction*

Understanding how the human body moves—hand movements, in particular—is important in the development of a handwriting or gesture recognition system. Writing and drawing are among one of the most complicated tasks carried out by humans. They require sensory-motor control mechanisms involving a large spectrum of cerebral activities in order to produce complex movements.

Concretely, handwriting has been studied in many fields of research. Many of these works rely on a stroke generation model that assumes that complex movements are made up of simple strokes, called *rapid-aimed* movements. These rapid-aimed movements share a velocity profile which is approximately bell-shaped [51–56], asymmetric [57,58] and its bell-shaped form is always preserved, existing only changes in the scale [55,59–62].

Different models have been proposed over the years to tentatively explain how the central nervous system generates and controls the kinematics of human movements. For example, equilibrium point models [63]; behavioral models [64]; kinematic models [65,66]; models using the minimum principle [67], where different variables are minimized: movement time [68], acceleration [69], jerk [62], snap [70] and torque changes [56]. Many models exploit the properties of various functions to reproduce human movements: exponentials [71], Gaussians [72], beta functions [73], splines [74], and trigonometrical functions [75].

Among these models, the kinematic theory of rapid human movements[1] [66], and its associated Sigma-Lognormal model [76], provides

[1] Also known as just the Kinematic Theory.

the most solid framework to date for the study of the production of human handwriting. This framework takes into account different psychophysiological features, such as the neuromuscular response time, and has been shown to outperform many other approaches [8].

The Kinematic Theory has been used to, among other things: analyze how children learn to write [77]; generate synthetic human-like samples [76, 78–80]; develop biomedical tools for neurodegenerative disorders diagnosis [81, 82]; and design signature verification systems [83, 84].

Most of these previous works are built upon the Sigma-Lognormal extractor presented by O'Reilly and Plamondon [85]. In order to analyze the strokes that composes a movements, and given that strokes are "hidden" in the movement, an extractor is necessary to automatically uncover them. As a results, the Sigma-Lognormal extractor provides the set of lognormals that better approximates the original trajectory. However, although the performances reported by those works are good, there is still room for improvement in order to reach a more accurate and robust lognormal extraction. Continuing the work initiated by O'Reilly and Plamondon regarding Sigma-Lognormal extraction, we present in this chapter a new Sigma-Lognormal extractor, hopping for a more accurate and robust extraction.

This chapter is organized as follows. Section 4.1 overviews the Kinematic Theory and its associated models, as well as the technique used to extract the Sigma-Lognormal model for a given trajectory. Then Section 4.2 presents a new Sigma-Lognormal extractor. This extractor is compared with the state-of-the-art extractor in Section 4.2.3. Finally, a discussion and some conclusions are provided in Section 4.3 and Section 4.4, respectively.

4.1 Overview of the Kinematic Theory

The Kinematic Theory, presented by Réjean Plamondon, was aimed at explaining the generation and control of rapid human movements. This theory has been proved in the past years to be one of the best approaches to describe the global properties of the neuromuscular networks involved in a synergistic action [86, 87]. It proposes expla-

nations about the emergence of the basic kinematic relationships and psychophysical laws that have been consistently reported in the studies dealing with rapid human movements [86].

The reader should note that the use of the word "rapid" in the kinematic theory name is due to historical reasons, as its first model was aimed at studying truly rapid movements, such as those involved in creating handwritten signatures. However, the theory has been generalized to any type of movements [76].

The basic idea behind the Kinematic Theory is that the neuromuscular network involved in the production of a rapid-aimed movement can be considered as a linear system made up of a large number of coupled subsystems [66,88,89]. For example, when writing on a piece of paper we use primarily, from the shoulder down to the joints of the fingers, each of which must be controlled by the muscle groups attached to them. The resulting velocity profile of a specific neuromuscular system converges toward a lognormal function, that is:

$$\|\vec{v}(t)\| = D\Lambda(t; t_0, \mu, \sigma^2) \tag{4.1}$$

being

$$\Lambda(t; t_0, \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}(t - t_0)} \exp\left[\frac{-(\ln(t - t_0) - \mu)^2}{2\sigma^2}\right] \tag{4.2}$$

where D describes the amplitude of the input command; t_0 is the time occurrence of the input command; μ is the neuromuscular system time delay on a logarithmic time scale and σ is the neuromuscular system response time on a logarithmic time scale.

There are many models derived from this *lognormal* paradigm. Among them, two stand out: the Delta-Lognormal model ($\Delta\Lambda$) and the Sigma-Lognormal model ($\Sigma\Lambda$).

4.1.1 DELTA-LOGNORMAL MODEL

According to the Delta-Lognormal model, the production of a rapid movement requires the activation of two neuromuscular system, one agonist and the other antagonist to the direction of the movement. Both systems are simultaneously activated by two input commands at

a time t_0. These synchronous commands propagate in parallel across the two neuromuscular systems, being both described by a lognormal impulse response. The Kinematic Theory predicts that the velocity profile magnitude of a rectilinear movement is completely described by a Delta-Lognormal equation [66, 90]:

$$\|\vec{v}(t)\| = D_1\Lambda_1(t; t_0, \mu_1, \sigma_1^2) - D_2\Lambda(t; t_{0_2}, \mu_2, \sigma_2^2) \tag{4.3}$$

4.1.2 SIGMA-LOGNORMAL MODEL

The Sigma-Lognormal Model is the latest and more complete representation in the family of models supported by the Kinematic Theory [76]. Unlike the Delta-Lognormal model, the Sigma-Lognormal model does not assume that the involved neuromuscular systems are working in precisely opposite directions. The synergy emerging from the interaction and coupling of many of these neuromuscular systems results in the generation of any complex movements, not limited to single stroke, such as the Delta-Lognormal model.

The generation of these complex movements obeys the *lognormality principle* [77]. This principle states that a user in total control of his movements produces the minimum number of perfect strokes in order to generate the intended trajectory. In contrast, when she is not in full control, the produced strokes will not be ideal lognormals or she will use a large number of these to produce the movement. Therefore, the lognormality of velocity profiles can be interpreted as reflecting the behavior of users who are ideal motion planners.

According to the Sigma-Lognormal model, the velocity of a complex movement is described by the temporal overlap of the velocities ($\vec{v}_i(t)$) of each involved stroke [91]:

$$\vec{v}(t) = \sum_{i=1}^{N} \vec{v}_i(t) = \sum_{i=1}^{N} \begin{bmatrix} \cos\phi_i(t) \\ \sin\phi_i(t) \end{bmatrix} D_i\Lambda(t; t_{0_i}, \mu_i, \sigma_i^2) \tag{4.4}$$

where N represents the number of strokes and $\phi_i(t)$ is the direction profile for each stroke, described by an error function:

$$\phi_i(t) = \theta_{s_i} + \frac{\theta_{e_i} - \theta_{s_i}}{2}\left[1 + \operatorname{erf}\left(\frac{\ln(t - t_{0_i}) - \mu_i}{\sigma_i\sqrt{2}}\right)\right] \tag{4.5}$$

Finally, the $x(t)$ and $y(t)$ Cartesian coordinates can be calculated integrating $\vec{v}(t)$:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \sum_{i=1}^{N} \int_{t_{0_i}}^{t} \vec{v}_i(\tau)\, \mathrm{d}\tau \tag{4.6}$$

or alternatively, $x(t)$ and $y(t)$ can also be computed directly from the Sigma-Lognormal parameters [85]:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \sum_{i=1}^{N} \frac{D_i}{\theta_{e_i} - \theta_{s_i}} \begin{bmatrix} \sin\phi_i(t) & - & \sin\theta_{s_i} \\ -\cos\phi_i(t) & + & \cos\theta_{s_i} \end{bmatrix} \tag{4.7}$$

Six Sigma-Lognormal parameters completely describe a stroke [87, 90, 92, 93]:

$$P_i = \{t_{0_i}, D_i, \mu_i, \sigma_i, \theta_{s_i}, \theta_{e_i}\} \tag{4.8}$$

The parameters reflect both the motor control process and the neuromuscular response. The parameter t_{0_i} indicates the instant when the stroke is sent to the input of the neuromuscular system. The space features are defined by the length D, the starting direction θ_{s_i} and the ending direction θ_{e_i}. The lognormal impulse response of the neuromuscular system is characterized by the logtime delay μ_i and the logresponse time σ_i.

Finally, as illustrated in Figure 4.1, if the strokes are executed sequentially, that is, the end point of the ith stroke is concatenated to the start point of the $i+1$th stroke, the result is called *action plan* [94, 95].

4.1.3 SIGMA-LOGNORMAL PARAMETERS EXTRACTOR

Parameters extraction is an essential step for Sigma-Lognormal representation. Given that strokes are "hidden" in a complex movement, a Sigma-Lognormal parameters extractor is needed to perfom a kind of "reverse engineering" process to uncover the values of the parameters that best explain the observed velocity profile.

Many attempts have been conducted to develop algorithms for that purpose. O'Reilly and Plamondon presented a Sigma-Lognormal parameters extractor, based on the XZERO algorithm [96]. Below we briefly describe the steps performed by this extractor.

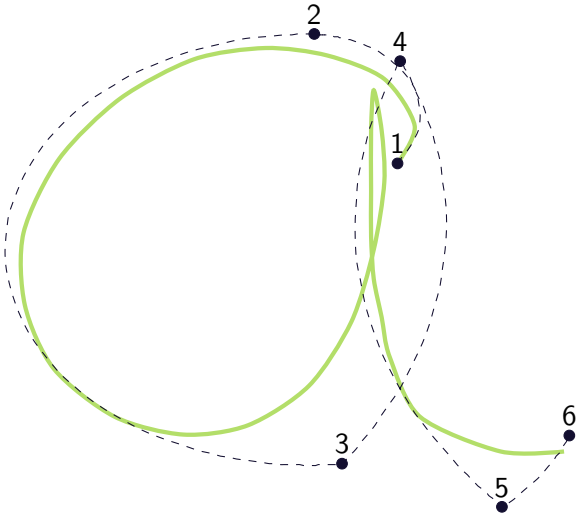


Figure 4.1. A handwritten letter 'a'. The fiducial trajectory (green thick line) is described by the temporal overlap of a series of strokes, called virtual trajectory (black dashed arcs), connecting a sequence of virtual targets (1-6 black circles). Each stroke is described by a lognormal equation.

4.1.3.1 Preprocessing

A preprocessing is applied to enhance the quality of the signal. The signal is interpolated using cubic splines and re-sampled to 200 Hz [85]. Then, the observed velocity[2] magnitude profile is calculated using:

$$\|\vec{\mathrm{v}}(t)\| = \sqrt{\mathrm{x}'(t)^2 + \mathrm{y}'(t)^2} \tag{4.9}$$

where the prime symbol refers to the first derivative with respect to t, which is computed using Equation (2.1). After that, a low-pass filter is applied to the speed to smooth the noise.

4.1.3.2 Stroke Identification

To estimate the Sigma-Lognormal parameters, strokes must first be identified in the velocity magnitude profile. This is achieved by look-

[2]Hereinafter we denote the observed velocity as $\vec{\mathrm{v}}(t)$, the analytic velocity as $\vec{v}(t)$. Analogously, the observed position is denoted as $\vec{\mathrm{x}}(t)$ and $\vec{\mathrm{y}}(t)$ and the computed position as $\vec{x}(t)$ and $\vec{y}(t)$.

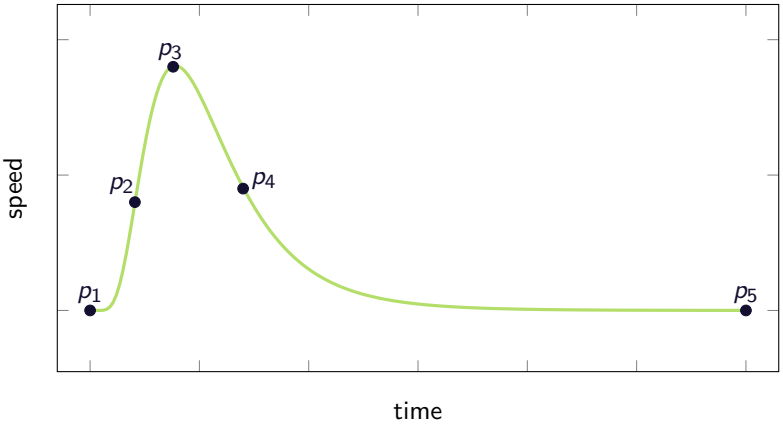


Figure 4.2. An example of a lognormal velocity magnitude profile. The black dots indicate, from left to right: the beginning of the lognormal (p_1), where $\|\vec{v}(t)\| \approx 0$; first inflexion point (p_2); local maximum velocity (p_3); second inflexion point (p_3) and the end of the lognormal (p_5), where $\|\vec{v}(t)\| \approx 0$.

ing for local maxima (p_3; see Figure 4.2) in $\|\vec{v}(t)\|$. Then, for each local maximum, 2 inflection points (p_2 and p_4) are sought in its vicinity. Inflection points are found by looking for changes in the sign of the curvature defined in Equation (2.4), or alternatively, points on $\|\vec{v}(t)\|$ at which the second derivative changes its sign. Finally, two local minima (p_1 and p_5), are estimated. We consider as local minima those values in $\|\vec{v}(t)\|$ with a magnitude of less than 1% of its local maximum. This calculation results in 5 characteristic points for each stroke. Each characteristic point is located at a certain time t and has a velocity magnitude $\|\vec{v}(t)\|$.

It is important to remark that the selection of these characteristic points must be robust, given that superposition can drift their location and noise can generate false positive identifications. First, their location is corrected taking into account the expected variability of μ and σ [85].

Three criteria are applied then to retain only the meaningful characteristic point sets. The first criterion states that the area under the curve delimited by p_1 and p_5 must be greater than the mean minus one standard deviation of the area under the curve of all selected characteristic point series. The second criterion states that the maximum

value of a characteristic point series (that is, $\|\vec{v}(t_3)\|$) is not more than 15 times smaller than the maximum value of $\|\vec{v}(t)\|$. A third condition is applied, it states that the value of t_3 of a series taken to estimate a new lognormal should be greater than those used previously, thus ensuring the progression of the algorithm.

4.1.3.3 Stroke Extraction

After stroke identification, sets of characteristic points are available. The following procedure is applied for each set of characteristic points in the order of their time occurrence:

1. The Sigma-Lognormal model parameters are calculated as shown in Sections 4.1.3.4 and 4.1.3.5.

2. The analytic estimated speed is subtracted from the observed velocity magnitude profile.

The intuition behind this approach is that it is believed to provide a better estimate of each lognormal, given that it minimizes the speed superposition effects from the left neighbor; i.e., $i-1$th stroke, by removing first its extracted value.

However, if the last set of characteristic points is reached without having obtained a *satisfactory* result, the extractor toggles a second mode, where the remaining lognormals are estimated as in the original mode, but in this case in the order of their area under the curve. Figure 4.3 shows an example of the stroke extraction using this technique.

4.1.3.4 Velocity Estimation

The Robust XZERO algorithm [85, 97] is used to estimate the speed-related parameters, i.e.; σ, μ, t_0 and D, for every lognormal. This algorithm exploits time and velocity constraints on 3 of the lognormal

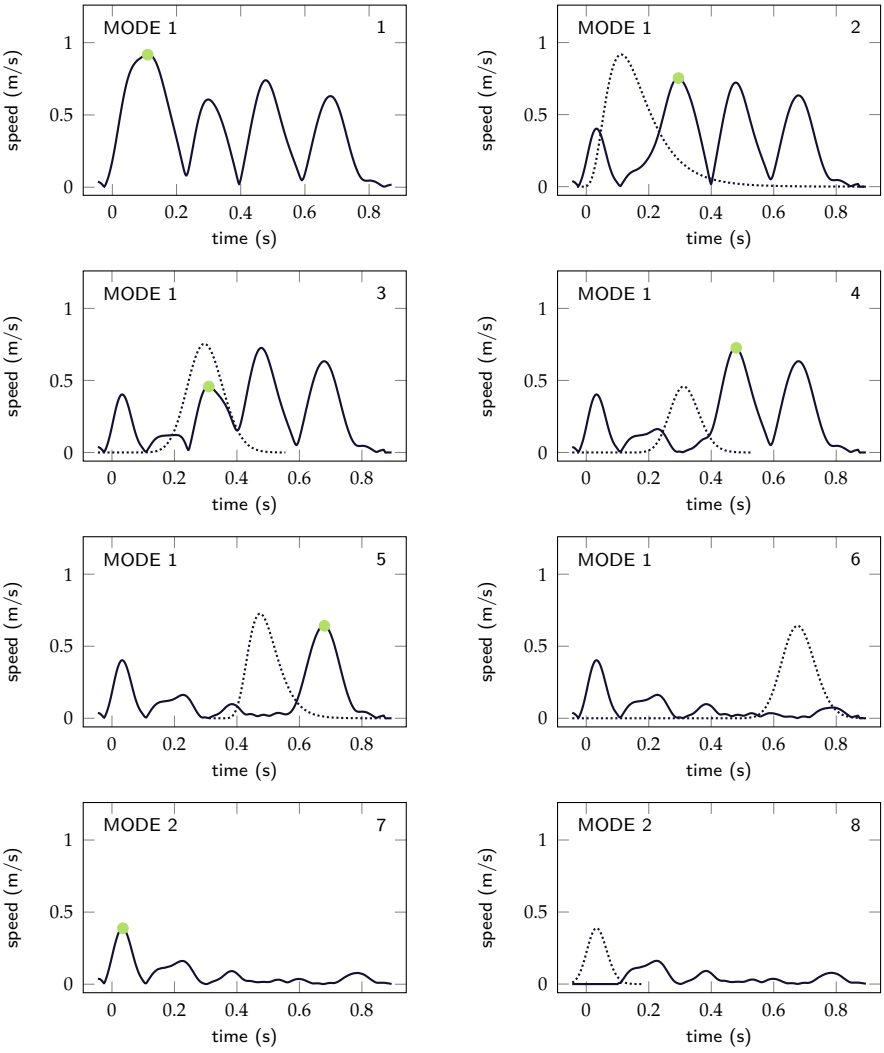


Figure 4.3. Step-by-step example of the Sigma-Lognormal extraction using the 2-mode-based extractor. The reading order is left-right, top-bottom. The solid black line indicates the velocity magnitude profile to be reconstructed, whilst the green dot indicates the following $\|\vec{v}(t_3)\|$ to be extracted. The dotted black line shows the lognormal equation extracted, and subtracted from the velocity magnitude, in the previous step.

characteristic points[3] (p_α):

$$\begin{aligned} t_\alpha &= t'_\alpha \\ \|\vec{\mathrm{v}}(t_\alpha)\| &= \|\vec{v}\,(t'_\alpha)\| \end{aligned} \tag{4.10}$$

where $\alpha \in \{2,3,4\}$. Left-side terms are observed values, obtained from the velocity magnitude profile whilst the right-side terms are calculated analytically.

The estimate of σ, μ, t_0 and D can be carried out using different two-element combinations of the lognormal characteristic points and Equations (4.11) to (4.15) that are derived from Equation (4.10).

$$\sigma^2 = \begin{cases} -2 - 2\ln r_{\alpha\beta} - \dfrac{1}{(2\ln r_{\alpha\beta})}, & \text{if } \alpha = 2,\ \beta = 3 \\ -2 + 2\sqrt{1 + \ln^2 r_{\beta\alpha}}, & \text{if } \alpha = 2,\ \beta = 4 \\ -2 - 2\ln r_{\beta\alpha} - \dfrac{1}{(2\ln r_{\beta\alpha})}, & \text{if } \alpha = 3,\ \beta = 4 \end{cases} \tag{4.11}$$

where $r_{ij} = \|\vec{\mathrm{v}}(t_i)\| / \|\vec{\mathrm{v}}(t_j)\|$.

$$\mu = \ln\left(\frac{t_\alpha - t_\beta}{e^{-a_\alpha} - e^{-a_\beta}}\right) \tag{4.12}$$

$$t_0 = t_\alpha - e^{\mu - a_i} \tag{4.13}$$

$$D = \|\vec{\mathrm{v}}(t_\alpha)\|\,\sigma\sqrt{2\pi}\exp\left(\mu + \frac{a_\alpha^2}{2\sigma^2} - a_\alpha\right) \tag{4.14}$$

where $\alpha, \beta \in \{2,3,4\}, \alpha < \beta$ and

$$a_i = \begin{cases} \frac{3}{2}\sigma^2 + \sigma\sqrt{\frac{\sigma^2}{4} + 1}, & \text{if } i = 2 \\ \sigma^2, & \text{if } i = 3 \\ \frac{3}{2}\sigma^2 - \sigma\sqrt{\frac{\sigma^2}{4} + 1}, & \text{if } i = 4 \end{cases} \tag{4.15}$$

The parameters are computed using all the possible combinations of p_2, p_3 and p_4 in order to provide more robustness to the estimation. Once all the estimates are available, the set of calculated parameters which minimizes the least-square error with respect to the original velocity magnitude profile is kept as solution.

[3] Although constraints are also met for p_1 and p_5, they are not very robust in practice and therefore it is not recommended to take them into account.

4.1.3.5 *Stroke Direction Estimation*

At this point, the direction parameters (θ_s and θ_e) remain still unknown. Nevertheless, since each stroke occurs along a pivot, it can be proved that the angular variation is proportional to the distance traveled along the trajectory [66]. This property can be exploited to perform a linear interpolation that computes the angular parameters θ_s and θ_e using Equations (4.5) and (4.16) to (4.19).

$$\theta_s = \phi(t_3) - \Delta\phi \cdot (\mathrm{d}(t_3) - \mathrm{d}(t_1)) \tag{4.16}$$

$$\theta_e = \phi(t_3) - \Delta\phi \cdot (\mathrm{d}(t_5) - \mathrm{d}(t_3)) \tag{4.17}$$

where

$$\Delta\phi = \frac{\phi(t_4) - \phi(t_2)}{\mathrm{d}(t_4) - \mathrm{d}(t_2)} \tag{4.18}$$

and

$$\mathrm{d}(t_i) = \begin{cases} 0, & \text{if } i = 1 \\ \frac{D}{2}[1 + \mathrm{erf}(-a_i/\sigma\sqrt{2})], & \text{if } i = 2, 3, 4 \\ D, & \text{if } i = 5 \end{cases} \tag{4.19}$$

4.1.3.6 *Optimization*

Once the extraction process is terminated and in order to enhance the result, a global velocity optimization is performed for each extracted stroke. This is done using the method of *least squares* [85], where the Sigma-Lognormal parameters are modified in order to minimize the differences with respect to the original speed.

4.1.4 Assessing the Sigma-Lognormal Parameters Extraction

A good extraction result is expected to have the following properties:

- The velocity reconstruction quality should be higher than the preset threshold.
- For a given velocity reconstruction, the smallest number of strokes is always desirable.

- The shape reconstruction quality should be the highest as possible.

The extraction quality will evaluated using 3 measures. The first measure is the signal to noise ratio (SNR) between the original and the reconstructed velocity profile (SNR_v):

$$\text{SNR}_v = 10 \log \left(\frac{\sum_{t=1}^{T} \|\vec{\text{v}}(t)\|^2}{\sum_{t=1}^{T} \|\vec{\text{v}}(t) - \vec{v}(t)\|^2} \right) \tag{4.20}$$

where $\vec{v}(t)$ is the analytic velocity, $\vec{\text{v}}(t)$ is observed velocity and T is the duration of the handwriting signal.

Previous works suggest that different SNR_v thresholds can be used to quantify what is considered as a good reconstruction [98], although in general values higher than 15-20 dB are considered to be a reference baseline [99].

As mentioned before, the generation of a human movement obeys the lognormality principle [77]. This principle states that a user in perfect control of his movements produces the minimum number of "perfect" lognormal strokes in order to generate the intended trajectory. The second measure evaluates the reconstruction quality according to the previous statement, calculating the ratio between the SNR_v and the number of extracted lognormals (nbLog or N in equations):

$$\text{SNR}_v/\text{nbLog} = 10 \log \left(\frac{\sum_{t=1}^{T} \|\vec{\text{v}}(t)\|^2}{N \sum_{t=1}^{T} \|\vec{\text{v}}(t) - \vec{v}(t)\|^2} \right) \tag{4.21}$$

where N is the number of lognormal strokes used in the reconstruction. The higher the measure, the better. Given two reconstructions with the same number of strokes, a bigger SNR_v is always preferable, and vice versa, if the same SNR_v is achieved, a smaller number of lognormal strokes is desirable. Moreover, this measure can also be estimated using the precalculated SNR_v value from Equation (4.20) using:

$$\text{SNR}_v/\text{nbLog} = \text{SNR}_v - 10 \log(N) \tag{4.22}$$

The third measure, the SNR_s, evaluates how similar is the reconstructed Cartesian coordinates with respect to the original ones. It is

calculated as the signal to noise ratio (SNR) between the original and the reconstructed shape:

$$\mathrm{SNR}_s = 10 \log \left(\frac{\sum_{t=1}^{T} [\mathrm{x}(t)^2 + \mathrm{y}(t)^2]}{\sum_{t=1}^{T} [(\mathrm{x}(t) - x(t))^2 + (\mathrm{y}(t) - y(t))^2]} \right) \tag{4.23}$$

where $\mathrm{x}(t)$ and $\mathrm{y}(t)$ are the original positions, whilst $x(t)$ and $y(t)$ are the reconstructed ones. The higher the measure, the better.

We use the SNR_s given that, in the context of handwriting or gesture recognition, shape deviations are usually more problematic than speed deviations, since some preprocessing techniques include a speed normalization step [12] and recognition systems usually rely on shape-based off-line information [100].

4.2 New Sigma-Lognormal Parameters Extractor

Although the results obtained by the Sigma-Lognormal Parameters extractor outlined in Section 4.1.3 are good, there is still room for improvement in order to reach better levels of accuracy, robustness and flexibility. Here we present a new powerful extractor to identify lognormal strokes and estimate their Sigma-Lognormal parameters.

In the following sections, we present the different steps performed to extract and estimate the strokes from a given signal. Given that many of these steps are common to the previous extractor, here we will focus on *preprocessing* and *stroke extraction*. After that, we conduct a rigorous experimentation using a public dataset to assess the capabilities of this new approach to identify and extract a better Sigma-Lognormal representation.

4.2.1 Preprocessing

As the extraction approach that we are going to use is velocity-based, it is more prone to spatial deviations as the input size gets larger. According to Equation (4.4), the velocity of the handwriting is the sum of each stroke velocity. Therefore, errors in the angle estimation are propagated over the whole signal, leading to increased spatial deviations for longer signals (see Figure 4.6) [79].

Intuition says that it would be most appropriate to divide the signal into smaller pieces. Where each piece could be analyzed and extracted independently, thus minimizing the accumulation of the spatial errors. For this reason, here a representation where the original signal is *chopped* into smaller pieces, is used. Here, segmentation is perfomed at pen-stroke level, preserving only those that are pen-down, which are in fact components [101].

Then, for each pen-stroke, its signal is interpolated using cubic splines and re-sampled to 200 Hz. Moreover, zero velocity truncation is enforced at the beginning and end of the velocity magnitude profile by keeping the signal artificially still at the start and end position for 50 ms sampled at 200 Hz.

4.2.2 Stroke Extraction

Given an initial velocity magnitude profile $\vec{v}(t)$, the goal is to account for the biggest amount of velocity, and therefore for the biggest amount of strokes present in the signal, by using lognormal equations. Each lognormal equation is represented by a set of characteristic points, which are obtained using the same technique as the one used in Section 4.1.3.

However, the stroke extraction method used in Section 4.1.3 is based on a greedy technique that extracts the sets of characteristic points from left to right. But the lognormal extraction order is really important. Given two partially overlapped strokes, the quality of the reconstruction if the leftmost is extracted first will be different than if the rightmost is extracted first. This is because the strokes tend to alter the characteristic points of their neighbors, due to the velocity overlap. In addition, the maximum velocity magnitude value of a lognormal stroke can be "hidden" under another stroke (see Figure 4.4; where a hidden local maximum is discovered in the second sub-figure by subtracting another stroke), thus being "invisible" to the extraction technique. On the other hand, according to the lognormality principle, the number of extracted strokes must be the minimized whilst the SNR_v must be maximized. That is, given two reconstructions with the same quality, the one with the lowest number of strokes is most desirable.

Considering the above, this problem can be expressed as (not only) estimating the minimum set of lognormal strokes (but also) in their best extraction order that reconstructs better the velocity magnitude profile. Therefore, a greedy approach is not powerful enough to solve this problem.

Here a *Breadth-first search* (BFS) algorithm [102] was used as a strategy to drive the extraction. BFS is an algorithm that expands level-by-level and examines all combination of sequences by systematically searching through every vertex.

As said before, the lognormal extraction order is really important. The BFS algorithm provides the extraction robustness required, given that it *estimates* all the combinations of extractions in different order. In addition, BFS is so named because it expands the frontier uniformly across the breadth of the frontier. That is, the algorithm discovers all the vertices containing partial extractions at level k before discovering any at $k+1$. Therefore this technique will always minimize the number of extracted strokes.

More technically speaking, BFS produces a *breadth-first tree* with root s that stores all reachable extractions in different vertices. For a given velocity magnitude profile stored in a vertex u, the characteristic points sets are found and a child vertex v is created for each one. Every sibling of v holds a lognormal equation, which is a reconstruction of one of the characteristic points set found in the velocity magnitude stored in u; as well as a reference to its parent u; an estimate of how good is this solution so far; and a new velocity profile magnitude, where the estimated velocity is subtracted from its parent velocity magnitude. Once this is performed, the BFS algorithm continues expanding a new level, in which now every vertex v becomes a parent vertex u. We must emphasize that, for each *expanded* parent vertex u, the sets of characteristic points must be recalculated. This is because, as said before, the subtraction of a lognormal stroke may discover other strokes that a priori were hidden. Figure 4.4 illustrates the stroke extraction using the BFS algorithm on a velocity profile magnitude sample.

Usually a breadth-first search ends when the vertex sought is found or, if it is not found, when every vertex is a leaf, that is, every vertex

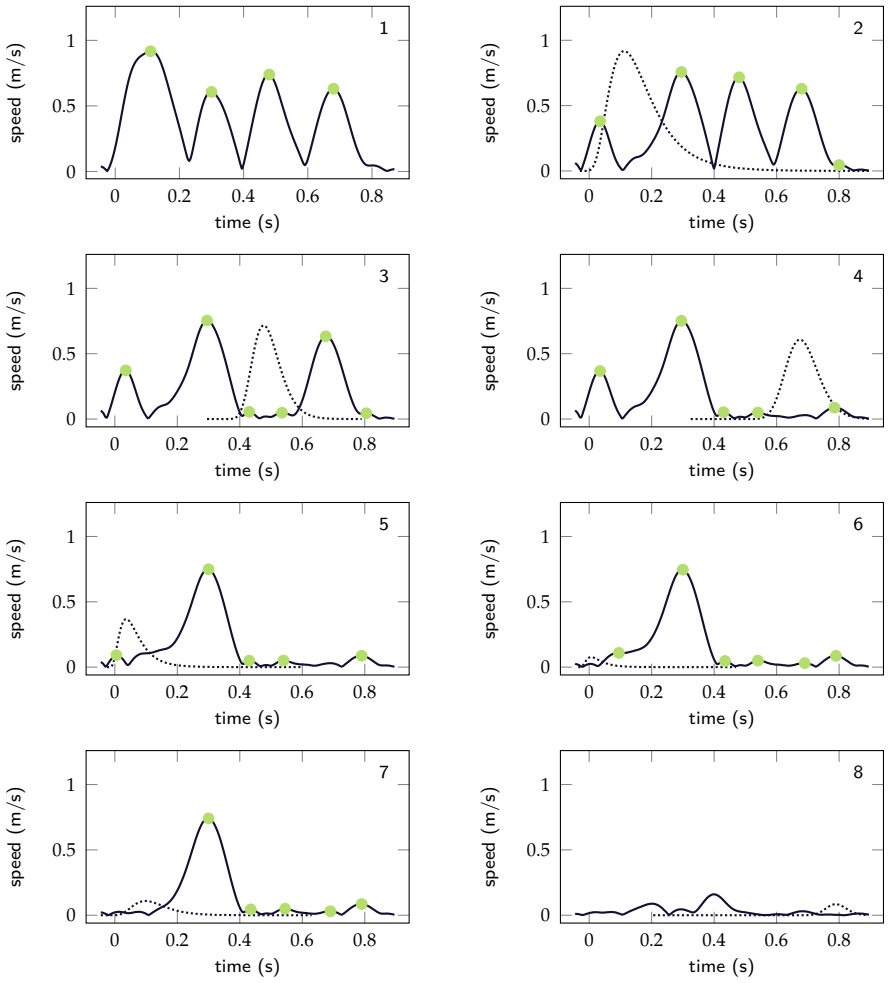


Figure 4.4. Step-by-step example of the Sigma-Lognormal extraction using the here proposed BFS-based extractor. The reading order is left-right, top-bottom. The velocity magnitude used is the same as in Figure 4.3. The solid black line indicates the velocity profile to be reconstructed, whilst the green dots indicate the $\|\vec{v}(t_3)\|$ siblings to be extracted. The dotted black line shows the lognormal equation extracted, and subtracted from the velocity magnitude, in the previous step. In this example, a beam width of 1 has been used and some steps have been omitted for the sake of clarity.

has no children. In this case, the BFS procedure is executed repeatedly until: a) a given vertex holds a reconstruction that is considered good enough. That is, its calculated SNR_v is bigger than a preset threshold; b) the height of the tree is bigger than a maximum value; or c) there is not any set of characteristic points left to explore. The a) option can be considered the optimal solution. If b) or c) options were the case, the best solution so far is returned. We tap on the lognormality principle to define what is the best solution so far, which is the solution with the highest value of SNR_v/nbLog (see Equation (4.21)).

The SNR_v/nbLog is used, instead of just SNR_v, to avoid overfitting. Let us assume a reconstruction that will never reach the SNR_v threshold. The extractor will try to *indefinitely* add new lognormal equations (modeling probably noise), that will only slightly increment the SNR_v value. Therefore we can not consider the best solution as the one with the highest SNR_v.

A beam search pruning approach is used to maintain tractability, given that the number of vertices to explore can easily explode because of the problematic discussed above. A priority queue, sorted decreasingly according to the values of SNR_v, is used to implement the beam search pruning. Thus for each level k, only a predetermined number (w) of the *most promising* vertices are kept, pruning the rest of the siblings. With an *infinite* beam width, no vertices are pruned and beam search is identical to BFS.

The procedure in Figure 4.5 shows how the extraction works. First, some attributes are attached to each vertex. The attribute n holds the number of extracted lognormals so far; two quality estimators (SNR_v and SNR_v/nbLog) are stored in the attributes m and r, respectively; the Sigma-Lognormal parameters are stored in the attribute g and the predecessor of the current node in the attribute p.

Lines 2 to 7 initializes the values of the different attributes of initial solution (s) and the best solution so far (b). Lines 8 and 9 initialize Q to the queue containing just the initial solution s. The *while* loop of lines Lines 10 and 29 and the *for* loop of lines Lines 11 and 27 iterate as long as there are remaining vertices. Line 12 initializes R as the auxiliary queue. R is necessary to implement the beam search prune. Lines 13 and 14 is the *optimal* stop criteria. If the current SNR_v

```
1:  procedure STROKE_EXTRACTION(signal)
2:      s.g = None
3:      s.p = None
4:      s.s = signal
5:      s.m = b.m = 0.0
6:      s. r = b.r = -inf
7:      s.n = 0
8:      Q = [ ]
9:      Q.append(s)
10:     while Q:
11:         R = [ ]
12:         for u in Q:
13:             if u.m >= m_min :
14:                 return u
15:             cps = identify_strokes(u.s)
16:             for cp in cps:
17:                 z.g = pars_estimation(cp)
18:                 z.s = u.s - getspeed(u.g)
19:                 z.m = calculate_snr(signal, u.s)
20:                 z.n = u.n + 1
21:                 z.r = calculate_snr_nblog(z.m, z.n)
22:                 z.p = u
23:                 if z.n >= nblogs_max :
24:                     continue
25:                 if z.r > b.r :
26:                     b = z
27:                 R.append(z)
28:         R.sort(key=lambda x: x.m)
29:         Q = R[ : w]
30:     return b
```

Figure 4.5. Pythonic pseudo code for the Sigma-Lognormal extractor. The meaning given by Python to statements, operators and methods has been used. Block structure is denoted using indentation.

value $u.m$ is bigger than the preset threshold, the procedure ends. Line 15 identifies the new set of characteristic points that are candidates (see Section 4.1.3). The *for* loop of lines 16 and 27 considers each set of candidate points (cp) in the list of sets of candidates. Line 17 performs the velocity and angle parameters estimation explained in Section 4.1.3. Line 18 calculates the velocity profile of the reconstruction using Equation (4.4). Lines 19 and 21 compute the SNR_v and SNR_v/nbLog using Equations (4.20) and (4.22), respectively. Line 20 increments the number of lognormals and line 22 records u as the parent of z. Lines 23 and 24 discard solutions with a high number of lognormals. Lines 25 and 26 are in charge of keeping updated the best solution throughout the procedure according to the highest value of SNR_v/nbLog ratio, as previously explained. Line 27 queues the new solution v in the auxiliary queue R. Lines 28 and 29 implement the beam search prune. Line 28 sorts the queue R taking into account the value of the SNR_v stored in $x.m$. Line 29 stores only the w most promising elements of the auxiliary list R in Q, where w is the *beam width*. If the procedure arrives to Line 30 without finding an *optimal* solution, it will return the best solution so far stored in b.

4.2.3 EVALUATION

We conducted a rigorous experimentation using two public datasets. The aim of this evaluation is to investigate whether the extractor presented in Section 4.2 provides a better Sigma-Lognormal representation. We compared our extractor against a baseline, the extractor described in Section 4.1.3 [85].

We evaluated both extractors taking into account quantitative criteria in terms of kinematics quality and shape fitness quality. The experimental details are described below.

4.2.3.1 Experimental Setup

The minimum value of SNR_v was set to 25 dB for both extractors. The beam search threshold for the new extractor was set to 2. Other parameters, common to both extractors, were set to the values proposed in [85].

We reconstructed samples from the following public datasets: the Unipen-ICROW-03 benchmark set and the IBM-UB Data Set (see Section 2.3.1). We chose these datasets as they are composed of isolated words.

A total of 13,119 on-line words from the Unipen-ICROW-03 benchmark set and 63,683 on-line words contained in the *query* part of the IBM-UB dataset have been reconstructed.

The IBM-UB dataset has been chosen for the experiments given that the samples of this dataset do not contain pen-ups. Thus the same preprocessing from Section 4.2.1 has been used for the baseline extractor. Therefore, results on this dataset can be used to make a *pure* lognormal extraction comparison, not influenced by the different preprocessing.

4.2.4 RESULTS

First, Figure 4.6 shows a visual comparison of the signal reconstruction using both extractors. Quality measures (SNR_v, SNR_v/nbLog and SNR_s) are shown in Figure 4.7. With respect to the Unipen-ICROW-03 dataset, the average SNR_v for the new extractor was 25.6 ± 2.2 dB, whilst the baseline extractor achieved 21.5 ± 5.9 dB. Here, we must remark that the baseline extractor is not able to achieve the minimum preset threshold of 25 dB. The average SNR_v value for the new extractor on the IBM-UB dataset was 27.2 ± 1.6 dB (mean $\pm$ SD), whereas for the baseline extractor was 25.4 ± 2.5 dB. In this dataset, both extractors are able to surpass the minimum present threshold of 25 dB. The differences in the SNR_v averages were found to be statistically significant for both datasets (Welch's two-sample t-test, $p < .0001$). According to these results, the new extractor provides a better (higher average SNR_v) and more consistent (lower SD values) velocity magnitude profile reconstruction.

The new extractor achieved an average value of 41 ± 14 extracted strokes, whilst the baseline approach had an average of 30 ± 11 extracted strokes on the Unipen-ICROW-03 dataset. With respect to IBM-UB dataset, an average of 40 ± 15 extracted strokes with the new extractor whilst the baseline extractor had a median value of 43 ± 18 extracted strokes. The differences in the the average number of ex-

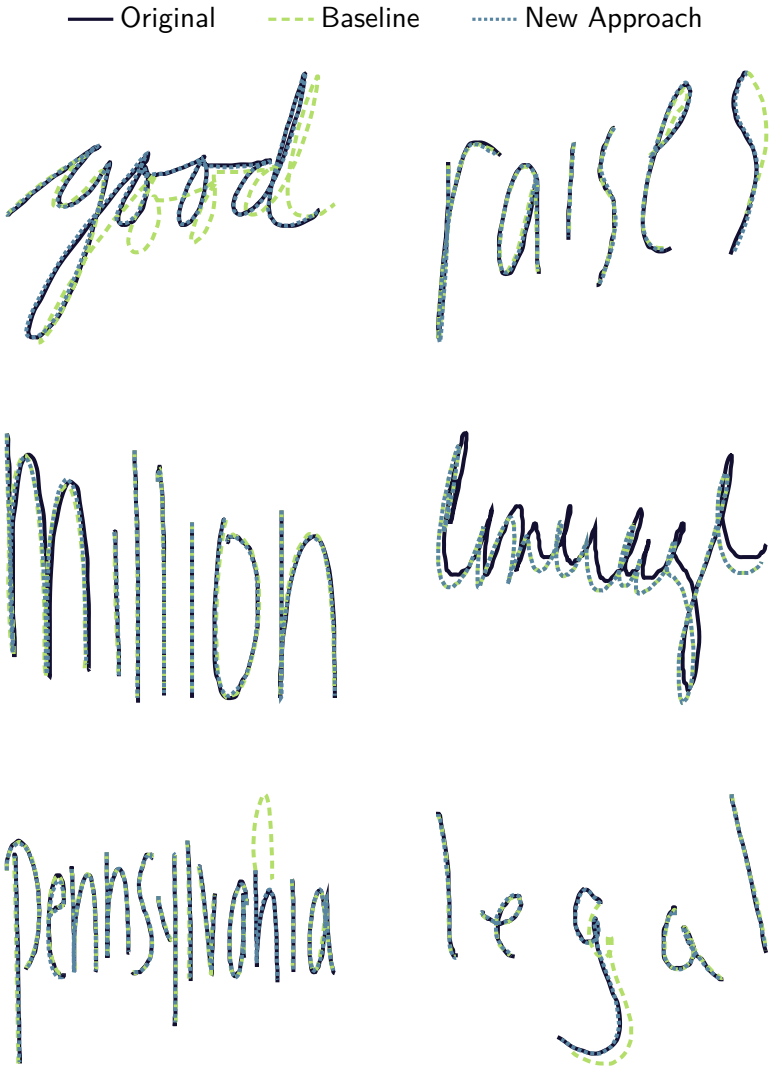


Figure 4.6. Reconstruction examples using both extractors. The solid black lines are the original signals, the dotted blue line are the reconstruction using the new approach and the dashed green line are the reconstruction using the baseline approach.

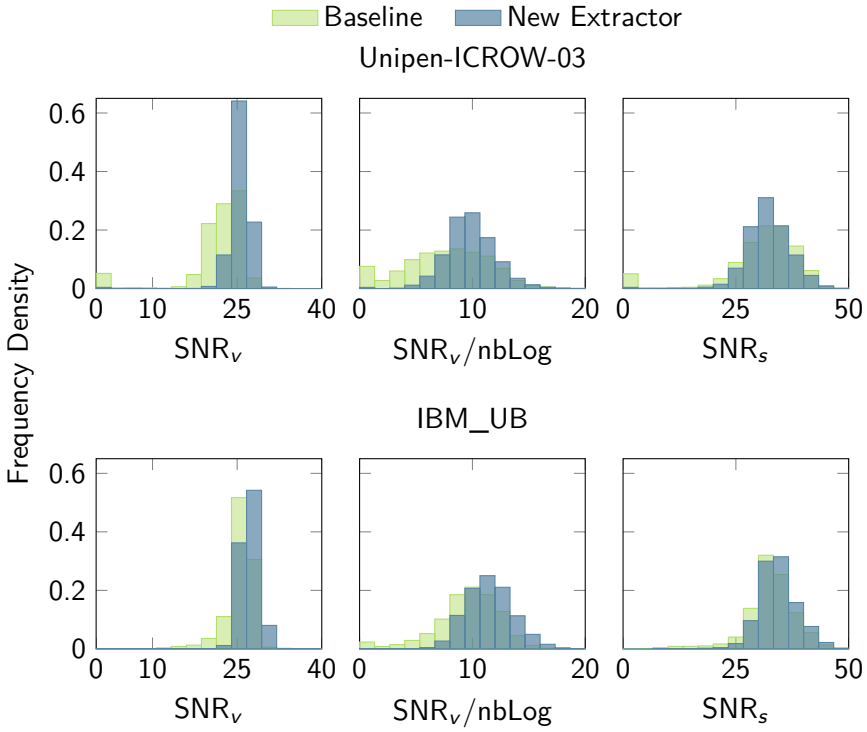


Figure 4.7. Density histograms showing the quality measures.

tracted lognormals were found to be statistically significant for both datasets (Welch's two-sample t-test, $p < .0001$). The new extractor obtains a more similar value in both datasets. However, if we take into account the SNR$_v$/nbLog ratio, the baseline extractor achieved an average value of 9.8 ± 2.2 dB per lognormal, whereas the new extractor achieved 7.5 ± 4.0 dB per lognormal for the Unipen-ICROW-03 dataset. As we can see, although the number of extracted lognormals by the new approach was higher for this dataset, we can see that its SNR$_v$/nbLog is higher. With respect to the IBM-UB dataset, the baseline obtained an average value of 9.5 ± 3.1 dB per lognormal, whereas the new extractor achieved 11.5 ± 2.2 dB per lognormal.

Taking into account the SNR$_s$, the average value on the Unipen-ICROW-03 dataset for the new extractor was 32.2 ± 5.0 dB and for the baseline approach was 30.6 ± 9.1 dB. The average SNR$_s$ on the IBM-

	nbLog	SNR_v	SNR_v/nbLog	SNR_s
Unipen-ICROW-03				
New Extractor	41 ± 14	25.6 ± 2.2	9.8 ± 2.2	32.2 ± 5.0
Baseline	30 ± 11	21.5 ± 5.9	7.5 ± 4.0	30.6 ± 9.1
IBM-UB				
New Extractor	40 ± 15	27.2 ± 1.6	11.5 ± 2.2	34.3 ± 4.4
Baseline	43 ± 18	25.4 ± 2.5	9.5 ± 3.1	32.5 ± 5.7

Table 4.1. Summary of the extraction results for the IBM-UB (top) and ICROW-03 (bottom) datasets. From left to right, average value $\pm$ SD for: the number of extracted lognormals (nbLog), SNR_v, SNR_v/nbLog and SNR_s. All measures, except nbLog, are expressed in dB.

UB dataset for the new approach was 34.3 ± 4.4 dB and 32.5 ± 5.7 dB for the baseline extractor. According to a Welch's two-sample t-test, the SNR_s average differences between both approaches are statistically significant ($p < .0001$). Thus the new extractor provides a better (higher average SNR_s) and more consistent (lower SD values) shape reconstruction.

As a final overview, Table 4.1 shows a summary of these results. According to them, the new Sigma-Lognormal extractor provides a significantly better performance than the baseline extractor.

4.3 Discussion

Although in this chapter lognormal-based models have been mainly applied to handwriting analysis, many studies have shown that they can be successfully applied to other types of movements. For example: reproducing wrist movement and eye saccades [89], 2D and 3D arm movements [103], and more recently, stroke gestures [104]. In other words, the Kinematic Theory provides a complete parametric representation space to study motor control behavior. This mathematical demonstration suggests that the asymptotic convergence toward lognormal impulse responses and velocity patterns can be interpreted as reflecting the behavior of subjects who are in total control of their movements. Among other interesting findings, it has been

shown a migration toward lognormality as young children grow up and the deviation from lognormality with aging [77]. Additionally, from a mathematical point of view, the Kinematic Theory is a theory of convergence toward smoothness. The lognormal function is an optimal descriptor of the velocity profiles: the smoothest velocity being reached when the energy associated with the convergence error toward lognormality is minimized. As such, the Kinematic Theory can be considered as an ultimate minimization theory.

Given that a stroke minimization is assumed, the reconstruction of a signal with the less strokes is more desirable. The new approach is able to minimize the number of extracted lognormals surpassing the preset SNR_v threshold, and thus to maximize the SNR_v/nbLog ratio, thanks to the breadth-first search technique used to control the stroke extraction.

However, what is preferable: a reconstruction with the biggest SNR_v possible or a reconstruction that surpasses a certain preset SNR_v threshold? That is, should the extraction technique reconstruct perfectly the handwriting (including noise from different sources and therefore incurring in over-fitting)? The answer to this questions is "probably depends on the purpose of the reconstruction". For example, if the purpose is to analyze a subject handwriting on medical grounds, is likely that a very high SNR_v value is desired. Given that, the reconstruction needs to model every little detail that might be indicative of some type of medical disorder [81,105]. On the other hand, if the purpose is to generate synthetic handwriting, it could be interesting to require a smaller SNR_v value, using the Sigma-Lognormal extraction as a pre-processing technique itself, removing some of the error that the capture device could have introduce. This statement is supported by previous works, where different degrees of reliability were suitable to model different forms of handwriting. For example, a $\text{SNR}_v \geq 15$ dB is enough to model gestures [99], whilst handwritten text requires $\text{SNR}_v \geq 25$ dB [98] and signatures a $\text{SNR}_v \geq 30$ dB [83, 84]. Our new approach not only allows tuning this parameter, but also maximizing the SNR_v/nbLog ratio, that is, minimizing the number of lognormals and maximizing the SNR_v. So the user is able to set a smaller value where coarser reconstructions are permissi-

ble, whereas a bigger value can be set if a very accurate reconstruction is required.

A current limitation of our Sigma-Lognormal extractor is the time complexity. Given that a BFS is performed, using the big O notation, the time complexity, can be expressed as $O(w^s)$, where w is the beam width and s is the number extracted lognormals. This high computational cost can be reduced using a data structure to store precomputed primitives. In this case, the number of strokes that it would be necessary to estimate could be greatly reduced, given that a high number of lognormal equations are shared between the current state and its successors as well as with the *neighboring* intermediate solutions.

4.4 Conclusion

In this chapter, a new method for the Sigma-Lognormal parameters extraction has been proposed. Different experiments have been performed to evaluate this new Sigma-Lognormal extractor. The new approach achieves excellent results, outperforming the state-of-the-art Sigma-Lognormal parameters extractor in terms SNR_v, SNR_v/nbLog and SNR_s.

5 Synthesizing Pen & Touch On-line Strokes

A pen computer relies heavily on a recognizer, as it gives a meaning to the user stroke-based interactions. The recognizer accuracy depends mostly on the size of the training set used. As a rule of thumb: the bigger the training set, the better the recognition accuracy.

Usually it is not possible to have a sufficiently large training set using natural human samples. Recruiting participants, data collection, labeling, etc. necessary for achieving this goal is rather time-consuming and expensive.

One way to overcome this problem is to create and use synthetically generated training data that *looks and feels* like human. Traditionally, the use of synthetic samples has been regarded as bad idea. However, stroke synthesis, which is the artificial generation of human-like strokes, has recently become a hot topic with increasing interest [80, 83, 106, 107].

In this chapter, we tap into the Kinematic Theory and its Sigma-Lognormal model to generate synthetic human-like strokes. We propose and evaluate 2 different scenarios where synthetic samples are employed as training data. In the first case, synthetic samples are used to enlarge the existing training set with the objective of reducing the error rate. In the latter, synthetic samples are used instead of real ones. This experiment aims to reduce the human effort while collecting data.

This chapter is organized as follows. Section Section 5.1 is a review of different stroke synthesis techniques. After that, in Section 5.1.3, a technique to synthesize strokes is reviewed. Section 5.2.1 presents and tests different scenarios where synthetic samples are used as train-

ing. Finally, conclusions are drawn in Section 5.3 and Section 5.4, respectively.

5.1 Overview of Stroke Synthesis

Stroke synthesis techniques can be divided into two categories according to their principles: "bottom-up" approaches, also called shape simulation, where the outcome is generated or "top-down" approaches, which are also called movement simulation, where the neuromuscular acts of writing are simulated [108].

5.1.1 Shape-simulation Synthesis Techniques

Shape-simulation synthesis can be classified into generation or concatenation techniques. Generation techniques synthesize a new sample for a given writing unit, whilst concatenation techniques join primitives to form letters or words. At the same time the generation techniques can be subdivided into:

Perturbation-based techniques: This group of techniques alter a sample to obtain a new one. The alterations include size, thickness and slant modifications at stroke [109] or line level [110]. Perturbation-based techniques are easy to apply, but the results can be unnatural due to non-calibrated parameter settings [111].

Fusion-based techniques: These techniques take some samples and fuse parts of them to generate a new one [112].

Model-based techniques: They capture the variations in handwriting from many samples to generate a model [108,110,113–115].

Concatenation techniques refers to any synthesis approach that combines input samples to generate outputs. The usual input units are characters or words, although sub-characters has been also used. With respect to concatenation techniques, these can be subdivided according to the type of connection they use:

No-connection techniques: This group of techniques paste input units, one next to another to form output units of higher semantic level [116, 117].

Direct-connection techniques: These techniques paste letters together such that their ending ligatures directly connect to the starting ligature of the next letter [118].

Modeled-connection techniques: These techniques add new connection ligatures synthesized by parametric curves [108–110, 113].

5.1.2 MOVEMENT-SIMULATION SYNTHESIS TECHNIQUES

Movement simulation is a top-down approach to stroke synthesis where the neuromuscular system involved in the generation is simulated.

Hollerbach [119] approach to stroke synthesis, proposes to model strokes as horizontal and vertical oscillations. The horizontal one controls the stroke shape, whilst the vertical controls the height. Based on the oscillatory approach proposed by Hollerbach, Gangadhar et al. [120] used a neural-network-based model where stroke velocities were expressed as oscillatory neural activities. Simard and LeCun [121] presented an approach where time trajectories were modeled using an oversampled reverse time delay neural network architecture. This network generate outputs that can control the pentip. Bayoudh et al. [122] approach proposes to use the principle of analogical proportion to synthesize new samples from an existing limited set of real samples. Here, each character is modeled as a sequence of Freeman chain codes including a set of anchorage points. Slim et al. [123] proposes to model strokes using electromyographic signals obtained from the forearm muscles. A radial basis function (RBF) neural network learns how to generate the Cartesian coordinates using the electromyographic signals. Finally, one of the most notable synthesis techniques is based on the ideas presented by Plamondon. Djioua and Plamondon [97] presented a technique for synthesizing handwriting strokes using the Kinematic Theory and its Sigma-Lognormal model.

Figure 5.1. Five samples of the word *home* written by the same person.

In the following section, the origin of handwriting variability according to the Kinematic Theory is explained, as well as its associated synthetic generation technique.

5.1.3 Synthesizing Strokes Using the Kinematic Theory

As shown in Figure 5.1, if a person is asked to write something in particular several times, stroke variability can be observed. Different kinematic models provide insights with respect to the origin of the "infinite" variability observed in handwriting.

For example, the Kinematic Theory predicts that human variability is produced by modifications in the action plan and/or due to the intrinsic variability of each stroke. This statement is supported by different studies that have demonstrated the strong connection between handwriting variability and the distortion of the Sigma-Lognormal parameters [80, 124]. Variations in μ_i and σ_i mimic peripheral noise, like a person who instantiates the same intention and executes it with an upper limb slightly different from one trial to another. On the other hand, the t_{0_i}, D_i, θ_{s_i} and θ_{e_i} variations refer to central fluctuations that might occur in the temporal or geometric position of the action plan from one trial to another, reflecting, for example, attention changes.

These fluctuations can be simulated by introducing local and/or global variations to the Sigma-Lognormal parameters. We understand as local variations those that only affects one stroke, whereas global variations affect all the strokes that compose a movement. In practice, local and global fluctuations are mixed, so different deformations can

be created by varying these parameters

$$\begin{aligned} D_i' &= D_i \pm K \pm k_i \\ \theta_{s_i}' &= \theta_{s_i} \pm P_s \pm \rho_{s_i} \\ \theta_{e_i}' &= \theta_{e_i} \pm P_e \pm \rho_{e_i} \\ t_{0_i}' &= t_{0_i} \pm \tau_i \\ \mu_i' &= \mu_i \pm M \pm \Delta\mu_i \\ \sigma_i' &= \sigma_i \pm S \pm \Delta\sigma_i \end{aligned} \tag{5.1}$$

where local variability is presented by lowercase Greek letters, whilst global variability is represent using uppercase Greek letters.

Several predictions can be made concerning variability. First, in terms of global fluctuations, variability of the parameter D_i produces an enlargement or shrinkage of the handwriting. Moreover, variations of the directional parameters (θ_{s_i} and θ_{e_i}) produces rotational effects. Peripheral parameters (μ_i and σ_i) variations, involves smoothing and sharpening effects in the handwriting. This is because modifications of these parameters causes an increase or decrease in the rate of overlapping between neighboring strokes, leading to smoother or sharper trajectories. The global variability of t_{0_i} does not result in any pattern deformation. The local fluctuations of each parameter lead to non-uniform deformations, characterized by local scale changes, rotations, smoothing and sharpening.

An example of applying distortions to the Sigma-Lognormal parameters can be observed in Figure 5.2. As can be seen, variability in D_i affects in the scale of the trajectory and the velocity but not in the profile. Moreover, variability in θ_{s_i} and θ_{e_i} affects the trajectory but not the velocity. But fluctuations in μ_i affects both the trajectory and the velocity profile magnitude. These variations leads to a smoother trajectory. On the other hand, alterations in σ_i affects also both the trajectory and the velocity profile magnitude, but in this case these variations leads to a sharper trajectory. Care should be taken with alterations of t_0, given that this parameter is very sensitive even to small variations. Finally, local variations affect locally the trajectory and the velocity profile.

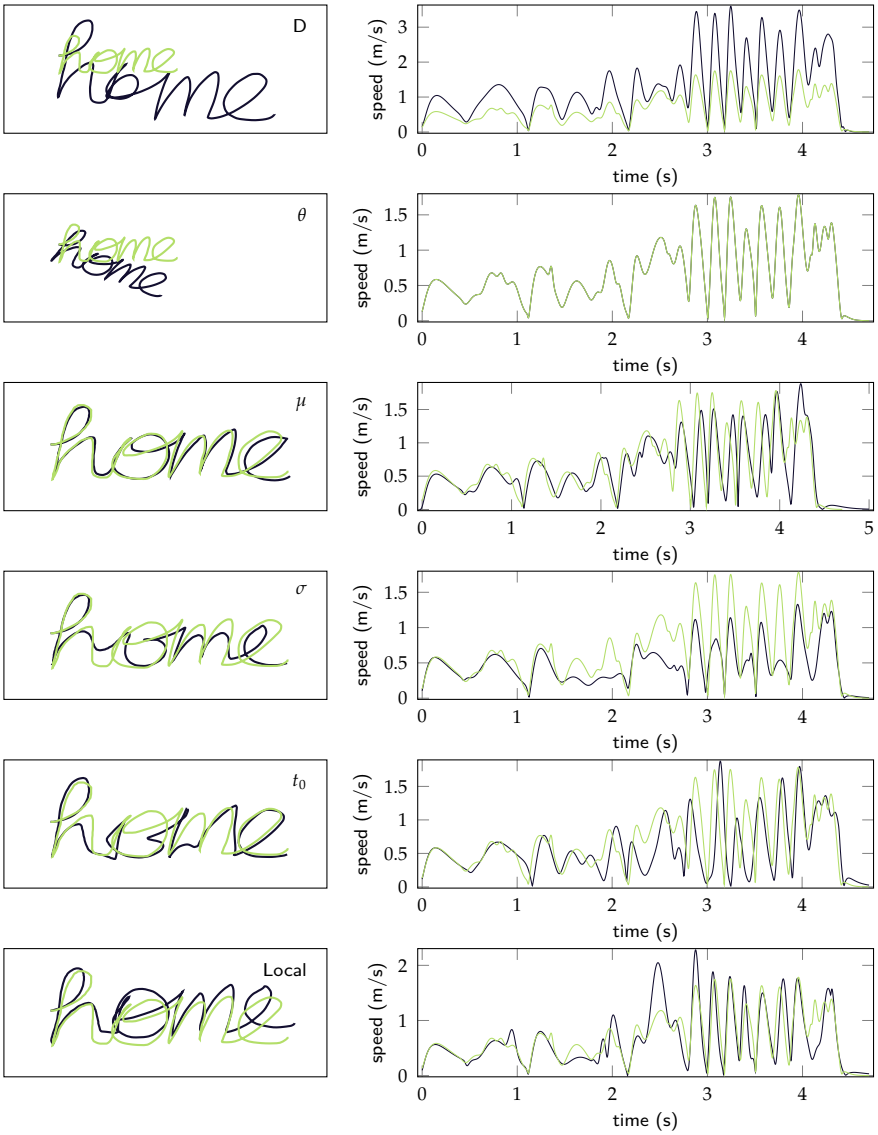


Figure 5.2. The green lines are the original handwriting, whereas black lines are the handwriting using variations in the Sigma-Lognormal parameters. First row: Global variations ($K = 2$) in D_i. Second row: Global variations ($P_s = P_e = 0.4$) in θ_{s_i} and θ_{e_i}. Third row: Global variations ($M = 0.1$) in μ_i. Fourth row: Global variations ($S = 0.1$) in σ_i. Fifth row: Variations of ($\tau_i = 0.1$) t_0 Sixth row: Local variations ($k_i = 0.1$, $\rho_{s_i} = \rho_{s_e} = 0.05$, $\tau_i = 0.0$, $\Delta\mu_i = 0.1$, $\Delta\sigma_i = 0.1$).

5.2 Using Synthetic Samples for Recognition Task

A serious problem in automatic recognition is the dependency of virtually all available recognition methods on large amounts of training data. Any method for handwriting or gesture recognition needs to be trained (e.g., neural networks, nearest neighbors classifiers, hidden Markov models, or support vector machine).

The most straightforward way to expand the training would be to collect additional human samples. But collecting real samples is a rather expensive and time consuming process. Alternatively, the training set can be expanded or replaced with synthetic samples.

Here we follow the approach reviewed in Section 5.1.3 for the generation of synthetic samples. A number of previous works have shown that the distortion of the Sigma-Lognormal parameters results in the generation of realistic human-like synthetic signatures [124], which in turn improves an existing recognizer's accuracy [80].

So far, this technique has been normally evaluated from the perspective of classification performance and not from the perspective of how "human-like" the synthesized gestures really are. A notable exception is a study by Galbally et al. [84] that examined the human likeness of synthetic handwriting. While it was found a high degree of similarity between synthesized and human handwriting, it is unclear whether this will hold for stroke gestures.

We address this open question in Appendix A, where we show that the Kinematic Theory produces stroke gestures that "look and feel" the same as human-generated gestures. We used relative measures to compare geometric, kinematic, and articulation aspects of thousands of human and synthetic gestures and found no practical differences between both populations.

In the following section we propose (and evaluate) 2 different scenarios where synthetic samples—handwriting and gestures—are employed as training data. In the first case, synthetic samples are used to enlarge the existing training set with the objective of reducing the error rate. In the latter, synthetic samples are used instead of real ones. This experiment aims to reduce the human effort while collecting data.

5.2.1 EVALUATION

We conducted a rigorous experimentation over different public datasets, in order to illustrate the value of this synthetic sample generation technique applied to handwriting and gesture recognition.

First, in Section 5.2.2.1, we perform a study regarding synthetic gesture variability. If we just copy one gesture sample over and over again to artificially increase the number of samples, we could incur in a case of overfitting, since the recognizer may not learn the "true" nature of each gesture class.

Then, in Section 5.2.2.2, we perform an experiment where we investigate the effect of including user-specific synthetic handwriting in a multi-writer dataset in contrast to use only real samples. This could be really useful within an interactive framework (as in Chapter 3), where user feedback could be incrementally used to adapt the recognition system to a specific user.

After that, in Section 5.2.2.3, different experiments are carried out to illustrate the value of this technique as a means to replicate human-generated datasets, in particular, gesture datasets. We compared the performance of synthetic samples with that of human samples in terms of: articulation speed, size of gesture vocabulary and input device.

5.2.1.1 Experimental Setup

We generated synthetic samples for the following public datasets: $1-GDS, MMG, chars74k[1] and Unipen-ICROW-03 (see Section 2.3.1 for more details).

Each sample in every dataset was modeled according to Equation (4.4) using the Sigma-Lognormal extractor presented in Section 4.2. After that, the Sigma-Lognormal parameters were distorted according to:

$$p'_i = p_i + \mathcal{U}(-n_{p_i}, n_{p_i}) \tag{5.2}$$

where p_i are the Sigma-Lognormal parameters, $\mathcal{U}(a,b)$ is a continuous uniform distribution and $n_{p_i} = \{k_i, \rho_{s_i}, \rho_{e_i}, \tau_i, \Delta\mu_i, \Delta\sigma_i\}$ using

[1]No timestamps are available in the chars74k dataset, so temporal information must be estimated for reconstruction from the given sampling rate.

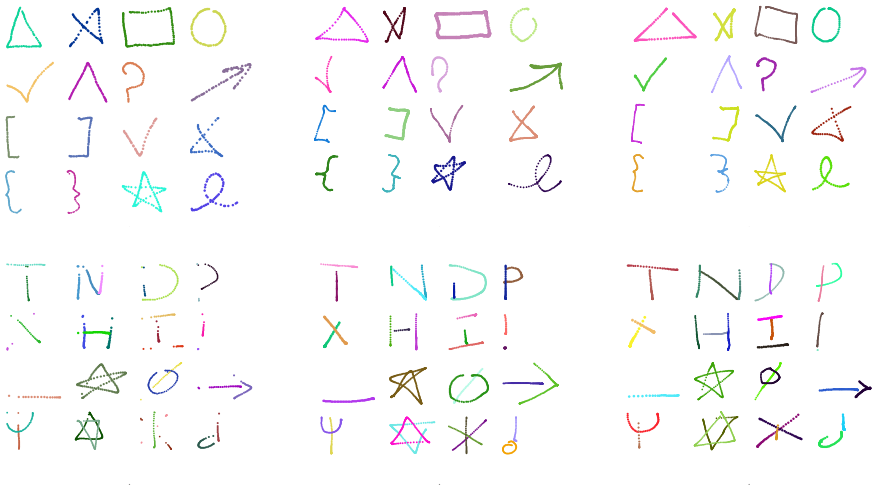


Figure 5.3. Human and synthetic samples, all samples picked at random. The samples on the left column are human generated, whilst center and right columns are synthesized.

$k_i = 0.15$, $\rho_{s_i} = \rho_{e_i} = 0.06$, $\tau_i = 0.005$, $\Delta\mu_i = \Delta\sigma_i = 0.1$ as noise values [84]. Finally, the Cartesian coordinates (x,y) were retrieved using Equation (4.7). Figure 5.3 provides a comparison of synthesized an real samples of the $1-GDS and MMG datasets.

5.2.2 RESULTS

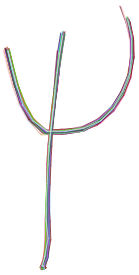
Below we describe each experiment and show its results.

5.2.2.1 Synthetic Gesture Variability

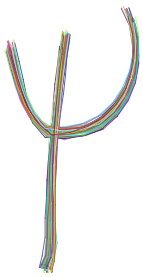
Here, we compare the variability of the synthesized samples against their original human samples, using different values of variability. We investigated this topic with the $1-GDS, MMG and chars74k datasets.

Samples were synthesized in batches of $N \in \{10, 100, 1000\}$ elements each using a modified version of Equation (5.2):

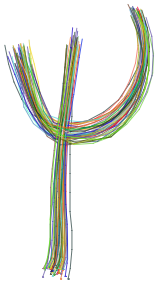
$$p'_i = p_i + \xi \cdot \mathcal{U}(-n_{p_i}, n_{p_i}) \tag{5.3}$$



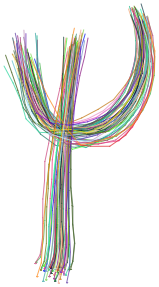
(a) $\xi = 0.2$



(b) $\xi = 0.5$



(c) $\xi = 0.7$



(d) $\xi = 1.0$

Figure 5.4. Visualizing the effect of the ξ parameter on gesture variability while synthesizing 50 samples.

where $\xi \in \{0.0, 0.5, 1.0\}$. Figure 5.4 illustrates the visual effect of this parameter on gesture variability. Then, we computed the mean squared error (MSE) of each synthetic sample with respect to the human sample from which it was generated. MSE values close to zero indicate that synthetic samples and their human counterparts look very similar. In contrast, higher MSE values indicate that synthetic samples diverge in shape from their human counterparts. Strokes were resampled in such a way that a human sample and its synthesized samples had the same number of points. MSE was averaged for each batch, variability level, and dataset.

Table 5.1 shows the results. As expected, it was found that synthetic samples are more variable as ξ increases. In general, we observed that requesting a small number of synthetic samples (10 samples per gesture) provides slightly less variable samples. Interestingly, for a given value of ξ, variability was found to increase as the number of requested synthetic samples increases, though we suspect it is because the MSE is underestimated for small batch sizes. Indeed, the standard error (SE) gets smaller as the number of samples gets larger, because the mean of a large sample is likely to be closer to the true population mean.

We also examined the intra-class variability of human gestures, distance-wise; i.e., how variable is a human gesture sample as compared to the rest of the human samples that belong to the same gesture class. No difference was found regarding the number of requested synthetic samples. The Pearson's correlation coefficient was found to be greater than 0.94 in *all* datasets, which indicates, for the human datasets, a large agreement regarding how users articulated gestures. Then, comparing synthetic gestures with their human counterparts resulted in Pearson's correlation coefficients decreasing as ξ increased; see Table 5.2. This was unsurprising and indicates that samples synthesized with a low variability degree look much more similar to the human samples from which they were generated.

5.2.2.2 Using Synthetic Samples on Writer Adaptation

For a given handwriting recognition task, a user-specific system will outperform a user-independent system. This statement is true as long

N	ξ	$1-GDS			MMG			chars74k		
		Mean	SD	SE	Mean	SD	SE	Mean	SD	SE
10	0.0	554.9	715.9	56.6	169.7	175.7	9.5	75.2	101.3	3.6
	0.5	579.4	774.4	61.2	250.1	271.7	14.7	359.8	407.6	14.7
	1.0	593.6	731.1	57.8	490.4	704.9	38.2	1181.5	1449.3	52.5
100	0.0	554.9	713.9	17.8	169.7	175.7	3.0	75.2	101.3	1.1
	0.5	576.8	754.7	18.8	256.1	273.4	4.6	377.2	408.3	4.6
	1.0	622.1	823.1	20.5	493.4	628.2	10.7	1298.9	1448.9	16.6
1000	0.0	554.9	713.6	5.6	169.7	175.7	0.9	75.2	101.3	0.3
	0.5	572.5	741.3	5.8	261.8	280.3	1.5	386.4	426.8	1.5
	1.0	621.4	813.3	6.4	498.7	646.0	3.5	1316.2	1452.7	5.2

Table 5.1. Gesture variability results for different number of synthesized samples and different values of the parameter ξ (see Figure 5.4 for each dataset. From left to right: MSE mean value, standard deviation (SD) and standard error (SE).

Dataset	Pearson's ρ		
	$\xi = 0.0$	$\xi = 0.5$	$\xi = 1.0$
\$1-GDS	0.94	0.94	0.93
MMG	0.94	0.93	0.89
chars74k	1.00	0.99	0.97

Table 5.2. Correlation for different values of the parameter ξ. Batch size did not make any difference in this study.

as the training set is big enough to obtain a good estimate of the user writing style. Usually this is not possible, since asking the user for a certain amount (usually big) of training samples is rather cumbersome and tedious. Under these conditions, one way to improve the system performance is to make use of the some multi- user existing knowledge, so that only a minimum amount of user-specific training data is sufficient to model the new writing style. Such a training procedure is often referred to as user or, in this case, writer adaptation.

Here we investigate the effect of adding synthetic samples in a multi-writer dataset in contrast to use only real samples. Furthermore, we analyze the impact of the number of collected samples from the specific writer used for adaptation. This experiment was conducted using the Unipen-ICROW-03 dataset and a HMM recognizer with a configuration similar to Section 2.1.1. The Unipen-ICROW-03 dataset was split as follows. We defined a training set, called `trn`, composed of 10,496 words from 56 writers. From the remaining 16 writers, which contained 2,623 words altogether, we randomly split 70% of the words as a test set, named `tst`, and the rest (a 30%) as an adaptation set, called `adp`. Therefore, the number of words per writer on average was 112 for `tst` and 50 for `adp`. To prevent the results from being influenced by the choice of these two partitions, we performed five trials. Results will we reported as the average for all trials.

In addition, a closed 1-gram language model was used here. The underlying vocabulary consists of 884 words, which included all the words seen in `trn`, `adp` and `tst` (some of these words appear in `adp` and/or `tst`, but not in `trn`). The language model was trained taking

		# replicated or synthetic samples (s)					
		10	20	50	100	150	200
# writer-specific samples (w)	20	11.7	11.4	10.2	9.5	9.5	8.9
	35	10.8	10.2	9.4	8.2	8.2	8.6
	50	10.4	9.8	8.7	8.1	8.5	8.5
	20	11.0	10.4	9.5	8.9	8.5	8.4
	35	9.9	9.1	7.9	7.0	6.6	6.4
	50	9.6	8.8	7.6	7.0	6.9	6.9

Table 5.3. Test set recognition error (%) for different values of s and w. Top: Using only real replicated adaptation samples. Bottom: Using both real and synthetic adaptation samples. Results are averaged for all writers.

into account the frequency of occurrence of the words which appear in trn and it is uniform for the remaining words.

To carry out this experiment, various morphological models have been created varying the amount $(20, 35, 50)$ of words (w) chosen from `adp`. We generate a number $s \in \{10, 20, 50, 150, 200, 250\}$ of synthetic words for each real word. For example, the model with $w = 20$ and $s = 150$, has 20 writer-specific samples plus 3000 $(20 \cdot 150)$ synthetic samples, making a total of $3,020$ words (in addition to the multi-writer data). As we aim to know whether the human-like variability present in the synthetic words improves the recognition results, we compare this approach with a baseline case, where only real samples are used for adaptation. Analogously to the approach using synthetic samples, we create the same number of models, but instead of using synthetic samples, we replicate each real word s times. This replication is equivalent to weigh the importance of the new sample relative to the rest [125]. This way, we can make a fair comparison between this baseline and the approach using synthetic words.

Table 5.3 shows the recognition error performance for the baseline and different morphological models using real and synthetic samples. The results presented are averaged for all writers and for the five proposed trials. The baseline approach obtained a 10.3% error rate. The approach using synthetic samples obtained a recognition error of

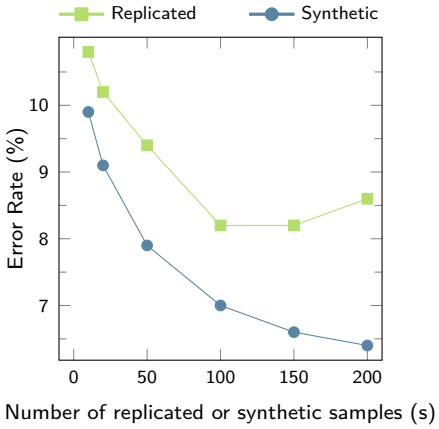


Figure 5.5. Evolution of the recognition rate for $w = 35$ real words and a varying number of s.

Error Rate			Rel. Improv.	
Baseline	AR	AS	Baseline	AR
10.3	8.2	6.4	37.9	22.0

Table 5.4. Best recognition error rates for the different approaches. From left to right: baseline scenario without using adaptation data (*Baseline*); adaptation using replicated data (**AR**) and adaptation using synthetic data (**AS**). Last 2 columns show the relative improvement between **AS** and *Baseline* and relative improvement between **AS** and **AR**. All results are percentages and averaged for the different writers and trials.

6.4% (using 200 synthetic samples for each real word), outperforming the best recognition rate of the adaptation baseline (8.1% replicating 100 times each real sample).

Figure 5.5 shows the evolution of the recognition rate when using 35 real samples ($w = 35$) and a varying number of synthetically generated or replicated samples. As we can see, the slope of the ER using replicated samples is less steep than the one using synthetic samples. The ER using synthetic samples achieves a better overall result than the approach using replicated samples. Moreover, the scenario using replicated samples suffers from overfit (ER starts to increase at the

range of 150-200 samples), which is not apparent for the scenario using synthetic samples.

Finally, Table 5.4 shows a summary of the best results using replicated and synthetic samples. As we can see, the adaptation approach improves by 22% the accuracy of the best adaptation baseline and by 38% the baseline scenario (no adaptation).

5.2.2.3 Replicating Human Gesture Samples

The objective of the following experiments is to compare the recognition accuracy of a system trained with only real gestures and another trained with one real sample (generation template) plus a set of synthetic gestures, having the same number of training samples in both cases. $ family of recognizers and DTW (see Section 2.2.1) have been used in the experiments.

Two different human gestures were chosen as generation template: the gesture sample with the smallest SNR_v (as long as it is more than 15 dB) and the gesture sample with the highest SNR_v. After that, two different sets of synthetic samples are generated from them. $Synth^-$ are synthetic samples generated using the gesture template with the smallest SNR_v and $Synth^+$ are the synthetic samples generated using the gesture template with the highest SNR_v. The purpose of this is to assess whether the reconstruction quality affects the generation of synthetic samples.

We conducted a number of user-dependent and user-independent tests. With respect to user-dependent, for each user, the recognizer is trained using a number of the user's gesture examples, and one example is used for testing. This is repeated for all users, and results are aggregated. Each user provided 10 examples of each gesture, so we increased the number of templates from 1 to 9. On the other hand, regarding user-independent tests, we used a leaving-one-user-out procedure: for each user, the recognizer is trained using the rest of the users and one user is left out for testing. This is repeated for all users, and results are aggregated.

As commented before, we compared the performance of synthetic samples with that of human samples in terms of: articulation speed,

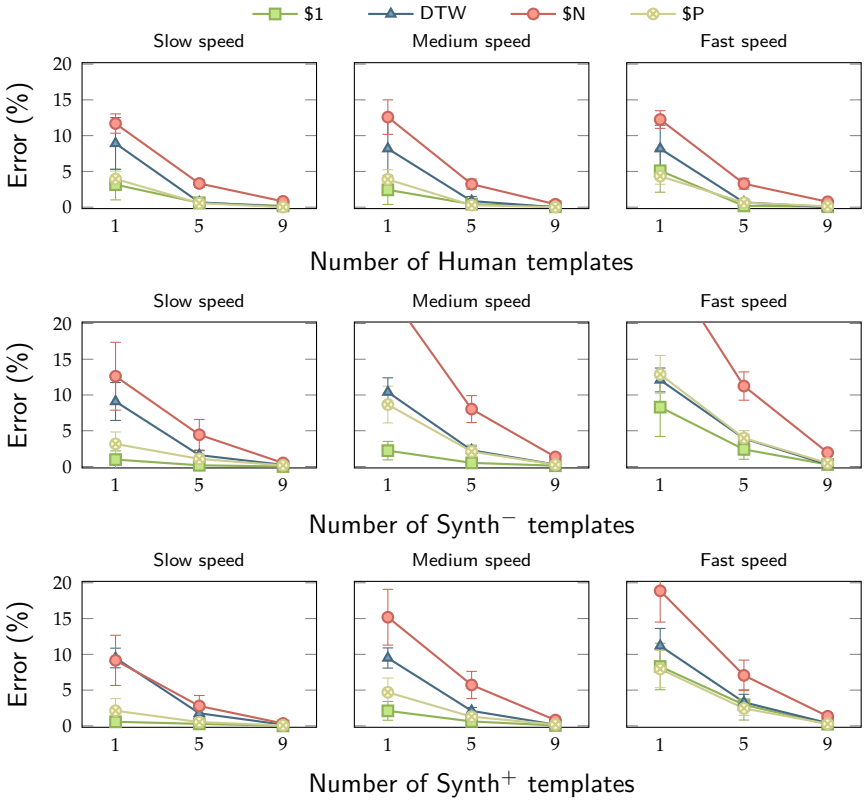


Figure 5.6. Impact of synthetic samples on articulation speed. User-dependent tests. Error bars denote 95% confidence intervals. Synth$^-$ and Synth$^+$ denote synthesized samples using the worst and best reconstructed human sample of each gesture, respectively.

size of gesture vocabulary and input device. The experiment taking into account different articulation speeds was conducted over the $1-GDS and MMG gesture datasets, as they were the ones that provided up to 3 articulation speeds: slow, medium, and fast. We tested the $1-GDS dataset with $1 and DTW, whereas MMG dataset was tested with $N and $P.

The user-dependent results are shown in Figure 5.6. Synthetic samples were found to achieve very similar performance to that of human samples. This observation was consistent for all articulation

speeds and number of templates, using either the best and worst reconstructed human samples. Differences between human and synthetic samples (either worst (Synth$^-$) and best (Synth$^+$) case examples) were not statistically significant (two-tailed paired t-tests with Bonferroni correction, $p > 0.05/6$). It is interesting to note that \$1 and \$P perform quite well with just one loaded template; these recognizers are about as twice accurate as DTW and \$N, respectively. Then, when the number of templates increases, differences fall away. With all user's templates loaded, all recognizers are very accurate (greater than 99%).

User-independent test results are shown in Figure 5.7. As in user-dependent, we noticed that synthetic samples provide similar performance to that of human samples. This was also consistent for all articulation speeds and number of templates, using either the best (Synth$^+$) and worst (Synth$^-$) generation template. Differences between human and synthetic samples were not found to be statistically significant (two-tailed paired t-tests with Bonferroni correction, $p > 0.05/6$). Again, \$1 and \$P performed better than DTW and \$N with just one loaded template. With all user's templates loaded, all recognizers provide similarly competitive advantage.

The next experiment evaluates the use of synthetic samples on a large gesture vocabulary. Usually, when the number of gesture classes in a dataset increases, the accuracy of a recognizer tends to decrease. This is so because of potential collisions introduced by perceptually similar classes. Some examples of such potential collisions occur in this dataset with `i` and `j`; `o`, `O`, and `0`; `c` and `C`, etc.

To test this hypothesis, we used the chars74k dataset. In chars74k dataset 55 users provided 1 sample per gesture class, so we can follow a procedure akin user-dependent tests. Given that this dataset includes multistroke samples, we tested \$N and \$P. The results are shown in Figure 5.8.

While differences between human and synthetic samples were not statistically significant (two-tailed paired t-tests with Bonferroni correction, $p > 0.05/2$), the recognizers performed worse in comparison to the results they achieved on the MMG dataset, as predicted; see Figure 5.6. Overall, it was found that synthetic samples achieved bet-

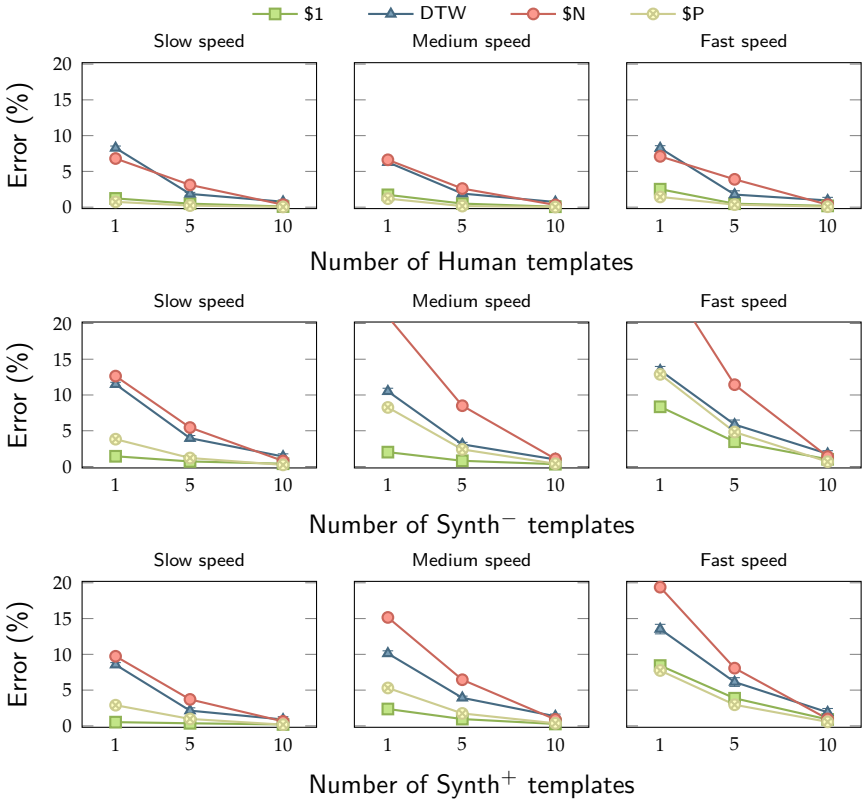


Figure 5.7. Impact of synthetic samples on articulation speed. User-independent tests. 95% confidence intervals are below 0.1%. Synth$^-$ and Synth$^+$ denote synthesized samples using the worst and best reconstructed human sample of each gesture, respectively.

ter results. This is true for samples synthesized from the best reconstructed samples as well as for samples synthesized from the worst reconstructed samples. For instance, with one loaded template, error rates surpassed 50% in case of human samples, while $N achieved 34.1% and 19.7% for the worst and best cases, whereas $P achieved 19.0% and 12.8%, respectively.

As usual, increasing the number of templates improved recognition accuracy. Although, this time the best improvements were achieved, by far, by the synthetic samples. Both $N and $P stabilized

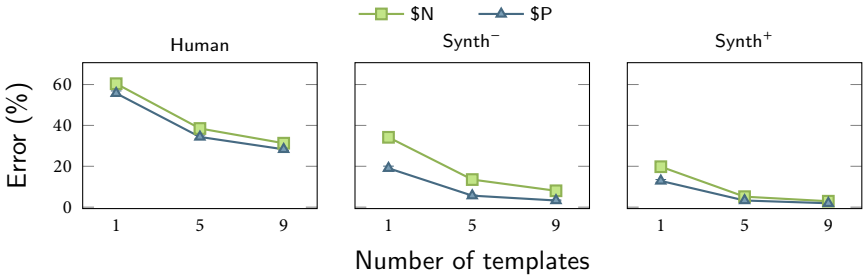


Figure 5.8. Impact of synthetic samples on a large gesture vocabulary (chars74k dataset, 62 classes). 95% confidence intervals are below 1%. Synth$^-$ and Synth$^+$ denote synthesized samples using the worst and best reconstructed human sample of each gesture, respectively.

around 30% error using up to 9 human samples as gesture templates, while achieving competitive results with synthetic samples, the error ranging between 8.0% ($N, worst case) and 1.9% ($P, best case). These results are encouraging and of potential interest for the design of gesture sets that use a large number of classes as part of their gesture vocabulary.

In addition, we wondered if there was any difference when users draw gestures with different input devices. Luckily, the MMG dataset allows us to test two conditions: finger and stylus. We conducted both user-dependent and user-independent tests. The 3 articulation speeds were averaged for these experiments. We followed the same procedure as in the previous experiments. First, user-dependent tests were performed.

The results are shown in Figure 5.9. This time, human samples performed better that their synthetic counterparts for 1 loaded template. This was found to be statistically significant for finger (two-tailed paired t-tests with Bonferroni correction, $p < 0.05/4$) but not for stylus with best case examples. Then, as soon as the number of templates increased, all conditions performed equally and statistically similar. With 9 templates, all recognizers achieved 99% of accuracy for both devices, using either human or synthetic samples.

Then, user-independent tests were performed using the same procedure as in the previous experiments. The results are shown in Fig-

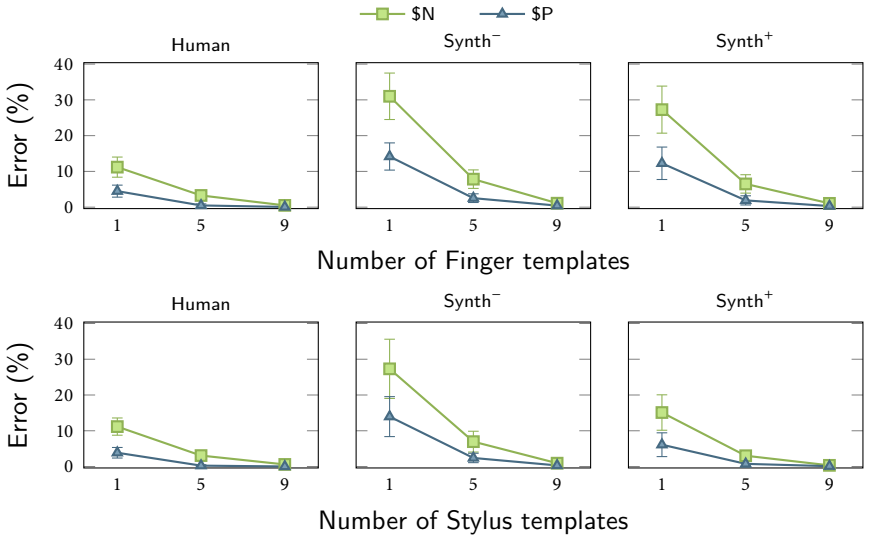


Figure 5.9. Impact of synthetic samples on input device. User-dependent tests. Error bars denote 95% confidence intervals. Synth$^-$ and Synth$^+$ denote synthesized samples using the worst and best reconstructed human sample of each gesture, respectively.

ure 5.10. We observed the same pattern as in the previous user-dependent experiments. With one loaded template, human samples achieved better accuracy. This was found to be statistically significant for both devices (two-tailed paired t-tests with Bonferroni correction, $p < 0.05/4$). With 5 templates per gesture, the best case samples performed equally similar to the original dataset samples. Then, with 9 templates loaded, all conditions performed equally and statistically similar. Further, all recognizers achieved more than 99% of accuracy for both devices, using 10 samples either of human or synthetic nature.

5.3 Discussion

According to the Kinematic Theory, the actual variability in handwriting articulation might come from two sources: the *action plan* of the user and the actual execution process. This is reflected by fluctuations in the control parameters (t_0, D, θ) and in the peripheral parameters

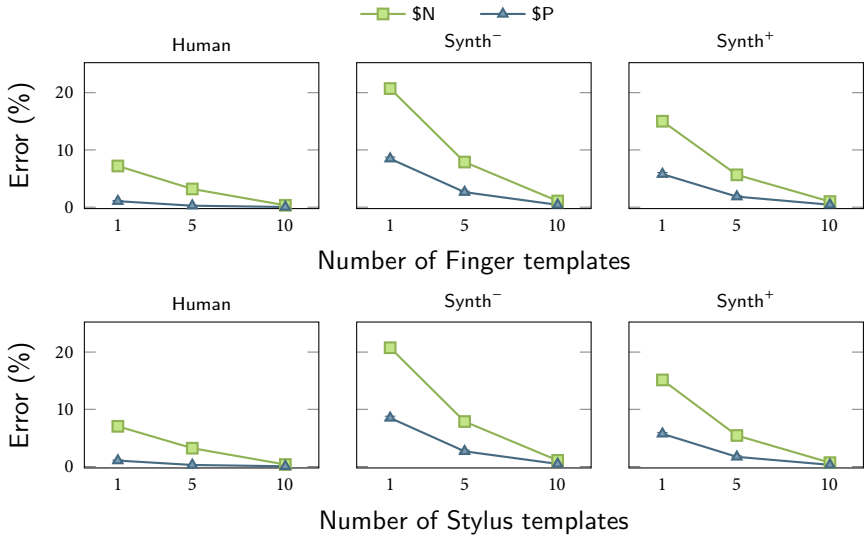


Figure 5.10. Impact of synthetic samples on input device. User-independent tests. 95% confidence intervals are below 1%. Synth$^-$ and Synth$^+$ denote synthesized samples using the worst and best reconstructed human sample of each gesture, respectively.

(μ, σ). The μ and σ variations mimic peripheral noise, like a writer who instantiates the same gesture intention and executes it with an upper limb slightly different from one trial to another. The t_0, D and θ variations refer to central fluctuations that might occur in the geometric and temporal position of the virtual targets of the action plan from one trial to another, reflecting for example attention changes. Combining both types of variations, the central and peripheral noise values being empirically tuned, reflects real-life situations like performing the same movement under different psychophysiological conditions.

The results of this chapter confirm previous works that report improved accuracy when training with a dataset that is extended with synthetic data [79, 80, 84]; e.g., from words or signatures collected from various writers sitting in front of a digitizer tablet, to sentences written on a whiteboard using full arm movements while standing up. In other words, while in practice there might be inherent differences between the generation of handwriting or gestures, it has been shown

that Sigma-Lognormal synthesized samples are actually reflective of how users produce them. Such a generalization has lead to postulate the underlying existence of a lognormality principle that guides human beings throughout their life, from the early steps of their motor learning processes to increasing departure from the ideal lognormal behavior, as the control of the fine motricity begins to decline with age and illness [77]. Finally, we should mention that Plamondon et al. have conducted several studies regarding human perception toward synthetic samples, showing showed that users cannot tell real and synthetic signatures apart [84].

Our work builds upon this fundamental notion, however we are the first to use the Sigma-Lognormal model to study finger writing behavior (see Section 5.2.2.3). This corroborates the prediction of the Kinematic Theory, where it is theorized that every human movement has a lognormal impulse response that results from the limiting behavior of a large number of interdependent neuromuscular networks.

5.4 Conclusion

This chapter has focused on synthetic stroke generation. Samples are synthesized using lognormal-based deformations on velocity profiles, which produces human-like results and ultimately helps a recognizer to perform well on unseen data. The experiments have been conducted using different types of recognizers. From template-matching recognizers, which typically achieve competitive accuracy with few training examples per gesture class, to HMMs, that is, recognizers that require a large number of training samples.

6 Applications

In this chapter we present 3 applications where the work of this thesis has been applied to. Section 6.1 presents Escritorie, a prototype of digital desk for interactive handwritten document analysis and text recognition.

Then, Section 6.2 presents "Gestures à Go Go", a web service, plus an accompanying web application, for bootstrapping gestures based on the work of Chapters 4 and 5.

Finally, Section 6.3 shows another example of an interactive application used within the pen computing paradigm. In this case, using the knowledge gained in Chapters 3 to 5, we study how translation reviewing can be done more ergonomically using a pen. This study was performed under the CASMACAT (Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation) project, funded by the Seventh Framework Programme for Research and Technological Development of the European Community.

6.1 Escritoire

Although digital data are increasingly more widely used, many documents are still on paper. Ideally, those documents should become accessible as a machine-readable text for searching, browsing, and editing. To bridge the gap between the *analog* and the digital, paper handwritten documents need to be captured [126], analyzed and transcribed.

We present Escritoire[1], a system that takes a step into this direction. The user works in a digital desk environment that combines the advantages of paper documents and the digital world, allowing an intuitive, natural and comfortable way to annotate, modify and work with both paper and digital documents in a seamless manner. This desk is continuously monitored using two cameras. The first one allows Escritoire to perform high-resolution scans. The second one is used by the built-in gesture recognizer. Escritoire automatically preprocesses the captured images, obtaining an adequate representation for the subsequent steps. Finally, if the document is handwritten, a multimodal interactive transcription process is carried out using a tablet where the user interacts with the comfort provided by a pen.

The major challenges in designing and implementing such system are: real-time performance, accurate detection of documents, reliable detection and interpretation of the user gestures, preprocessing and layout analysis of camera-based captured handwritten documents, interactive transcription and customized interfaces design.

Below we comment on prior works that bear direct relevance with the digital desk presented here.

6.1.1 Related Work

There is a huge body of research on capture of paper documents. Liang et al. [126] present a survey regarding the state-of-the-art on the document capture and detection. Lampert et al. [127] presents a prototype where capture is triggered by some pointing gestures and performed using a consumer camera. Unfortunately, this system only deals with printed documents, for which transcription can be carried out using OCR. User interface design is not an easy task, mainly because designers do not tend to follow any strict rule-based procedure. Several studies on user-friendly interfaces, have been carried out. Terry and Mynatt [128] presents a set of shortcomings in current user interfaces along with some guidelines. Given that design involves personal stylistic preferences, Eisenstein and Puerta [129] present a system that applies an adaptive algorithm to interface design. Many works have been carried out on layout analysis and text

[1] Video demo at: `https://www.prhlt.upv.es/showcase/htr/docs/video-HP.avi.mid.flv`

line segmentation. However, unsupervised segmentation quality does not reach the acceptable levels needed by end-user applications involving handwritten documents [130,131].

6.1.2 INTERACTING WITH ESCRITOIRE

Escritoire interface combines real and digital objects. On the one hand, there are physical objects, such as the different sheets of paper to be captured, the tablet to perform interactive transcription, etc. On the other hand, there are the different digital assets, such as the digital folders or the GUI elements (see Figure 6.1).

We will present the system by following Alice—a hypothetical user—while she uses the digital desk. Alice wants to transcribe a handwritten document. So she places the sheet in the capture zone (Figure 6.1a) and points with her index finger the *capture* button (Figure 6.1b). Escritoire captures and preprocesses the document, generating a better representation. Escritoire automatically detects that the document is handwritten and proceeds to transcribe it. After this, Alice decides to save the document for now. She aims with her index finger at the digital document and moves it into one of the available folders (Figure 6.1c).

Sometime later, Alice realizes that the system proposed transcription was not entirely correct. So she aims with her index finger at the folder to open it and extracts the saved transcribed document. Alice points at the Tablet button (Figure 6.1b) to send the document to the (physical) interactive transcription tablet (Figure 6.1d). When she is happy with the result, she clicks the *Return to Escritoire* button at the tablet interface and the document returns modified to the digital desktop.

6.1.3 SYSTEM IMPLEMENTATION

A system comprising a digital desktop can be devised in many different ways: a conventional screen providing interactions using a digitizer tablet; or a screen projected in a real desk, where interactions are provided using a digital pen; or directly work in a large desktop-sized multi-touch screen. Figure 6.2a shows the prototype that we have built. We decided to use a physical desk where the screen was

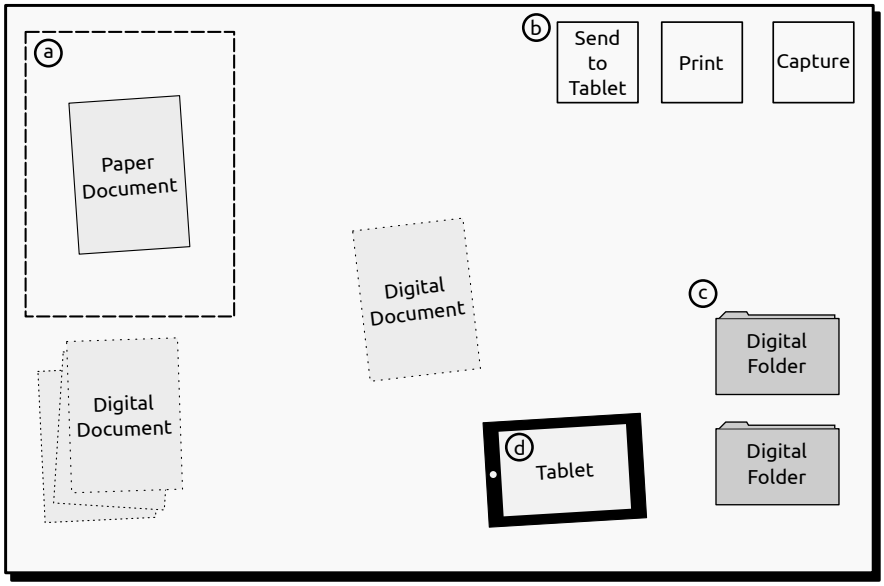
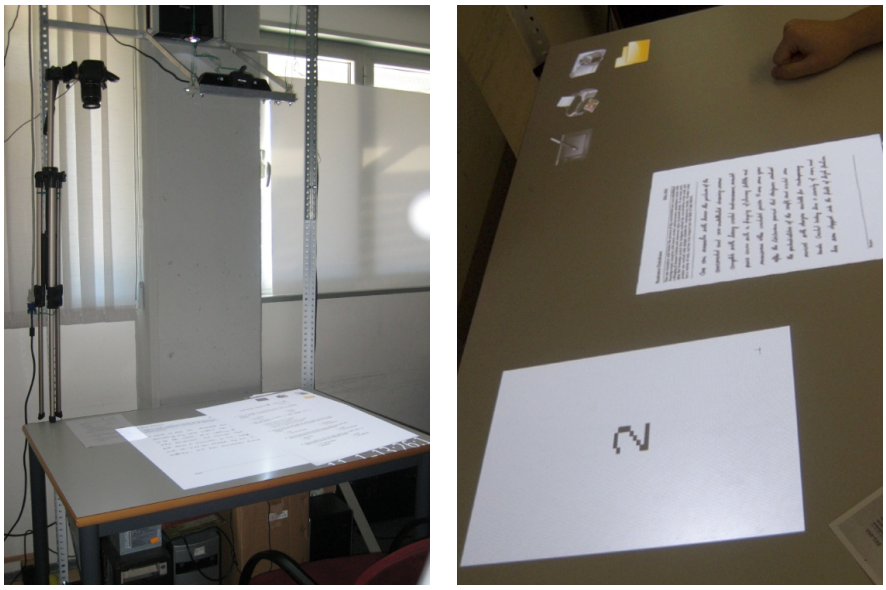


Figure 6.1. Escritoire user interface mock-up. (a) Capture zone where a real document is placed to be digitized. (b) Action buttons: The first button, labeled on the real interface with a tablet icon, allows the user to transfer a transcribed document to the tablet to improve the transcription. The second one, tagged with a printer icon, allows the user to print a copy of a digitized document. Finally, by pressing the button tagged with a camera and after a five-seconds countdown, any document located in the capture zone will be digitized. (c) Documents can be stacked or arranged into folders in Escritoire, similarly to traditional desktops. (d) A physical tablet used to perform interactive transcription.

projected. In our opinion, this was the simplest and cheapest way to create a prototype. Due to the distance between the desktop and the projector, a short throw projector (*InFocus IN1503*) was chosen, allowing us to display the proper image size. Two cameras were used: a fixed-location high-resolution camera (*Canon EOS 1100D*) is responsible for capturing documents. This camera is zoomed and focused on the *capture zone* (Figure 6.1a). This type of set-up provides us with a more robust configuration (same light conditions, less geometric distortions due to perspective, etc.), therefore simplifying the subsequent steps. A camera with a depth sensor (*Microsoft Kinect*) was used to detect the finger gestures. Finally, the interactive transcription is carried out using a *Lenovo Thinkpad Tablet 2* instead of directly on the



(a) **(b)**

Figure 6.2. (a) Escritoire prototype showing two digitized documents. (b) A sheet of paper near the capture zone, showing the five-seconds countdown.

desktop. This decision is based on previous experience. We realized that, due to the minimum font size and the projector resolution, trying to write directly on the desktop is cumbersome and inaccurate.

6.1.3.1 Gesture monitoring and detection

The Kinect of Escritoire monitors the work area searching for possible user gestures. To date, we use a simple set of gestures that are performed using one or the two index fingers. The current set of gestures includes:

Select: the system will select the item shown on Escritoire under the finger.

Move: after selecting an item, the user can translate it by just moving the hand around.

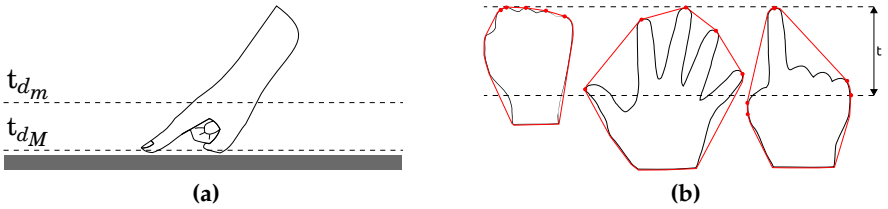


Figure 6.3. (a) Maximum and minimum threshold for defining a pixel of interest. t_{d_m} and t_{d_M} were empirically tunned. (b) Example (with $n = 4$) of convex hull for different hand positions.

Rotate: pointing both index fingers to a document and rotating hands.

Zoom: pointing both index fingers to a document and pinch open or close.

Since the set of gestures is performed with the index fingers, our system must be able to detect them. The first step in order to achieve this is to be able to differentiate the hands from the background. As the virtual desktop is displayed over the desk surface, we could not use any color-based technique (e.g., skin detection) to segment the hands.

Here we used the depth map provided by the Kinect, which captures depth information under any ambient light condition. From this depth image we need to isolate the pixels of interest (Figure 6.3a). We will calculate for every pixel contained in the depth image if they are a pixel of interest. Equation (6.1) provides the formal definition of pixel of interest ($\mathcal{S}_{ij}$). We compute the depth median value for every pixel ($\eta_{d_{ij}}$) during a period of 2 seconds. This way we can obtain a more stable value, minimizing the influence of outlayers derived from the sensor. Then, we calculate the difference between $\eta_{d_{ij}}$ with respect to the current depth value ($c_{d_{ij}}$). If this difference is within a minimum (t_{d_m}) and a maximum (t_{d_M}) threshold we can say that the current pixel is a pixel of interest.

$$\mathcal{S}_{ij} = \begin{cases} \text{True} & \text{if } t_{d_m} \leq \eta_{d_{ij}} - c_{d_{ij}} \leq t_{d_M} \\ \text{False} & \text{otherwise} \end{cases} \tag{6.1}$$

After segmenting the pixels of interest from the depth image, we want to know whether a group of pixels of interest are a hand. Previous to this, we apply a *closing* to the image, to remove any possible internal small hole and an *opening*, to remove any small noise object. As a simplification, we will assume that the biggest volumes, with a maximum of two, exceeding a certain area threshold, will be considered hands. This area threshold was empirically tunned to distinguish between noise and actual hands.

Once the hand—or hands—has been segmented from the background, the location of the index finger(s) must be found. We assumed that the user is always interacting with the system in front of the desk, thus the hands will always point *forward*. We also assume that the user hand has only 3 different states: pointing with the index finger, hand with the fingers clenched or flat.

Therefore, to distinguish between these cases we perform the following process. First, we compute the convex hull [132] of the contours that we consider hands. Then we apply the following technique: if the distance between the higher y-value vertex of the convex hull contour and the next n y-value vertex is for any case greater than a threshold d_t, we will say that the highest y-value point of this contour is the tip of an index finger (where n and d_t are parameters to be optimized). Otherwise, we assume that the user has the hand flat or with the fingers clenched. Figure 6.3b illustrates this process. Once we have found the tip of the index finger(s), a simple Nearest Neighbor tracking algorithm was applied to track their consecutive positions. After this, a Kalman filter [133] is applied to the tracked path(s) in order to reduce the noise.

Finally, depending on the number of fingers that the system has recognized and their position with respect to the desktop (located over a document, a button, etc), we can clearly identify the user gesture and react with the corresponding action.

6.1.3.2 Document capture and management

The document capture is carried out using a *Canon EF-S 18-55mm f/3.5-5.6 IS II* objective. The capture is performed when Alice selects the camera button. The system automatically will show the capture area,

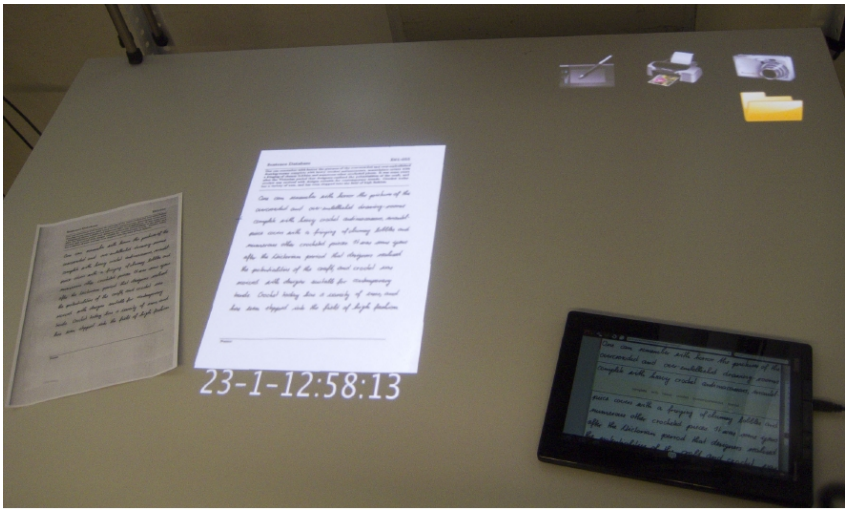


Figure 6.4. From left to right: a paper document, its digital version shown on Escritoire and tablet.

where the sheet must be placed in order to be captured. Figure 6.2b shows a captured document and the countdown. Alice can work with the document on Escritoire once it has been captured. As previously explained, there are several options: the document can be moved, rotated and zoomed. In addition, documents can be archived in folders.

6.1.3.3 Handwritten Text Recognition

Once the document has been captured, it is automatically transcribed. The HTR system employed here is the same as in Section 2.1.2. If Alice is not satisfied with the result, she moves the document to the tablet icon on the desktop and the document is transfered to the tablet to perform interactive transcription[2] (see Figure 6.4). In this tablet, she interacts with the system using handwritten text and stroke gestures—to perform certain actions such as: delete, insert, etc.

[2]Demo available at: `http://cat.prhlt.upv.es/iht/`

6.1.4 CONCLUSION

We have presented Escritoire, a pen computer prototype to manage documents, where digital and paper documents coexist. Currently, the system allows us to capture documents, organize documents or perform interactive transcription.

Right now, interaction with documents is integrated using finger gestures. However, these finger gestures are very simple. Further work regarding this topic is performed in the following section, where we present a web application to easily design gesture sets, by bootstrapping synthetic samples, that can be deployed with a ready-to-use recognizer.

6.2 Gestures à Go Go

Gestures are increasingly becoming a predominant input modality in today's user interfaces (UIs). Gesture interaction is possibly one of the most researched areas in Human-Computer Interaction (HCI), with a long history that started as early as 1960, with the Sketchpad project [134] and the RAND tablet [135].

Gestures can be mid-air (more prominent in gaming applications) or stroke based. We are particularly interested in the latter type, motivated by the fact that stroke gestures are becoming more and more relevant to mainstream products such as touchscreen-capable devices like smartphones and tablets (such as the Escritoire tablet).

Stroke gestures have existed in the industry for decades. Early examples of commercial products that successfully incorporated gestures are, e.g., PDAs like the Palm Pilot or the Apple Newton, and the Windows Tablet. These devices featured the Graffiti and Unistroke shorthand writing systems, which used a single stroke Roman letter-like gesture vocabulary. Today, stroke gestures are mostly used in consumer devices for executing simple actions, such as pinching a picture to zoom in/out, swiping to reveal an options menu, or panning to switch between apps. Nevertheless, stroke gestures are increasingly being incorporated to facilitate random access to smartphone contents, such as invoking a command hidden in an advanced settings menu or quickly searching for a friend's email in the contacts

list. Therefore, it is expected that stroke gestures will make a notable impact in consumers' lives.

In general, any application that is driven by gestures must rely on some recognition-based techniques. These techniques often require expert knowledge in pattern recognition or machine learning, something that is typically beyond the reach of many developers and UI designers. Furthermore, recruiting participants, data collection and labeling, etc. necessary for using these techniques are usually time-consuming and expensive. Thus, it is important to investigate how to empower developers to quickly collect gesture samples.

Here we present "Gestures à Go Go" (G3), a web service plus an accompanying web tool for bootstrapping stroke gesture samples based on the kinematic theory of rapid human movements [66,89]. The user only has to perform a gesture sample once, and G3 will generate from tens to thousands of synthetic human-like samples.

This aims for creating better gesture recognizers, eliminating the overhead of recruiting and data collection, and reducing the need for expert knowledge in machine learning. Together with the synthesized data, a number of gesture recognizers are available in different programming languages, thus allowing developers to create a competitive recognizer in a few clicks. As such, the outcome of G3 can be directly incorporated into production-ready applications.

6.2.1 Related Work

There is a huge body of research on gesture interaction. Over the past few years, new touchscreen-based products have taken off rapidly, boosting the popularity of stroke gestures as commands and symbols. An excellent integrative review of the state-of-the-art research on stroke gestures is provided by Zhai et al. [5]. Below we comment on prior works that bear direct relevance to this application.

6.2.1.1 Gesture Boostrapping

Training data is the key factor to build a competitive gesture recognizer. For instance, the Freehand Formula Entry System (FFES) suggests 20–40 examples per symbol per user [136]. Koch et al. [137]

studied gesture recognition using a Nintendo Bluetooth Wiimote controller as input, and found that 120 training patterns of accelerometer-based data are a lower bound; below that threshold the error rate increased dramatically.

A number of approaches are aimed at simplifying the process of designing gesture sets. An interesting example is Gesture Script [138], which allows developers to describe the structure of a gesture and its parts. With this information, Gesture Script synthesizes new gesture samples by changing the relative scale of each part and their rotation angles. Unfortunately, Gesture Script can only deal with unistroke gestures that are performed in a unique way. Besides, having to provide too detailed information for each gesture can be difficult and time-consuming for the user.

Gesture Marks [139] allows users to access applications and websites using gestures without having to define them first, by means of crowdsourcing and the combination of gesture and handwriting recognizers. Gestalt [140] supports the entire process of applying machine learning: implementing a classification pipeline, analyzing data as it moves through that pipeline, and easily transitioning between them. CrowdLearner [141] enables developers to quickly acquire a usable recognizer for their specific applications by spending a moderate amount of money (about $10) in a short period of time (about 2 hours).

Notable systems aimed at building gesture recognizers tailored to developers and end-users include MAGIC [142, 143] and Gesture Follower [144]. Both systems provide the user with a means of generating synthetic gesture samples in 3D space. MAGIC performs local perturbations to the resampled points of a gesture, whereas Gesture Follower introduce some variations to a gesture template using Viviani's curve formulation. These tools decrease the number of iterations needed to build a fast and stable gesture recognition interface, however there is no evidence that they can produce human-like samples. Further, these artificially generated samples usually perform poorly since they are not sufficiently variable for high-quality training [80]. However, these prior projects demonstrate the ongoing im-

portance of and interest in improving gesture recognition by acquiring large data samples.

6.2.1.2 Gesture Design Tools

Gesture design tools are well studied in the HCI community [145,146]. Example-based approaches like GRANDMA [22], GDT [147], Gesture Coder [148], or Gesture Studio [149] allow developers to create and test gestures by recording examples.

There are a number of similar systems tailoring end-users. For instance, EventHurdle [150] is a visual gesture-authoring tool to support designers' explorative prototyping. It supports remote gestures from a camera, handheld gestures with physical sensors, and touch gestures by utilizing a touchscreen. Also, designers can visually define and modify gestures through interaction workspace and graphical markup language with hurdles. A CAPpella [151] is another programming by demonstration environment intended for end-users. It allows users to *program* their desired behavior without writing any code, by demonstrating it to the system and by annotating the relevant portions of the demonstration. GestIT [152] allows declarative and compositional definition of gestures for different categories (e.g. multitouch and full-body gestures).

Other tools like iGesture [153] and InkKit [154] offer several algorithms through a high-level interface. However, they are more intended for recognition benchmarking. In this line, Beuvens and Vanderdonckt [155] devised a desktop application for facilitating the integration of gestures in UIs by describing the roles of the gesture specialist and other stakeholders involved in the development life cycle. G3 preserves these core interactions, though its goal goes further.

6.2.1.3 Gestures as A Service

To close this related work section, we should mention a number of previous works that offered gesture development over the Web. First, Seghbroeck et al. [156] described WS-Gesture, a framework that allows users to control different devices based on the DPWS (Devices Profile Web Services) standard. Although an important piece of the

framework, gesture recognition itself was not the topic of their research.

Second, Vatavu et al. [157] developed Gesture Profile for Web Services (GPWS), an event-driven architecture based on the service-oriented architecture (SOA) standard. GPWS focuses on delivering gesture recognition services, for which a user must provide the data required to train a recognizer.

Third, MAGIC 2.0 [158] is a web-based prototype that allows users to interface with the MAGIC framework [142]. Users can design motion gesture datasets while testing for and preventing false positives, which is important for sensor-based prototypes, as the user may accidentally trigger unintended gestures.[3] Thus MAGIC is related to gesture classification systems, instead of gesture bootstrapping. G3 fills this gap and optionally allows for quickly creating a ready-to-use, simple gesture recognizer that is suitable for use on prototypes in different programming languages.

6.2.2 G3 WEB SERVICE

The web service[4] was designed with simplicity and ease of use in mind. Therefore, a single URL is made available as endpoint, accepting: the number of desired synthetic samples (10 by default), the degree of gesture variability (ξ, 1.0 by default) and an example gesture.

For instance, the following HTTP request will return 15 synthetic samples having a relatively high variability degree:

```
POST /synthesize HTTP/1.1
Host: http://g3.service.url
Content-Type: application/json

{ "gesture": "{···}", "num_samples": 15, "variability": 0.75 }
```

where `"gesture"` points to a JSON-formatted string representing a stroke sequence:

```
{ "id1":[[x1,y1,t1]···[xM,yM,tM]], ···, "idN":[···] }
```

[3] This phenomenon is related to the well-known "Midas Touch problem" in eye-gaze interaction: the sensors cannot be used directly as a pointer device, because the sensors are always "on."

[4] More details at `https://g3.prhlt.upv.es/about/`

Each stroke is a tuple of 2D coordinates plus timestamp, and has an ID in order to make the format compatible with multitouch gestures.

We should mention that the ξ parameter is just meant for fine-tuning while operating the G3 interface: while 1.0 is a reasonable choice (it provides more diverse gesture exemplars), setting it to 0.0 is useful for research (e.g. inspect sample reconstruction, test and compare different noise generation techniques) or user adaptation purposes (replicating the same sample is equivalent to weighing its importance in a dataset).

On the other hand, if no timestamps are available, the web service accepts an optional third parameter to set a sampling frequency (200 Hz by default). For instance, assuming that in the previous example the coordinates do not have associated timestamps but we know they were acquired at 100 Hz, we can specify such sampling rate as follows:

```
POST /synthesize HTTP/1.1
Host: http://g3.service.url
Content-Type: application/json

{ "gesture": "{···}", "num_samples": 15,
  "variability": 0.75, "rate": 100 }
```

The web service returns a JSON-encoded string:

```
HTTP/1.1 200 OK
Connection: close

{
  "success": true, "error_code": null,
  "samples": [gesture01, ···, gesture15]
}
```

The `success` property informs about the bootstrapping result: `true` if success, `false` otherwise. Each synthesized gesture sample has the same number of strokes as the original gesture, and is resampled according to the user's articulation speed[5] Finally, JSONP requests are possible, in order to enable cross-domain communication.

[5]Estimated either from the timestamps or the optional **`rate`** parameter.

6.2.3 G3 WEB APPLICATION

The web service alone might not be of practical use for novice developers or UI designers, as they may not be familiar with machine learning techniques or gesture recognition algorithms. For that reason, we release a web-based application[6] interfacing with the web service that allows users to make the most of its potential. A detailed overview is given in the next section.

The web application incorporates a number of template-matching recognizers (see next section), so that it is possible to build a working prototype together with the synthesized data. Further, being open source, our application allows anyone interested to contribute to improving it. For instance, an envisioned task is that of including a particular gesture recognizer in a particular programming language. To do so, currently a developer must put the source code of said recognizer in a special folder inside the application's working directory together with a `g3manifest.json` file. The following is an example:

```
{
  "name": "NN-DTW", "description": "NN classifier with DTW",
  "author": "mlpy", "version": "3.4.0", "language": "Python",
  "website": "http://example.com/dtw/",
  "capabilities": ["unistrokes"],
  "preprocessor": "/path/to/file"
}
```

Here, the interesting part is the file indicated in the `preprocessor` property. This file takes as input a gesture set in JSON format and outputs a working gesture recognizer in a ZIP file. The preprocessor instructs our tool how to build the recognizer, e.g. feeding a template library with the new samples provided or even training a sophisticated classifier. Thus, our tool abstracts away the recognizer's implementation. The `capabilities` option provides additional information about the recognizer; e.g. whether it can deal with multistroke gestures or if it only targets unistroke gestures. This is used to inform the user prior selecting a combination of recognizer plus programming language (see Figure 6.5).

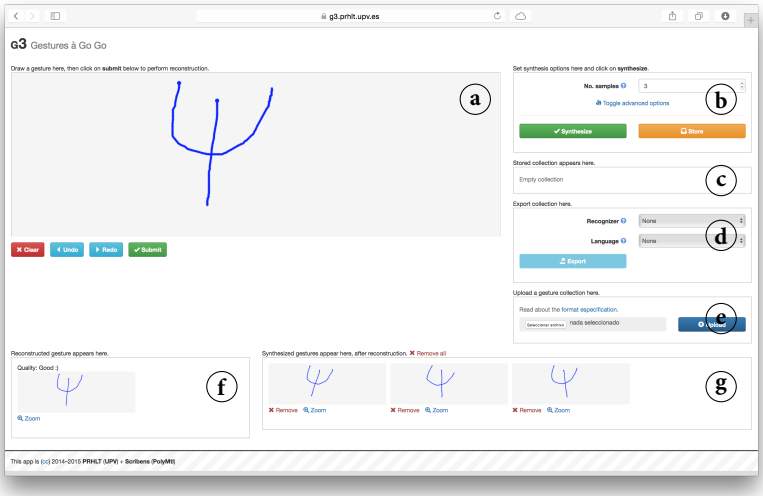


Figure 6.5. G3 user interface. **a**: Drawing area. The dot indicates the starting point of each gesture pen-stroke. **b**: Options area. Besides the number of gestures to generate, advanced options allow the user to indicate e.g. a desired variability degree. **c**: Collection area. Each gesture is presented as an ordered list, with the possibility of adding or removing gesture examples. **d**: Export area. The user can optionally export a gesture recognizer available in different programming languages. **e**: Import area. A JSON file comprising a collection of gesture examples can be submitted. **f**: Reconstruction area. A reconstruction of the user gesture is used for later synthesis. **g**: Synthetic gestures area. Generated samples appear here.

6.2.4 INTERACTING WITH G3

Like prior example-based systems, our web application allows developers to quickly create a gesture set plus a suitable gesture recognizer. However, within G3 interface the developer simply demonstrates one example of each gesture. The web application currently supports unistroke, multistroke, and multitouch gestures drawn on a 2D canvas. Below we introduce our tool by following a developer, Alice, as she creates a set of customized gestures. Alice needs to build a recognizer that handles 5 different gestures in her application.

[6]Demo at `https://g3.prhlt.upv.es/`

6.2.4.1 Example-based Demonstration of Gestures

For each gesture, Alice records just one example by drawing on the web canvas. We are not aware of any other tool that requires such a small user intervention at this time. Next, G3 verifies that the gesture data is of enough quality for later synthesis. It is considered so when $SNR_v \geq 15$ dB (see Section 4.1.4). When the SNR_v of the provided gesture example is below that threshold, G3 informs Alice so that she can draw the gesture again. The reasons why a gesture might not achieve a proper SNR_v include: too fast-paced execution (low number of captured points), under-resourced hardware (e.g. an old smartphone), or simply a symptom of possible problems in the user's motor control system [77].

6.2.4.2 Synthesizing Additional Examples

Once a gesture has been drawn, Alice chooses the desired number of samples that will be synthesized as well as the degree of variability for such synthetic samples. Next, she clicks on the *Submit* button and G3 displays a list with the synthetic gesture samples.

6.2.4.3 Iterative Refinement

Alice does not like 2 of the synthesized gestures, so she clicks on *Remove* icon attached to each gesture and requests 2 additional samples. When she is happy with the result, she clicks on the *Store* button and the samples of the current gesture are saved. The original hand-made gesture is also stored together with the synthetic samples.

6.2.4.4 Exporting Gesture Data

After gesture synthesis, Alice immediately has a gesture set of an arbitrary size (e.g., 20 samples for each of the 5 gestures provided, 100 samples overall). By clicking on the *Export* button, the data are exported as a JSON file. G3 will remember the generated gesture set, in case Alice wants to modify some of the gestures later.

Alice also decides to try a number of the recognizers provided by the G3 interface, so she selects the desired combination of recognizer

plus programming language and clicks on the *Export* button. As a result, Alice gets a recognizer together with the synthesized gesture set. G3 currently features the $ family and DTW in different programming languages (JavaScript, ActionScript, Python, C#, Java, C++, PHP), which are well-suited for experimentation by developers and UI designers. Excepting JavaScript, not all combinations of recognizer plus programming language are available at this time.

6.2.4.5 *Incorporating a Recognizer*

In case a recognizer is created within G3, incorporating it in the developer's application is analogous to prior works [138, 141, 148]. Alice will add the recognition module containing the implemented algorithms and the gesture files as created and exported from G3. We have decided not to incorporate recognizers that require extensive training because such approaches do not fit our desire of quickly providing users with a simple and ready-to-use gesture recognizer.

6.2.4.6 *Importing Gesture Data*

Alice remembers that some time ago she downloaded a small dataset consisting of 2 examples per gesture, 16 different gestures in total. Now she wants to generate a bigger dataset to train a custom recognizer, so she prepares a JSON file according to the following specification:

```
{
  "a gesture label": [gesture01, ···, gestureN],
  ···,
  "another gesture label": [gesture01, ···, gestureN]
}
```

where each gesture follows the format described in subsection 6.2.2. She then clicks on the *Upload* button and the collection module gets updated (Figure 6.5, D). Next, she goes through each imported gesture and requests the desired number of samples. As previously commented, if some of the uploaded samples were reconstructed with $\text{SNR}_v < 15$ dB, G3 interface will inform Alice. Finally, she clicks on the *Export* button and she gets delivered the synthesized dataset, and optionally an accompanying gesture recognizer.

6.2.5 DISCUSSION

The fundamental advantage of G3 over other approaches is that the user just needs to draw one example of each gesture. Then, samples are synthesized using lognormal-based deformations on velocity profiles, which produces human-like results and ultimately helps a recognizer to perform well on unseen data.

With G3, the user can request more or less diverse samples, according to the ξ parameter. However, we should mention that such parameter is just for fine-tuning. This is so because it has been shown that generally people tend to perform gestures consistently and it is hard to manually introduce variation [138]. That said, we believe that practitioners would use $\xi = 1.0$ most of the time, as it provides more diverse gesture exemplars, whereas researchers might want to experiment with other values.

In most interactive applications, a gesture is generally submitted as a stream of asynchronous timestamped events. Interestingly, because G3 has access to a model of the original human gesture, it can return reconstructed samples in a variety of ways, e.g., unevenly or uniformly distributed coordinates either in space or time. Right now G3 transforms the original stream into constant frequency; i.e., the resulting synthesized coordinates are uniformly distributed in time. This allows to *fix* downsampled pen-strokes that were acquired with under-resourced hardware; see e.g. the last 2 examples of human-made gestures in the MMG dataset (Figure 5.3). Ultimately, this approach may have a significant impact on the design of template-based gesture recognizers, since some preprocessing steps prior to recognition (e.g. resampling) could be omitted.

We should point out that a synthesized gesture has the same number of pen-strokes as the original gesture after reconstruction, because G3 uses only 1 example for bootstrapping, aimed at unburdening the user. While this is not a limitation for synthesizing unistroke gestures (for obvious reasons), one might want synthesized multistroke gestures to have a different number of pen-strokes than their human counterparts. If so, the user can simply provide another example of the same gesture executed with a different number of pen-strokes.

An actual limitation of our current implementation is related to temporal performance. While G3 actually does not mind generating either ten or thousand gesture samples from one given example, the payload resides in the reconstruction process, which is proportional to the number of points (pen-stroke-wise) and the number of pen-strokes. For instance, generating 1,000 samples of each gesture in the $1-GDS dataset took on average 68.7% of the processing time (SD=13.1%) in reconstructing each human sample. The same procedure took a bit more time in the MMG dataset (82.1% $\pm$ 16.5%) since there are multistroke samples. Results with the chars74k dataset stayed in between (76.1% $\pm$ 15.5%) since there are less multistroke samples than unistrokes.

6.2.6 CONCLUSION

G3 is an intuitive web application that offers active design exploration by prototyping gesture datasets and deploying them with ready-to-use recognizers. At the moment, G3 can only process 2D gestures. Thus, in future work we would like to extend it for processing 3D gestures, derived from e.g. accelerometers, Kinects, Wii controllers, and similar devices.

6.3 Interactive Translation Reviewing Using a Pen

This section shows another example of an interactive application used within the pen computing paradigm. In this case, we focus on how translation reviewing can be done more ergonomically using a pen. This work was developed under the CASMACAT project.

The objective of this project was to build the next generation translator's workbench to improve productivity, quality, and work practices in the translation industry. Different cognitive studies were carried out of actual translator behavior, based on key logging and eye tracking. The acquired data was examined for how interfaces with enriched information are used, to determine translator types and styles, and to build a cognitive model of the translation process.

Based on insights gained in these studies, novel types of assistance to human translators were developed. One of these was the use of a

	R01	R02	R03
Years of translator training	4	4	5
Years of professional experience	8	3	31
Reviewing experience	✓	✓	✓
Have you ever used a pen for reviewing?	✗	✗	✗

Table 6.1. Profile of the reviewers.

pen in a reviewing scenario, where the user is likely to introduce less changes. Previous works have shown that, from the user viewpoint, the use of a pen is better than a keyboard when there is pointing, writing or drawing involved [159–161].

The objective here was to study how to take advantage of the different contributions of this thesis to improve the performance of the reviewing reviewing system using a pen.

6.3.1 FIELD EVALUATION

The reviewers were sitting in front of a computer screen and had a WACOM tablet with a pen. Reviewers were supposed to amend translation mistakes using the pen using handwritten text and a set of gestures—deletion, insertion and undo—. Before starting the evaluation, the reviewers were briefly instructed on how to use the pen.

Two rounds of pen reviewing with 3 different reviewers were performed. Table 6.1 shows information about the 3 reviews. The first round was performed using an on-line HTR system similar to the one presented in Section 2.1.1, where interactions were provided at word level as in Section 3.1.3. Gesture recognition was performed using MinGestures [30]. HMMs were trained with the the IBM-UB-1 dataset, whereas the EMEA dataset [162] was used to train 2-gram language models, with a 5,000 words as vocabulary. In order to improve recognition rates, the system was *writer-adapted* using 100 different words prior to the experiment as shown in Section 5.2.2.2. On the other hand, the second round was performed employing a commercial handwritten text recognizer.

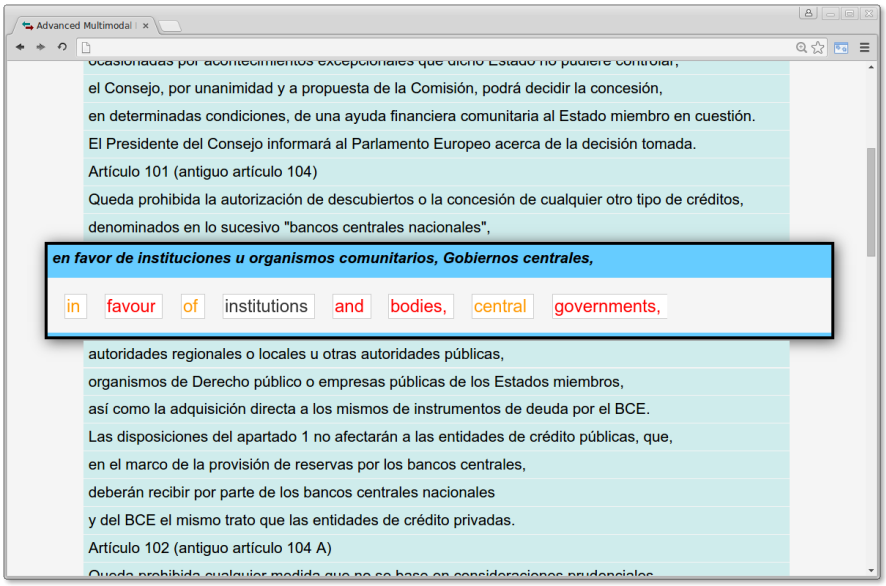


Figure 6.6. CASMACAT pen reviewing user interface.

Then, a baseline scenario was set, were the reviewers performed the corrections using a keyboard.

6.3.1.1 Results

Reviewers' feedback was collected through talk-aloud interviews. We started each interview with an informal conversation, where we asked each reviewer about their subjective perception of the pen recognition accuracy. In the first round of experiments, users estimated that pen recognition accuracy was about 50%, whereas the accuracy for the second round of experiments was 80% recognition. After this informal start, we continued collecting reviewers' feedback through these three basic questions:

- How would you evaluate revision using a pen: 1 *(very dissatisfying)* - 5 *(very satisfying)*?

- According to your own personal opinion, what are the advantages and disadvantages of using a pen for revision (as compared to using a keyboard)?
- How would you suggest to make the pen interaction more successful (more productive)?

Two out of the three reviewers in the study evaluated the use of a pen for reviewing purposes as 2 (dissatisfying), whereas the third rated it as 3 (neutral). When asked to enumerate possible advantages and disadvantages of reviewing using a pen, all participants mentioned recognition errors as the main disadvantage, together with the time spent to handwrite their corrections. R02 said: *"Despite 100% successful recognition using a pen, I think I will never be as fast as typing"*. As a way to make pen interaction more successful, R01 suggested to make the pen window bigger in order to have more freedom of movement. The same participant also suggested to increase the separation between words in the pen reviewing mode in order to make the gesture for insertion more accurate and less error-prone. R02 and R03 mentioned that user experience can be improved if the pen could be directly used on the screen instead of having to use an external tablet to handwrite while looking at the computer screen. R01, R02 and R03 perceived gestures as very useful. In the second round, the users noticed that the handwriting recognizer was more accurate. However, they did not change their opinion with respect the pen as a tool for reviewing.

The results of the different scenarios can be found in Figure 6.7. As we can see, reviewers spent more time in the first pen round than in the second. This is probably due to the fact that the recognizer committed more errors and, thus, reviewers had to spend more time retrying to introduce the correct text. Second, we can observe that, for reviewers R01 and R02, pen reviewing was just a bit slower than the baseline—using a keyboard—. Although we had expected that for small number of correction pen could be faster, it was also expected that handwriting would be slower than typing. In addition, in handwriting one must wait the for the server recognition response. In average, the server took 355ms to return the answer, whereas in 95% of the cases is was lower than 800ms, which could explain part

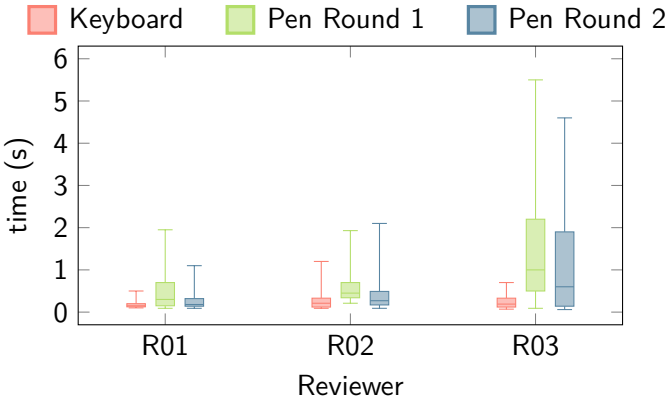


Figure 6.7. Time (in seconds) that reviewers spent on each segment normalized by source character. The time accounts for the total time spent in the segment from the moment it was opened until it was closed. Segments that were not modified are not considered in this plot.

of the difference between pen and keyboard. Also, we should note that pen only allows to work at word level at this moment. However, keyboard interactions can be at subword level. For example, we observed that `aripiprazole` was frequently replaced by `aripiprazol`. When using the keyboard, the correction consisted in just one mouse movement and click followed by a character deletion with the keyboard. Conversely, the whole word had to be written with the pen. Taking all that into account, the differences between keyboard and the second round of pen are rather small for R01 and R02. In contrast, R03 performed very bad with the pen. R03 was not only the reviewer with most experience (by large) but also the one who complained most about the pen system limitations. Thus, we deduce that for some reason R03 did not get used to that kind of interaction.

One of our assumptions was that working with the pen should allow for starting with the corrections earlier. Thus, for each segment we computed the time from the moment the segment was opened to the first interaction. The differences in the medians[7] indicate that R01 took 302 ms more with pen than with keyboard, whereas R02 and R03

[7]Medians values are reported, instead of the means, since they are more robust.

used 520ms and 3s more, respectively. As expected, R03 was the one who performed worst.

6.3.2 Laboratory Evaluation

Our on-line HTR system did not performed as well as expected, therefore we decided to create a dataset from the pen evaluation in the second round to run lab experiments that could identify the problems of our system. Thus, we analyze and compare here the performance of our recognizer and a commercial HTR system in order to close the gap observed during the evaluation. The pen-strokes captured in the second round of pen reviewing were manually annotated and thereby built into an experimental dataset. Then using this dataset, we report results of using the contributions of this thesis to in order to improve the pen recognition accuracy.

6.3.2.1 Experimental Setup

The target was to recognize 266 handwriting samples obtained from the second round, performed by 3 different reviewers. Three different setups were used, including the commercial one used in the second round.

The configuration of our system was the same as the one used in Section 6.3.1, except for the language model. Two different language models were tested in these experiments. LM1 is a closed 2-gram language model, consisting of 5,080 words, which included the 5,000 most frequent words in training plus 80 unseen test words. The purpose of this language model is to make sure that all the words which are going to appear in the test set—all the words which the writers actually wrote—have a chance of being recognized. The 2-gram language model was trained taking into account the frequency of occurrence of the words which appear in training and it is uniform for the remaining words. On the other hand, LM2 is an open vocabulary 2-gram language model, which included the 2,000 most frequent words in training and the 5,000 most probably words from the source sentences, altogether 6,072 words. The rate of running out of vocabulary words is 15.4%. Therefore, this is the minimum error which could ever be achieved using a word-based recognizer.

System	Error
Commercial	14%
Commercial (*)	12%
LM1	
Initial Result	22%
Writer Adaptation	15%
Writer Adaptation (*)	11%
LM2	
Writer Adaptation	33%
Writer Adaptation (*)	26%

Table 6.2. E-pen handwritten text recognition results. (*) indicates case insensitive.

6.3.2.2 Results

Results are shown in Table 6.2. In this case, results are better and the improvement with the writer-adapted system is bigger. On the other hand, since the recognizer is word based, in the open vocabulary experiments there is a minimum possible error of 15.4% because of the 80 test words which can never be recognized. Therefore, results should be considered better than they appear. In fact it is expected that many of the out-of-vocabulary errors can be recovered by using a word-based recognizer which can also work at the character level as a back-off method.

6.3.3 DISCUSSION

The evaluation of the pen as a reviewing tool has provided us with very valuable information. First, we have discovered that pen interactions for reviewing needs a specific user interface. Probably, a pen on a tactile screen would have been a much better choice since users can directly place the pen where they have their sight fixated. This fact was already acknowledged by reviews in the reviewers' feedback. Previously to the evaluation we had argued whether this could be a problem or not, and we had arguments in favor and against both, tablet and digitizer pen. Indeed, we had performed an early test with a Lenovo tablet with pen technology. However, that suggested that

the CASMACAT interface was too resource hungry for that kind of devices. Thus, a digitizer tablet with a desktop computer was our only choice. It would be interesting to see if their views become more positive after some more hours of interaction with the pen and changing the setup to a tactile screen.

Secondly, reviewers have shown their preference for a full-screen writing experience, as sometimes they found themselves writing on the border of the pen area. On the other hand, font sizes should be bigger and word separation wider so as to allow the user to be more precise when issuing gestures. Not only that, but the GUI should be redesigned to fit reviewers' needs. Furthermore, the HTR recognition accuracy is very important, given that is the main factor why productivity suffers when compared to the keyboard. Users are typically willing to accept error rates up to about 3% before deeming the technology as too encumbering [163]. On the positive side, gestures were perceived as very useful., suggesting that a mixture of gestures and keyboard could be a good choice.

6.3.4 CONCLUSION

We have tackled the problem of the reviewer under two perspectives. First, we have studied the use of the pen to correct translator's mistakes. Second, we have analyzed the reviewers' corrections in order to find patterns that could be performed automatically.

7 Conclusions

The work presented in this thesis has focused on improving the pen computer recognition accuracy. A pen computer is computer that allows a user to interaction with it by using a pen or touch-based interactions. This interaction methods are the most precise and comfortable, as well as the most flexible. However, pen and touch interactions require a recognition step to decode their non-deterministic meaning. In this thesis we have focused first on investigating how to take advantage of interaction-derived information to improve the accuracy of the pen computer recognizer, particularly in the context of interactive handwritten text recognition. Then, we have focused on the study of human movements—in particular, hand movements—from a generation point of view using the kinematic theory of rapid human movements. Understanding how the human body generates movements is important in the development of a recognition system, particularly to understand the origin their variability. Finally, we have investigated the benefits of using synthetic data to enlarge the training data.

7.1 Scientific Contributions

The different contributions of this thesis have been materialized in publications. As a result, 2 journals (plus 1 more submitted) and 5 conference papers have been generated. Besides these publications, an additional journal publication has been published with content closely related to this thesis. Below we sum up the different scientific contributions.

7.1.1 Interactive Handwritten Text Transcription

Previously, multimodal interaction had been studied only at whole-word level. However, character-level pen-stroke interactions may lead to more ergonomic and friendly interfaces. Empirical tests show that this approach can save significant amounts of user effort with respect to both fully manual transcription and non-interactive post-editing correction. Multimodal interaction at character-level was studied in:

- Daniel Martín-Albo, Veróncia Romero, Alejandro H. Toselli and, Enrique Vidal. Character-level interaction in multimodal computer-assisted transcription of text images. *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2011, pp. 684-691.

Previous work was extended with fine-grained multimodal interactions that allow taking advantage of interaction-derived context to signicantly improve feedback decoding accuracy. This was presented in:

- Daniel Martín-Albo, Veróncia Romero, Alejandro H. Toselli and, Enrique Vidal. Multimodal Computer-Assisted Transcription of Text Images at Character-Level Interaction. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 2012.

Finally, a more natural way of interaction, using pen-strokes forming character sequences was presented in:

- Daniel Martín-Albo, Verónica Romero, and Enrique Vidal. Interactive Off-line Handwritten Text Transcription Using On-line Handwritten Text as Feedback. *International Conference on Document Analysis and Recognitio (ICDAR)*, 2013.

The pen computer prototype *Escritoire* was published in:

- Daniel Martín-Albo, Verónica Romero, and Enrique Vidal. Escritoire: A Multi-touch Desk with e-Pen Input for Capture, Management and Multimodal Interactive Transcription of Handwrit-

ten Documents. *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2015, pp. 471-478.

7.1.2 HUMAN MOVEMENT MODELING

A new approach to better extract and estimate the lognormal primitives and parameters was developed. Through a comprehensive evaluation using 32,000 words from a public database, we show that our approach greatly improves the state-of-the-art extractor. This work was presented in:

- Daniel Martín-Albo, Réjean Plamondon and, Enrique Vidal. Improving Sigma-Lognormal Parameter Extraction. *International Conference on Document Analysis and Recognitio (ICDAR)*, 2015.

Web users often have a specific goal in mind comprising various stages that are reflected, as executed, by their mouse cursor movements. Therefore, is it possible to detect automatically which parts of those movements bear any intent and discard the parts that have no intent? We analyzed more than 10,000 browsing sessions comprising about 5 million of data points, and compared different segmentation techniques to detect discrete cursor chunks that were then reconstructed with the Sigma-Lognormal model. Our main contribution is thus a novel methodology to automatically tell chunks with and without intention apart. We also contribute with kinematic compression, a novel application to compress mouse cursor data while preserving most of the original information. This work was published in:

- Daniel Martín-Albo, Luis A Leiva, Jeff Huang, Réjean Plamondon. Strokes of insight: User intent detection and kinematic compression of mouse cursor trails. *Information Processing & Management*, 2016.

7.1.3 SYNTHESIZING PEN & TOUCH ON-LINE STROKES

We investigate the effect of adding synthetic samples in a multi-writer training dataset in:

A method for the automatic generation of synthetic handwriting was developed. To generate a new synthetic sample, first a real word is modelled using the Sigma- lognormal model. Then the Sigma-lognormal parameters are randomly perturbed within a range, introducing human-like variations in the sample. The experimental results confirm the great potential of the Kinematic Theory applied to writer adaptation. This work was presented in:

- Daniel Martín-Albo, Réjean Plamondon and, Enrique Vidal. Training of On-line Handwriting Text Recognizers with Synthetic Text Generated Using the Kinematic Theory of Rapid Human Movements. *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

Training a high-quality gesture recognizer requires providing a large number of examples to enable good performance on unseen, future data. However, recruiting participants, data collection, and labeling, etc., necessary for achieving this goal are usually time consuming and expensive. Thus, it is important to investigate how to empower developers to quickly collect gesture samples for improving UI usage and user experience. In response to this need, we introduce Gestures à Go Go (g3), a web service plus an accompanying web application for bootstrapping stroke gesture samples based on the Kinematic Theory. This work was presented in:

- Luis A. Leiva, Daniel Martín-Albo, and Réjean Plamondon. Gestures à Go Go: Authoring Synthetic Human-like Stroke Gestures Using the Kinematic Theory of Rapid Movements. *ACM Transactions on Intelligent Systems and Technology*, 2015.

Finally, we quantify the variability of synthetic gestures in:

- Daniel Martín-Albo, Luis A. Leiva, and Réjean Plamondon. Does the Kinematic Theory of Rapid Human Movements Produce Actual Human-like Stroke Gestures? *Interacting with Computers*, 2016. In revision.

7.2 Future Work

Finally, we identify the future research directions we intend to explore to extend the work presented in this thesis.

With respect to interactive transcription, there is still room for improvement. So far, if the feedback is provided by means of pen-strokes, the recognition is carried out using a two-step approach as seen in Section 3.1.3. Pen-strokes are decoded into the most probable sequence of characters and then this information is used to form a prefix, which is used by the off-line HTR system to provide a better transcript. However, this approach is too simplistic given that this approach can be formulated in only one step, where both the main and the feedback data stream help each-other to optimize overall accuracy. For example, Confusion Networks have been previously studied for the combination of handwriting and speech recognition with good results [164, 165]. Results show that several errors can be corrected by combining the off-line on-line HTR systems outputs, reducing the required human effort, and allowing to speed up the transcriber task.

The Kinematic Theory provides a well-established and solid framework for the study of the production of human movements and in previous works it has been shown that a computer mouse is reliable enough to be considered as a velocity acquisition device [166]. Yet to date, the Sigma-Lognormal model has not been used to study mouse cursor movements on websites. There has never been any study into treating cursor trails as the result of complex human motor control behaviors. After all, given that a mouse cursor movement is derived from a human movement, it makes sense to use the Sigma-Lognormal model to study user intent on websites from mouse cursor trails.

Moreover, hand-held touch-capable devices have become one of the most popular and fastest growing consumer products. It seems logical therefore to think of such devices as Personal Digital Bodyguards (PDBs), in charge for example of biometrical, biomedical, and neurocognitive (active) monitoring. However, the realization of this vision is a difficult challenge. Indeed, handwriting entails complex neuromotor skills. Producing a handwritten message requires the performance of numerous cognitive tasks leading to the production of words from the motor action plans that have been learned over the

years. According to the Kinematic Theory, these plans activate specific neuromuscular networks to produce a given pen tip trajectory by combining lognormal strokes, the fundamental units of handwriting movements, and superimposing them in time.

Most of the research regarding this theory has been done in well-controlled protocols and experimental setups, using standard digitizers characterized by their stable sampling frequency and high spatial resolution. One practical question that emerges when it comes to making a technology transfer toward hand-held devices (e.g. tablets, phablets, smart-phones...) is the following. Is today's hardware ready for such a move?

Finally, at the moment, the Sigma-Lognormal parameters extractor can only process 2D trajectories. Thus, in future work we would like to extend it for processing 3D gestures, derived from e.g. accelerometers, Kinects, Wii controllers, and similar devices. Actually, this kind of data can be reconstructed using lognormals provided that the torsion is taken into account [103], which just requires introducing an additional parameter to the Sigma-Lognormal model.

A *Evaluating Synthetic Gestures Human Likeness*

Training a high-quality gesture recognizer requires providing a large number of examples to enable good performance on unseen, future data. However, recruiting participants, data collection and labeling, etc. necessary for achieving this goal are usually time-consuming and expensive. In order to address this issue, previous works have proposed a technique to synthesize strokes [80, 83, 104] based on the Kinematic Theory and its associated Sigma-Lognormal model.

However, researchers have evaluated this technique from the perspective of classification performance and not from the perspective of how "human-like" the synthesized gestures really are. A notable exception is a study by Galbally et al. [84] that examined the human likeness of synthetic handwritten signatures. While it was found a high degree of similarity between synthesized and human signatures, it is unclear whether this will hold for stroke gestures. In sum, a dedicated evaluation has been pending for too long.

In this appendix, we prove that the Sigma-Lognormal model produces synthetic stroke gestures that hold similar characteristics as human-generated gestures by computing relative measures comparing geometric, kinematic, and articulation aspects using GREAT [167] (Gesture RElative Accuracy Toolkit).

A.1 Gesture Relative Accuracy Measures

This toolkit evaluates the deviation of each sample gesture with respect to a gesture task axis. The gesture task axis is a fixed example gesture, reflective of relative differences between individual executions, that serves as a reference against which the measures are

computed. Here we chose the *k*-medoid of each gesture class as task axis, as it is less sensitive to alignment errors (see Figure A.1). The *k*-medoid can be defined as the closest user-articulated sample to the median gesture[1]. Second, unistroke gestures were aligned in their chronological order of input; whilst multistroke gestures were aligned using the point-cloud matching procedure [168], which is invariant to the number of strokes and stroke ordering. Prior to alignment, gestures were resampled to 32 points and centered at the origin.

Geometric accuracy measures captures tendencies of the users to stretch and bend strokes during articulation. They evaluate the deviation of each sample gesture in terms of:

Shape Error represents the average absolute deviation of the candidate gesture points from the task axis in terms of the Euclidean distance.

Shape Variability computes the standard deviation of the distances between the points of the candidate and the task axis.

Length Error measures the user tendency to *stretch* pen-strokes with respect to the task axis.

Size Error captures the user tendency to stretch pen-strokes in terms of the area size.

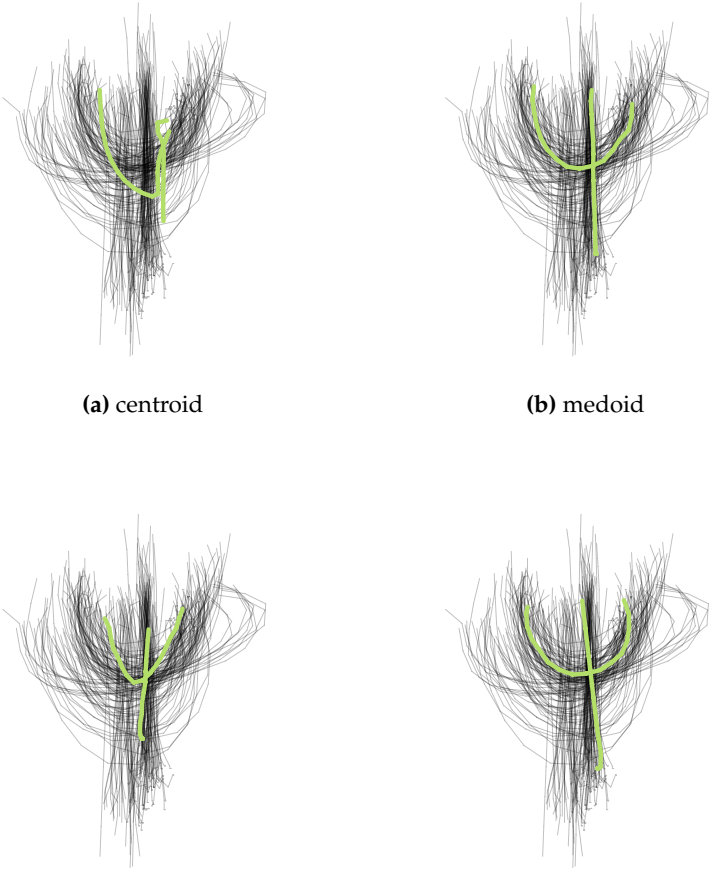
Bending Error measures the user tendency to *bend* the pen-strokes of the articulated handwriting with respect to the gesture task axis.

Bending Variability computes the standard deviation of the differences in turning angle.

The kinematic accuracy measures evaluate articulation differences in the time domain, and capture how fluent or smooth the handwriting is in terms of:

Time Error measures the duration difference between the candidate and the task axis.

[1]We modified GREAT to compute the task axes shown in Figure A.1.



(a) centroid **(b)** medoid

(c) *k*-centroid **(d)** *k*-medoid

Figure A.1. Different task axes (green lines) of a multistroke gesture from the MMG dataset. Being more robust to noise and outliers, the *k*-medoid was chosen to compute all relative measures.

Time Variability represents the standard deviation of the differences between timestamps measured at each individual point on the gesture path.

Speed Error measures the difference in the velocity magnitude profiles of the candidate and the gesture task axis.

Speed Variability represents the standard deviation of the local differences between the velocity magnitude profiles.

Finally, articulation accuracy measures how consistent users are in producing the individual pen-strokes of gestures. The articulation accuracy measures are:

Stroke Count Error reports the difference in the number of strokes between the candidate and the task axis.

Stroke Ordering Error is an indicator of the stroke ordering accuracy. That is, if the candidate gesture has been articulated in the same way as the gesture task axis, the ordering error will be low.

A.2 Evaluation

In order to address the fundamental question regarding human likeness, we replicated 2 public datasets: $1-GDS and MMG datasets (see Section 2.3.1).

A.2.1 Method

Each gesture sample in both datasets was modeled according to Equation (4.4) using the Sigma-Lognormal parameters extractor presented in Section 4.2. Gesture primitives were then distorted according to Equation (5.2), using $n_\mu = n_\sigma = 0.1$, $t_0 = 0.005$, $n_D = 0.15$, $n_{\theta_s} = n_{\theta_e} = 0.06$, the same values used by Galbally et al. [84]. Finally, the Cartesian coordinates (x,y) were retrieved using Equation (4.7).

Next, we used GREAT to compute geometric, kinematic, and articulation features of synthetic and human stroke gestures production.

A.2.2 RESULTS

To begin, we analyzed all input speeds together. We used unpaired two-sample *t*-tests (two-tailed, Bonferroni corrected) both for user-independent and user-dependent tests. In user-independent tests, all measures are computed considering each gesture sample as an independent observation, whereas user-dependent tests are computed for each user and then results are aggregated.

Figures A.2 and A.3 summarize the results. As can be observed, we did not find statistically significant differences between synthesized and human-generated gestures in most cases. Interestingly, significant differences in Speed Error/Variability in the MMG dataset appeared because mostly half of the timestamps in the human samples are duplicated ($M = 1.7$, $SD = 0.3$) possibly due to being acquired with "higher-than millisecond" precision.

However, how important are these results? When examining large samples, significance testing can be misleading because even small differences are likely to produce a statistically significant result. Indeed, as shown in the results, differences between populations are rather small; see e.g., Shape Variability: 5.5 *vs.* 6.9 px (GDS, user-independent) or Speed Error: 1.7 *vs.* 0.6 px/ms (MMG, user-dependent). Furthermore, the observed effect sizes in all cases suggest low practical significance (Cohen's $d < 0.2$, ($M = 0.13$, $SD = 0.1$). This was true both for user-dependent and user-independent tests. It should be noted that effect sizes have consequential validity, as they describe the magnitude of the differences.

Next, we split each dataset in terms of input speed and repeated the same analysis. Again, we did not find statistically significant differences in most cases (not discussed for brevity's sake) with similarly low effect sizes. We also analyzed the MMG dataset in terms of input device (stylus vs. finger) and observed the same outcome. Therefore, we conclude that there is no practical difference in gesture production between human and synthesized samples and that they "look and feel" the same. In other words, synthesized gestures are actually reflective of how users produce stroke gestures.

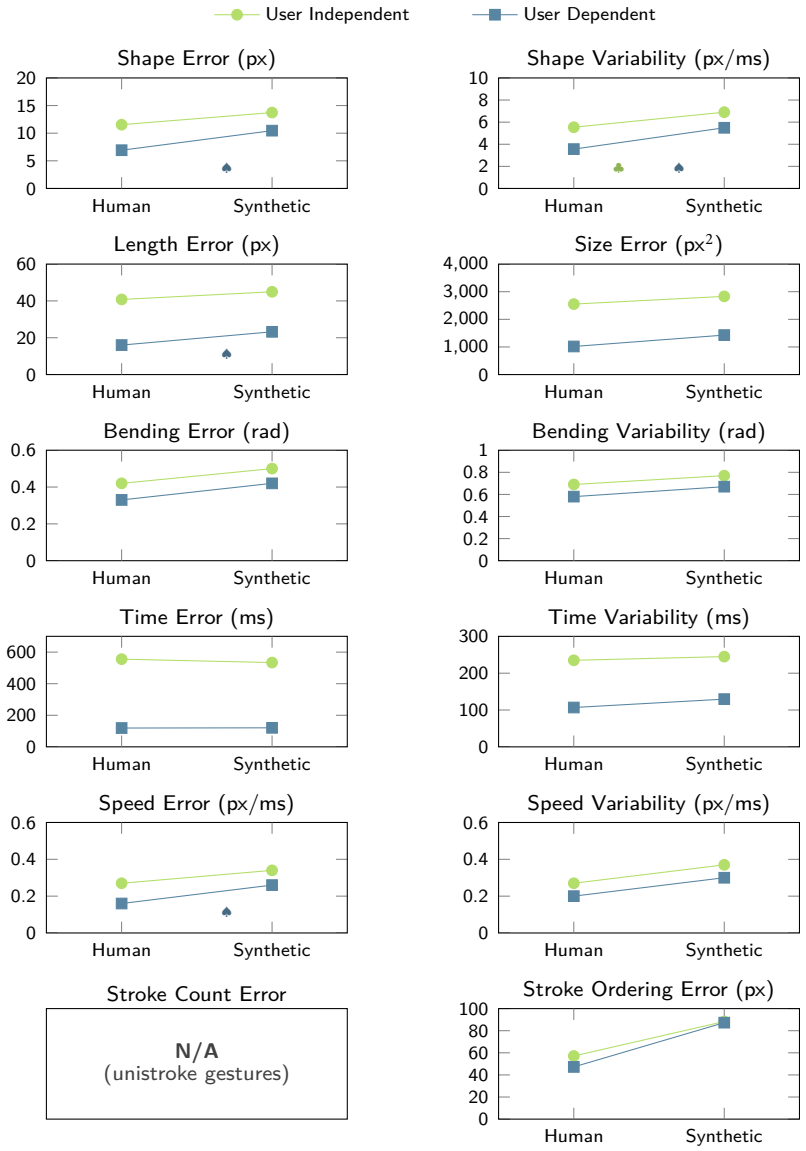


Figure A.2. Overall results with human and synthesized gestures for GDS dataset. Green circles denote user-independent tests and blue squares denote user-dependent tests. 95% confidence intervals are all below 1%, so they are omitted. A statistically significant difference between populations ($p < .05/12$) is denoted with green clubs for user-independent tests and blue for spades user-dependent tests.

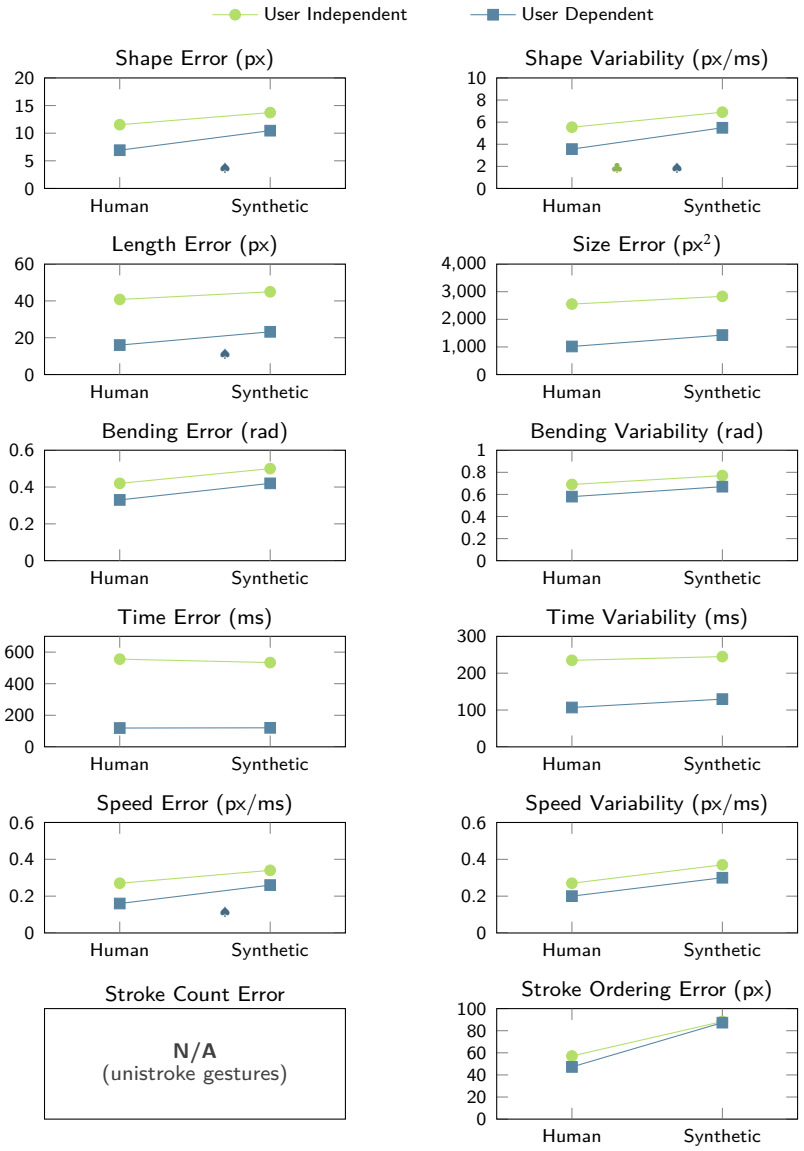


Figure A.3. Overall results with human and synthesized gestures for MMG dataset. Green circles denote user-independent tests and blue squares denote user-dependent tests. 95% confidence intervals are all below 1%, so they are omitted. A statistically significant difference between populations ($p < .05/12$) is denoted with green clubs for user-independent tests and blue spades user-dependent tests.

A.3 Conclusion

We fail to reject the null hypothesis that there is difference between human and synthesized gestures. Taking into account the large number of observations involved in each comparison, we conclude that synthesized stroke gestures are articulated very similar to their human counterparts. Therefore, there is no practical difference in gesture production between human and synthesized samples and that they *look and feel* the same. In other words,the kinematic theory of rapid human movements produces actual synthetic human-like stroke gestures. It is thus reliable to generate synthetic datasets this way, since the overall performance and behavior of gesture samples will be consistently similar to that of actual users.

List of Figures

List of Tables

Bibliography

[1] Fortune, *The new computer revolution. cover story*, June, 1993.

[2] M. Igbaria, J. Iivari, and H. Maragahh, *Why do individuals use computer technology? A finnish case study,* Information & management **29** (1995), no. 5, 227–238.

[3] A. Meyer, *Pen computing: a technology overview and a vision,* ACM SIgCHI Bulletin **27** (1995), no. 3, 46–90.

[4] R. Plamondon and S. N. Srihari, *On-line and off-line handwriting recognition: A comprehensive survey,* IEEE Trans. on Pattern Analysis and Machine Intelligence **22** (2000), no. 1,.

[5] S. Zhai, P. O. Kristensson, C. Appert, T. H. Andersen, and X. Cao, *Foundational issues in touch-screen stroke gesture design-an integrative review,* Foundations and Trends in Human-Computer Interaction **5** (2012), no. 2, 97–205.

[6] E. Vidal, L. Rodríguez, F. Casacuberta, and I. García-Varea, *Interactive pattern recognition*, in *Proc. MLMI*, pp. 60–71, 2007.

[7] A. H. Toselli, E. Vidal, and F. Casacuberta, *Multimodal interactive pattern recognition and applications*. Springer Science & Business Media, 2011.

[8] R. Plamondon, A. M. Alimi, P. Yergeau, and F. Leclerc, *Modelling velocity profiles of rapid movements: a comparative study,* Biol. Cybern. **69** (1993), no. 2,.

[9] E. Anson, *The device model of interaction,* SIGGRAPH Comput. Graph. **16** (1982), no. 3, 107–114.

[10] S. Jaeger, S. Manke, J. Reichert, and A. Waibel, *Online handwriting recognition: the npen++ recognizer,* Int. Journal on Document Analysis and Recognition **3** (2001), no. 3,.

[11] M. Pastor, A. Toselli, and E. Vidal, *Writing speed normalization for on-line handwritten text recognition*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 1131–1135, 2005.

[12] A. H. Toselli, M. Pastor, and E. Vidal, *On-line handwriting recognition system for tamil handwritten characters*, in *Proc. Iberian Conference on Pattern Recognition and Image Analysis*, pp. 370–377, 2007.

[13] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, *A novel connectionist system for unconstrained handwriting recognition,* Pattern Analysis and Machine Intelligence, IEEE Transactions on **31** (2009), no. 5, 855–868.

[14] F. Jelinek, *Statistical Methods for Speech Recognition*. MIT Press, 1998.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[16] L. E. Baum, *An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of a Markov Process,* Inequalities **3** (1972).

[17] A. H. Toselli, A. Juan, J. González, I. Salvador, E. Vidal, F. Casacuberta, D. Keysers, and H. Ney, *Integrated handwriting recognition and interapretation using finite-state models,* Int. Journal of Pattern Recognition and Artificial Intelligence **18** (2004), no. 04, 519–539.

[18] E. Kavallieratou and E. Stamatatos, *Improving the quality of degraded document images*, in *Proc. Int. Conf. Document Image Analysis for Libraries*, 2006.

[19] M. Pastor, *Aportaciones al reconocimiento automático de texto manuscrito*. PhD thesis, Universitat Poliécnica de Valéncia, 2007.

[20] V. Romero, M. Pastor, A. Toselli, and E. Vidal, *Criteria for handwritten off-line text size normalization,* Proc. VIIP **6** (2006).

[21] A. Butter and D. Pogue, *Piloting Palm: The Inside Story of Palm, Handspring, and the Birth of the Billion-Dollar Handheld Industry*. John Wiley & Sons, Inc., 1st ed., 2002.

[22] D. Rubine, *Specifying gestures by example,* Proc. annual Conf. on Computer graphics and interactive techniques (SIGGRAPH) **25** (1991), no. 4, 329–337.

[23] S. D. Connell and A. K. Jain, *Template-based on-line character recognition,* Pattern Recognition **34** (2000), no. 1, 1–14.

[24] A. Belaid and J. Haton, *A syntactic approach for handwritten formula recognition,* IEEE Trans. on Pattern Analysis and Machine Intelligence **6** (1984), no. 1, 105–111.

[25] R. Marzinkewitsch, *Operating computer algebra systems by hand-printed input*, in *Proc. ISSAC*, pp. 411–413, 1991.

[26] M. Koschinski, H. J. Winkler, and M. Lang, *Segmentation and recognition of symbols within handwritten mathematical expressions*, in *Proc. ICASSP*, pp. 2439–2442, 1995.

[27] G. Costagliola, V. Deufemia, G. Polese, and M. Risi, *A parsing technique for sketch recognition systems*, in *Proc. VLHCC*, pp. 19–26, 2004.

[28] C. Bahlmann, B. Haasdonk, and H. Burkhardt, *On-line handwriting recognition with support vector machines: A kernel approach*, in *Proc. Intl. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 49–54, 2001.

[29] V. Deepu, S. Madhvanath, and A. G. Ramakrishnan, *Principal component analysis for online handwritten character recognition*, in *Proc. Intl. Conf. on Pattern Recognition (ICPR)*, pp. 327–330, 2004.

[30] L. A. Leiva, V. Alabau, V. Romero, A. H. Toselli, and E. Vidal, *Context-aware gestures for mixed-initiative text editing UIs,* Interact. Comput. **27** (2014), no. 1,.

[31] J. O. Wobbrock, A. D. Wilson, and Y. Li, *Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes*, in *Proc. Annual ACM Symp. on User Interface Software and Technology (UIST)*, pp. 159–168, 2007.

[32] L. Anthony and J. O. Wobbrock, *A lightweight multistroke recognizer for user interface prototypes*, in *Proc. Graphics Interface (GI)*, pp. 245–252, 2010.

[33] Y. Li, *Protractor: a fast and accurate gesture recognizer*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 2169–2172, 2010.

[34] L. Anthony and J. O. Wobbrock, *$N-protractor: a fast and accurate multistroke recognizer*, in *Proc. Graphics Interface (GI)*, pp. 117–120, 2012.

[35] J. B. Kruskal and M. Liberman, *The symmetric time-warping problem: from continuous to discrete,* Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison (1983) 125–161.

[36] U.-V. Marti and H. Bunke, *A full english sentence database for off-line handwriting recognition*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 705–708, 1999.

[37] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet, *Unipen project of on-line data exchange and recognizer benchmarks*, in *Proc. Intl. Conf. on Pattern Recognition (ICPR)*, pp. 29–33, 1994.

[38] A. Shivram, C. Ramaiah, S. Setlur, and V. Govindaraju, *IBM_UB_1: A dual mode unconstrained english handwriting dataset*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, 2013.

[39] L. Febvre and H.-J. Martin, *The coming of the book: the impact of printing 1450-1800*, vol. 10. 1997.

[40] G. Dimauro, S. Impedovo, R. Modugno, and G. Pirlo, *A new database for research on bank-check processing*, in *Proc. Workshop on Frontiers in Handwriting Recognition*, pp. 524–528, 2002.

[41] S. Srihari and E. Keubert, *Integration of handwritten address interpretation technology into the united states postal service remote computer reader system*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 892–896, 1997.

[42] A. H. Toselli, V. Romero, L. Rodríguez, and E. Vidal, *Computer Assisted Transcription of Handwritten Text*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 944–948, 2007.

[43] V. Romero, A. Toselli, and E. Vidal, *Character-level interaction in computer-assisted transcription of text images*, in *Proc. Intl. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pp. 539–544, 2010.

[44] V. Romero, A. H. Toselli, and E. Vidal, *Multimodal interactive handwritten text transcription*, vol. 80. World Scientific, 2012.

[45] J. Civera, J. Vilar, E. Cubel, A. Lagarda, S. Barrachina, F. Casacuberta, E. Vidal, D. Picó, and J. González, *A syntactic pattern recognition approach to computer assisted translation*, in *Proc. Joint IAPR Int. Workshops SSPR and SPR*, 2004.

[46] S. Barrachina, O. Bender, F. Casacuberta, J. Civera, E. Cubel, S. Khadivi, A. L. H. Ney, J. Tomás, and E. Vidal, *Statistical approaches to computer-assisted translation,* Computational Linguistics (2008) 3–28.

[47] L. Leiva, V. Romero, A. Toselli, and E. Vidal, *Evaluating an interactive-predictive paradigm on handwriting transcription: A case study and lessons learned*, in *Proc. Computer Software and Applications*, pp. 610–.617, 2011.

[48] A. H. Toselli, V. Romero, M. Pastor, and E. Vidal, *Multimodal interactive transcription of text images,* Pattern Recognition **43** (2010), no. 5, 1814–1825.

[49] V. Romero, A. H. Toselli, and E. Vidal, *Multimodal Interactive Handwritten Text Transcription*. World Scientific Publishing, 2011.

[50] S. Johansson, E. Atwell, R. Garside, and G. Leech, *The tagged lob corpus. user's manual,* Norwegian Computing Center for the Humanities (1986).

[51] A. P. Georgopoulos, J. F. Kalaska, and J. T. Massey, *Spatial trajectories and reaction times of aimed movements: effects of practice, uncertainty, and change in target location,* J. Neurophysiology **46** (1981), no. 4, 725–743.

[52] P. Morasso, *Spatial control of arm movements,* Experimental brain research **42** (1981), no. 2, 223–227.

[53] J. Soechting and F. Lacquaniti, *Invariant characteristics of a pointing movement in man,* J. Neuroscience **1** (1981), no. 7, 710–720.

[54] W. Abend, E. Bizzi, and P. Morasso, *Human arm trajectory formation.,* Brain: a journal of neurology **105** (1982), no. 2, 331–348.

[55] C. G. Atkeson and J. M. Hollerbach, *Kinematic features of unrestrained vertical arm movements,* J. Neuroscience **5** (1985), no. 9, 2318–2330.

[56] Y. Uno, M. Kawato, and R. Suzuki, *Formation and control of optimal trajectory in human multijoint arm movement,* Biol. Cybern. **61** (1989), no. 2, 89–101.

[57] H. Nagasaki, *Asymmetric velocity and acceleration profiles of human arm movements,* Experimental Brain Research **74** (1989), no. 2, 319–326.

[58] H. N. Zelaznik, R. A. Schmidt, and S. C. Gielen, *Kinematic properties of rapid aimed hand movements,* J. Motor Behavior **18** (1986), no. 4, 353–372.

[59] J. M. Hollerbach and T. Flash, *Dynamic interactions between limb segments during planar arm movement,* Biological Cybernetics **44** (1982), no. 1, 67–77.

[60] J. Ruitenbeek, *Invariants in loaded goal directed movements,* Biological Cybernetics **51** (1984), no. 1, 11–20.

[61] J. Soechting, *Effect of target size on spatial and temporal characteristics of a pointing movement in man,* Experimental Brain Research **54** (1984), no. 1, 121–132.

[62] T. Flash and N. Hogan, *The coordination of arm movements: an experimentally confirmed mathematical model,* J. Neuroscience **5** (1985), no. 7, 1688–1703.

[63] A. Feldman, *Functional tuning of the nervous system with control of movement or maintenance of a steady posture,* Biophysics **11** (1966), no. 1, 565–578.

[64] A. J. W. M. Thomassen, P. J. G. Keuss, and G. van Galen, *Motor aspects of handwriting,* Acta Psychol. **54** (1983), no. 1–3, 354.

[65] D. E. Meyer, J. E. K. Smith, S. Kornblum, R. A. Abrams, and C. E. Wright, *Speed-accuracy tradeoffs in aimed movements: Toward a theory of rapid voluntary action,* Atten. and Perform. **13** (1990), no. 23, 173–226.

[66] R. Plamondon, *A kinematic theory of rapid human movements: Part I: Movement representation and generation,* Biol. Cybern. **72** (1995), no. 4, 295–307.

[67] S. E. Engelbrecht, *Minimum principles in motor control,* J. Mathematical Psychology **45** (2001), no. 3, 497–542.

[68] J. D. Enderle and J. W. Wolfe, *Time-optimal control of saccadic eye movements,* IEEE T. Biomedical Engineering **1** (1987) 43–55.

[69] P. Neilson, *The problem of redundancy in movement control: the adaptive model theory approach,* Psychol. Res. **55** (1993), no. 1, 99–106.

[70] S. Edelman and T. Flash, *A model of handwriting,* Biol. Cybern. **57** (1987), no. 1-2, 25–36.

[71] R. Plamondon and F. Lamarche, *Modelization of handwriting: A system approach*, in *Graphonomics: Contemporary Research in Handwriting*, pp. 169–183, 1986.

[72] F. Leclerc, R. Plamondon, and G. Lorette, *Des gaussiennes pour la modélisation des signatures et la segmentation des tracés manuscrits,* Trait. Signal **9** (1992), no. 4, 347–358.

[73] M. A. Alimi, *Beta neuro-fuzzy systems,* 2003.

[74] P. Morasso, F. A. Mussa Ivaldi, and C. Ruggiero, *How a discontinuous mechanism can produce continuous patterns in trajectory formation and handwriting,* Acta Psychol. **54** (1983), no. 1, 83–98.

[75] F. J. Maarse, *The Study of Handwriting Movement: Peripheral Models and Signal Processing Techniques*. Swets & Zeitlinger, 1987.

[76] R. Plamondon and M. Djioua, *A multi-level representation paradigm for handwriting stroke generation,* Hum. Mov. Sci. **25** (2006), no. 4–5, 586–607.

[77] R. Plamondon, C. O'Reilly, C. Rémi, and T. Duval, *The lognormal handwriter: learning, performing, and declining,* Front. Psychol. **4** (2013), no. 1, 945:1–945:14.

[78] D. Martín-Albo, R. Plamondon, and E. Vidal, *Training of on-line handwriting text recognizers with synthetic text generated using the kinematic theory of rapid human movements*, in *Proc. Intl. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pp. 543–548, 2014.

[79] A. Fischer, R. Plamondon, C. O'Reilly, and Y. Savaria, *Neuromuscular representation and synthetic generation of handwritten whiteboard notes*, in *Proc. Intl. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[80] R. Plamondon, C. O'Reilly, J. Galbally, A. Almaksour, and E. Anquetil, *Recent developments in the study of rapid human movements with the kinematic theory: Applications to handwriting and signature synthesis,* Pattern Recognition Letters **35** (2014) 225–235.

[81] R. Plamondon, C. O'reilly, and C. Ouellet-Plamondon, *Strokes against stroke—strokes for strides,* Pattern Recognition **47** (2014), no. 3, 929–944.

[82] A. Van Gemmert, R. Plamondon, and C. O'Reilly, *Using the sigma-lognormal model to investigate handwriting of individuals with parkinson's disease*, in *Int. Conf. Graphonomics*, pp. 119–122, 2014.

[83] J. Galbally, R. Plamondon, J. Fierrez, and J. Ortega-Garcia, *Synthetic on-line signature generation. Part I: Methodology and algorithms,* Pattern Recognition **45** (2012), no. 7, 2610–2621.

[84] J. Galbally, J. Fierrez, J. Ortega-Garcia, and R. Plamondon, *Synthetic on-line signature generation. Part II: Experimental validation,* Pattern Recognition **45** (2012), no. 7, 2622–2632.

[85] C. O'Reilly and R. Plamondon, *Development of a sigma-lognormal representation for on-line signatures,* Pattern Recognition **42** (2009), no. 12, 3324–3337.

[86] R. Plamondon and A. M. Alimi, *Speed/accuracy trade-offs in target-directed movements,* Behavioral and Brain Sciences **20** (1997), no. 02, 279–303.

[87] W. Guerfali and R. Plamondon, *A new method for the analysis of simple and complex planar rapid movements,* J. Neuroscience Methods **82** (1998), no. 1,.

[88] C. Ghez and J. Krakauer, *Voluntary movement,* Principles of neural science **3** (1991) 622–624.

[89] R. Plamondon, *A kinematic theory of rapid human movements. Part II: Movement time and control,* Biol. Cybern. **72** (1995), no. 4, 309–320.

[90] R. Plamondon, C. Feng, and A. Woch, *A kinematic theory of rapid human movement. part IV: a formal mathematical proof and new insights,* Biological Cybernetics **89** (2003), no. 2, 126–138.

[91] R. Plamondon and W. Guerfali, *The 2/3 power law: When and why?,* Acta psychologica **100** (1998), no. 1, 85–96.

[92] W. Guerfali and R. Plamondon, *The delta lognormal theory for the generation and modeling of cursive characters*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 495–498, 1995.

[93] T. Varga, D. Kilchhofer, and H. Bunke, *Template-based synthetic handwriting generation for the training of recognition systems*, in *Proc. Biennial Conf. of the Int. Graphonomics Society*, pp. 206–211, 2005.

[94] D. Bullock, S. Grossberg, and C. Mannes, *A neural network model for cursive script production,* Biol. Cybern. **70** (1993), no. 1, 15–28.

[95] C. M. Privitera and R. Plamondon, *A system for scanning and segmenting cursively handwritten words into basic strokes*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 1047–1050, 1995.

[96] C. O'Reilly and R. Plamondon, *A software assistant for the design and analysis of neuromuscular tests*, in *Proc. BIOCAS*, pp. 107–110, 2007.

[97] M. Djioua and R. Plamondon, *An interactive system for the automatic generation of huge handwriting databases from a few specimens*, in *Proc. Intl. Conf. on Pattern Recognition (ICPR)*, pp. 1–4, 2008.

[98] D. Martín-Albo, R. Plamondon, and E. Vidal, *Improving sigma-lognormal parameter extraction*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, 2015.

[99] L. A. Leiva, D. Martín-Albo, and R. Plamondon, *Gestures à go go: Authoring synthetic human-like stroke gestures using the kinematic theory of rapid movements,* ACM Transactions on Intelligent Systems and Technology. In press.

[100] M. Liwicki, A. Graves, H. Bunke, and J. Schmidhuber, *A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, 2007.

[101] R. Plamondon and F. Maarse, *Writing generation model for health care neuromuscular system investigation,* IEEE T. on Systems, Man and Cybernetics (1989), no. 5, 1–16.

[102] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. The MIT Press, 3rd ed., 2001.

[103] N. Leduc and R. Plamondon, *A new approach to study human movements: The three dimensional Delta-Lognormal model*, in *Proc. Biennial Conf. of the Intl. Graphonomics Society (IGS)*, pp. 98–102, 2001.

[104] A. Almaksour, E. Anquetil, R. Plamondon, and C. O'Reilly, *Synthetic handwritten gesture generation using sigma-lognormal model for evolving handwriting classifiers*, in *Proc. Biennial Conf. of the Int. Graphonomics Society*, 2011.

[105] C. O'Reilly and R. Plamondon, *Design of a neuromuscular disorders diagnostic system using human movement analysis*, in *Int. Conf. Information Science, Signal Processing and their Applications (ISSPA)*, pp. 787–792, IEEE, 2012.

[106] M. Helmers and H. Bunke, *Generation and use of synthetic training data in cursive handwriting recognition*, in *Pattern Recognition and Image Analysis*, pp. 336–345. 2003.

[107] E. Anquetil, L. Miclet, S. Bayoudh, et al., *Synthetic on-line handwriting generation by distortions and analogy*, in *Proc. Biennial Conf. of the Int. Graphonomics Society*, pp. 10–13, 2007.

[108] J. Wang, C. Wu, Y.-Q. Xu, and H.-Y. Shum, *Combining shape and physical models for online cursive handwriting synthesis,* Int. J. Document Analysis and Recognition **7** (2005), no. 4, 219–227.

[109] Z. Lin and L. Wan, *Style-preserving english handwriting synthesis,* Pattern Recognition **40** (2007), no. 7, 2097–2109.

[110] J. Wang, C. Wu, Y.-Q. Xu, H.-Y. Shum, and L. Ji, *Learning-based cursive handwriting synthesis*, in *Proc. Intl. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pp. 157–162, 2002.

[111] T. Varga and H. Bunke, *Perturbation models for generating synthetic training data in handwriting recognition*, in *Machine Learning in Document Analysis and Recognition*, pp. 333–360, 2008.

[112] Y. Zheng and D. Doermann, *Handwriting matching and its application to handwriting synthesis*, in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pp. 861–865, 2005.

[113] P. Rao, *Shape vectors: an efficient parametric representation for the synthesis and recognition of hand script characters*, Sadhana **18** (1993), no. 1, 1–15.

[114] O. Stettiner and D. Chazan, *A statistical parametric model for recognition and synthesis of handwriting*, in *Int. Conf. Pattern Recognition*, vol. 2, pp. 34–38, 1994.

[115] H. Choi, S.-J. Cho, and J. H. Kim, *Generation of handwritten characters with bayesian network based on-line handwriting recognizers*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, p. 995, 2003.

[116] I. Guyon, *Handwriting synthesis from handwritten glyphs*, in *Proc. Intl. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pp. 140–153, Citeseer, 1996.

[117] C. Jawahar and A. Balasubramanian, *Synthesis of online handwriting in indian languages*, in *Proc. Intl. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, 2006.

[118] R. Saabni and J. El-Sana, *Efficient generation of comprehensive database for online arabic script recognition*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 1231–1235, 2009.

[119] J. M. Hollerbach, *An oscillation theory of handwriting,* Biological Cybernetics **39** (1981), no. 2, 139–156.

[120] G. Gangadhar, D. Joseph, and V. S. Chakravarthy, *An oscillatory neuromotor model of handwriting generation,* Int. Journal on Document Analysis and Recognition **10** (2007), no. 2, 69–84.

[121] P. Simard and Y. LeCun, *Reverse tdnn: an architecture for trajectory generation*, in *NIPS*, pp. 579–588, 1991.

[122] S. Bayoudh, H. Mouchère, L. Miclet, and E. Anquetil, *Learning a classifier with very few examples: analogy based and knowledge based generation of new examples for character recognition*, in *Machine Learning: ECML*, pp. 527–534, 2007.

[123] M. A. Slim, A. Abdelkarim, and M. Benrejeb, *Handwriting process modelling by artificial neural networks,* Int. J. Comput. Inf. Syst. Ind. Manag. Appl **5** (2013) 297–307.

[124] M. Djioua and R. Plamondon, *Studying the variability of handwriting patterns using the kinematic theory,* Hum. Mov. Sci. **28** (2009), no. 5, 588–601.

[125] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley, 2001.

[126] J. Liang, D. Doermann, and H. Li, *Camera-based analysis of text and documents: a survey,* Int. Journal on Document Analysis and Recognition (IJDAR) (2005) 84–104.

[127] C. H. Lampert, T. Braun, A. Ulges, D. Keysers, and T. M. Breuel, *Oblivious document capture and real-time retrieval*, in *Int. Workshop on Camera Based Document Analysis and Recognition*, pp. 79–86, 2005.

[128] M. Terry and E. D. Mynatt, *Recognizing creative needs in user interface design*, in *Proc. C&C*, pp. 38–44, 2002.

[129] J. Eisenstein and A. Puerta, *Adaptation in automated user-interface design*, in *Proc. Int. Conf. on Intelligent User Interfaces*, pp. 74–81, 2000.

[130] U.-V. Marti and H. Bunke, *Text Line Segmentation and Word Recognition in a System for General Writer Independent Handwriting Recognition*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 159–163, 2001.

[131] D. Keysers, F. Shafait, and T. M. Breuel, *Document image zone classification - a simple high- performance approach*, in *Proc. Int. Conf. on Computer Vision Theory*, pp. 44–55, 2007.

[132] A. Andrew, *Another efficient algorithm for convex hulls in two dimensions*, Information Processing Letters **9** (1979), no. 5, 216–219.

[133] R. E. Kalman, *A new approach to linear filtering and prediction problems*, Transactions of the ASME–Journal of Basic Engineering (1960).

[134] I. E. Sutherland, *Sketchpad: A man-machine graphical communication system*, Tech. Rep. 296, Lincoln Laboratory, MIT, 1963.

[135] M. Davis and T. Ellis, *The RAND tablet: A man-machine graphical communication device*, in *Proc. American Federation of Information Processing Societies (AFIPS)*, pp. 325–331, 1964.

[136] S. Smithies, K. Novins, and J. Arvo, *Equation entry and editing via handwriting and gesture recognition*, Behav. Inform. Technol. **20** (2001), no. 1, 53–67.

[137] P. Koch, W. Konen, , and K. Hein, *Gesture recognition on few training data using slow feature analysis and parametric bootstrap*, in *Proc. IEEE Intl. Joint Conf. on Neural Networks (IJCNN)*, 2010.

[138] H. Lü, J. A. Fogarty, and Y. Li, *Gesture Script: Recognizing gestures and their structure using rendering scripts and interactively trained parts*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 1685–1694, 2014.

[139] T. Ouyang and Y. Li, *Bootstrapping personal gesture shortcuts with the wisdom of the crowd and handwriting recognition*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 2895–2904, 2012.

[140] K. Patel, N. Bancroft, S. M. Drucker, J. Fogarty, A. J. Ko, and J. Landay, *Gestalt: Integrated support for implementation and analysis in machine learning*, in *Proc. Annual ACM Symp. on User Interface Software and Technology (UIST)*, pp. 37–46, 2010.

[141] S. Amini and Y. Li, *CrowdLearner: Rapidly creating mobile recognizers using crowdsourcing*, in *Proc. Annual ACM Symp. on User Interface Software and Technology (UIST)*, pp. 163–172, 2013.

[142] D. Ashbrook and T. E. Starner, *MAGIC: A motion gesture design tool*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 2159–2168, 2010.

[143] D. K. H. Kohlsdorf and T. E. Starner, *MAGIC summoning: Towards automatic suggesting and testing of gestures with low probability of false positives during use,* J. Mach. Learn. Res. **14** (2013), no. 1, 209–242.

[144] B. Caramiaux, N. Montecchio, A. Tanaka, and F. Bevilacqua, *Adaptive gesture recognition with variation estimation for interactive systems,* ACM T. on Interactive Intell. Syst. (2014).

[145] J. I. Hong and J. A. Landay, *SATIN: A toolkit for informal ink-based applications*, in *Proc. Annual ACM Symp. on User Interface Software and Technology (UIST)*, pp. 63–72, 2000.

[146] K. Kin, B. Hartmann, T. DeRose, and M. Agrawala, *Proton: Multitouch gestures as regular expressions*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 2885–2894, 2012.

[147] A. C. Long, J. A. Landay, and L. A. Rowe, *Implications for a gesture design tool*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 40–47, 1999.

[148] H. Lü and Y. Li, *Gesture Coder: A tool for programming multi-touch gestures by demonstration*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 2875–2884, 2012.

[149] H. Lü and Y. Li, *Gesture Studio: Authoring multi-touch interactions through demonstration and declaration*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 257–266, 2013.

[150] J.-W. Kim and T.-J. Nam, *EventHurdle: Supporting designers' exploratory interaction prototyping with gesture-based sensors*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 267–276, 2013.

[151] A. K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu, *A CAPpella: Programming by demonstration of context-aware applications*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 33–40, 2004.

[152] L. D. Spano, A. Cisternino, F. Paternò, and G. Fenu, *GestIT: A declarative and compositional framework for multiplatform gesture definition*, in *Proc. ACM SIGCHI Symp. on Engineering Interactive Computing Systems (EICS)*, pp. 187–196, 2013.

[153] B. Signer, U. Kurmann, and M. C. Norrie, *iGesture: A general gesture recognition framework*, in *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 954–958, 2007.

[154] B. Plimmer and I. Freeman, *A toolkit approach to sketched diagram recognition*, in *Proc. British HCI Group Annual Conference on People and Computers*, pp. 205–213, 2007.

[155] F. Beuvens and J. Vanderdonckt, *Designing graphical user interfaces integrating gestures*, in *Proc. ACM Intl. Conf. on Design of Communication (SIGDOC)*, pp. 313–322, 2012.

[156] G. van Seghbroeck, S. Verstichel, F. D. Turck, and B. Dhoedt, *WS-Gesture, a gesture-based state-aware control framework*, in *Proc. IEEE Intl. Conf. on Service-Oriented Computing and Applications (SOCA)*, pp. 1–8, 2010.

[157] R.-D. Vatavu, C.-M. Chera, and W.-T. Tsai, *Gesture profile for web services: An event-driven architecture to support gestural interfaces for smart environments*, in *Proc. Ambient Intelligence (AmI)*, pp. 161–176, 2012.

[158] D. Kohlsdorf, T. E. Starner, and D. Ashbrook, *MAGIC 2.0: A web tool for false positive prediction and prevention for gesture recognition systems*, in *Proc. IEEE Intl. Conf. on Automatic Face & Gesture Recognition and Workshops (FG)*, pp. 1–6, 2011.

[159] R. Plamondon and R. Baron, *A dedicated microcomputer for handwritten interaction with a software tool: system prototyping*, Journal of microcomputer applications **9** (1986), no. 1, 51–61.

[160] R. Plamondon and R. Baron, *On-line recognition of handprinted schematic pseudocode for automatic fortran code generator*, in *Proceedings of the Eight International Conference on Pattern Recognition*, pp. 741–744, 1986.

[161] F. Nouboud and R. Plamondon, *A handwriting interface to a schematic pseudocode generator*, Pattern Recognition-Architectures, Algorithms & Applications, World Scientific, Singapore (etc.) (1991) 301–310.

[162] J. Tiedemann, *News from OPUS - A collection of multilingual parallel corpora with tools and interfaces*, in *Recent Advances in Natural Language Processing*, pp. 237–248. John Benjamins, Amsterdam/Philadelphia, 2009.

[163] M. LaLomia, *User acceptance of handwritten recognition accuracy*, in *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pp. 107–108, 1994.

[164] S. Ishimaru, H. Nishizaki, and Y. Sekiguchi, *Effect of confusion network combination on speech recognition system for editing*, in *Proc. of APSIPA Annual Summit and Conf*, vol. 4, pp. 1–4, 2011.

[165] E. Granell and C.-D. Martínez-Hinarejos, *Multimodal output combination for transcribing historical handwritten documents*, in *Computer Analysis of Images and Patterns*, pp. 246–260, Springer, 2015.

[166] C. O'Reilly and R. Plamondon, *Can computer mice be used as low-cost devices for the acquisition of planar human movement velocity signals?*, Behav. Res. Methods **43** (2011), no. 1, 229–238.

[167] R.-D. Vatavu, L. Anthony, and J. O. Wobbrock, *Relative accuracy measures for stroke gestures*, in *Proc. Intl. Conf. on Multimodal Interaction (ICMI)*, pp. 279–286, 2013.

[168] R.-D. Vatavu, L. Anthony, and J. O. Wobbrock, *Gestures as point clouds: a $p recognizer for user interface prototypes*, in *Proc. Intl. Conf. on Multimodal Interaction (ICMI)*, pp. 273–280, 2012.