

Estimating the Pen Trajectories  
of Static Handwritten Scripts  
using Hidden Markov Models

by

Emli-Mari Nel

Dissertation approved for the degree of

Doctor

in

Electronic Engineering

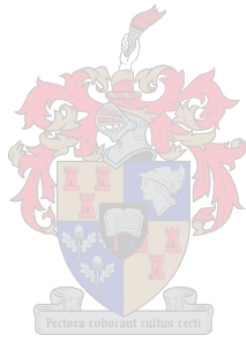
at the

University of Stellenbosch

December 2005

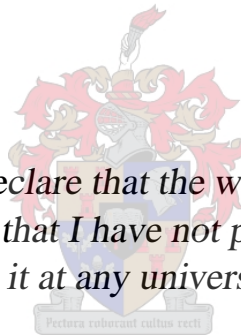
Promoters: Prof. J.A. du Preez

Prof. B.M. Herbst



## Declaration

*I, the undersigned, hereby declare that the work contained in this dissertation is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.*



---

Signature

---

Date

# Abstract

Individuals can be identified by their handwriting. Signatures are, for example, currently used as a biometric identifier on documents such as cheques. Handwriting recognition is also applied to the recognition of characters and words on documents—it is, for example, useful to read words on envelopes automatically, in order to improve the efficiency of postal services. Handwriting is a dynamic process: the pen position, pressure and velocity (amongst others) are functions of time. However, when handwritten documents are scanned, no dynamic information is retained. Thus, there is more information inherent in systems that are based on dynamic handwriting, making them, in general, more accurate than their static counterparts. Due to the shortcomings of static handwriting systems, static signature verification systems, for example, are not completely automated yet.

During this research, a technique was developed to extract dynamic information from static images. Experimental results were specifically generated with signatures. A few dynamic representatives of each individual's signature were recorded using a single digitising tablet at the time of registration. A document containing a different signature of the same individual was then scanned and unravelled by the developed system. Thus, in order to estimate the pen trajectory of a static signature, the static signature must be compared to pre-recorded dynamic signatures of the same individual. Hidden Markov models enable the comparison of static and dynamic signatures so that the underlying dynamic information hidden in the static signatures can be revealed. Since the hidden Markov models are able to model pen pressure, a wide scope of signatures can be handled. This research fully exploits the modelling capabilities of hidden Markov models. The result is a robustness to typical variations inherent in a specific individual's handwriting. Hence, despite these variations, our system performs well. Various characteristics of our developed system were investigated during this research. An evaluation protocol was also developed to determine the efficacy of our system. Results are promising, especially if our system is considered for static signature verification.

# Opsomming

Handskrif kan gebruik word om individue te identifiseer. Daar word steeds van handtekeninge gebruik gemaak as 'n biometriese identifiseerder op dokumente soos tjeks. Handskrifherkenning word ook onder andere gebruik vir die herkenning van karakters en woorde op dokumente. Dit is byvoorbeeld nuttig om die adresse op koeverte outomaties te lees om sodoende posdienste se effektiwiteit te verhoog. Handskrif is 'n dinamiese proses: die pen se posisie, druk en snelheid (onder andere) is funksies van tyd. Wanneer handskrif egter ingeskandeer word, gaan al hierdie omvattende dinamiese inligting verlore. Omdat stelsels gebaseer op statiese handskrif van minder inligting gebruik maak, is hulle meestal nie so akkuraat soos hulle dinamiese ekwivalente nie. Juis as gevolg van hierdie tekortkominge is statiese handtekeningverifikasie nog nie ten volle geoutomatiseer nie.

Gedurende hierdie navorsing is 'n tegniek ontwikkel om dinamiese inligting uit ingeskandeerde prentjies van handskrifte te onttrek. Eksperimentele resultate is gegenereer vanaf ingesamelde handtekeninge. 'n Paar dinamiese voorbeelde van elke individu se handtekening is opgeneem met behulp van 'n enkele digitale tablet tydens registrasie. 'n Dokument wat 'n ander voorbeeld van dieselfde individu se handtekening bevat, word dan ingeskandeer. Die stelsel onttrek slegs die trajek wat die pen gevolg het tydens die vorming van die handtekening. In die proses om die statiese handtekening te ontrafel, moet die statiese handtekening dus vergelyk word met reeds bestaande dinamiese handtekeninge. Verskuilde Markov modelle maak die vergelyking van die statiese en dinamiese handtekeninge moontlik, sodat die onderliggende dinamiese prosesse van statiese handtekeninge ontbloot kan word. Aangesien die verskuilde Markov modelle ook dinamiese pendruk kan modelleer, kan die ontwikkelde tegniek 'n wye verskeidenheid van statiese prentjies hanteer. Hierdie navorsing maak ten volle gebruik van verskuilde Markov modelle se modelleringskrag. Verskuilde Markov modelle is byvoorbeeld in staat om die variasies, wat kenmerkend is van 'n spesifieke individu se handtekening, te modelleer. Gevolglik lewer die stelsel steeds goeie resultate op, ten spyte van hierdie variasies. Verskeie van die ontwikkelde stelsel se karakteristieke is ondersoek. 'n Evalueringstegniek is ook ontwikkel om die akkuraatheid van die stelsel te meet. Resultate is belowend, veral vir die gebruik van die stelsel vir statiese handtekeningverifikasie.

# Acknowledgements

I would like to express my gratitude to the following individuals and institutions enabling me to complete this dissertation:

- My promoters Prof. Johan du Preez and Prof. Ben Herbst—I am grateful for their guidance and support. Without them, this research would simply not have been possible. In addition to their invaluable comments, I really appreciate their creativity and passion for life. Without a conscious effort, they have also provided me with non-academic related skills which I will always treasure.
- All the facilities provided by the DSP lab (including the great coffee). Ludwig Schwardt, Herman Engelbrecht, Johan Cronje and Francois Cilliers were system administrators of note during my time here and I will always appreciate Charlene’s helpfulness. I would also like to thank all the individuals from the DSP lab who contributed to the software contained in *Patrec*. The *Patrec* software was a singularly useful tool for this dissertation. I also made some great friends here (including my boyfriend).
- Harry Crossly, NNS, HB & MJ THOM Trust, Stellenbosch University, Ben Herbst, Johan du Preez and my parents for financial support during this research.
- Dr. Dolfing for the use of his database, especially for generating the results presented in [58].
- All the students at Stellenbosch University and my friends who contributed to US-SIGBASE.
- Ludwig Schwardt and Dr. Barry Scherlock for constructive comments during the final edits of [58].
- Robert Fanner and my parents for their love and support.

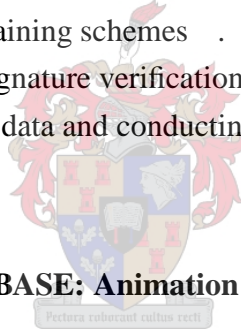
# Contents

<b>Abstract</b>	<b>i</b>
<b>Opsomming</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>List of symbols</b>	<b>x</b>
<b>List of acronyms</b>	<b>xiv</b>
<b>Glossary</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem statement and motivation . . . . .	1
1.2 Literature overview . . . . .	2
1.3 Overview of this dissertation . . . . .	5
1.3.1 Statistical pattern recognition: A brief background . . . . .	5
1.3.2 Preprocessing . . . . .	8
1.3.3 Deriving an HMM from a static script . . . . .	8
1.3.4 Evaluation protocol and results . . . . .	9
1.4 Research objectives . . . . .	10
1.5 Contributions . . . . .	10
<b>2 Literature study</b>	<b>14</b>
2.1 Restrictions . . . . .	14
2.2 Rule-based methods . . . . .	16
2.3 Graph-theoretical methods . . . . .	20
2.3.1 Graph-theoretical background . . . . .	20
2.3.2 Trajectory estimation algorithms that solve the Chinese postman and travelling salesman problems . . . . .	23
2.4 Local correspondences with dynamic exemplars . . . . .	26
2.5 Summary . . . . .	26

<b>3</b>	<b>Preprocessing</b>	<b>29</b>
3.1	Skeletonisation . . . . .	30
3.1.1	Introduction . . . . .	30
3.1.2	Literature synopsis . . . . .	31
3.1.3	Overview of our pseudo skeletonisation algorithm . . . . .	32
3.1.4	Shape partitioning . . . . .	34
3.1.5	Removing artifacts . . . . .	37
3.1.6	Results: The final skeletons . . . . .	43
3.1.7	Summary and conclusions . . . . .	44
3.2	Orientation normalisation . . . . .	46
3.2.1	Summary . . . . .	48
3.3	Resampling . . . . .	49
3.3.1	A critical point resampling scheme . . . . .	50
3.3.2	Summary . . . . .	51
<b>4</b>	<b>The HMM for a single-path static script</b>	<b>52</b>
4.1	HMM background . . . . .	53
4.2	First-order HMMs . . . . .	54
4.3	Second-order HMMs and their first-order equivalents . . . . .	57
4.3.1	General higher-order HMM theory . . . . .	57
4.3.2	Application of higher-order HMM theory to static handwritten scripts . . . . .	59
4.4	HMM topology for line segments . . . . .	60
4.5	HMM topology for crosspoints and endpoints . . . . .	64
4.6	HMM PDFs . . . . .	67
4.7	A summary of empirical HMM parameters . . . . .	68
4.8	Writer-specific HMM training . . . . .	69
4.9	Summary . . . . .	73
<b>5</b>	<b>The HMM for a multi-path static script</b>	<b>74</b>
5.1	Hierarchical HMMs . . . . .	76
5.2	Identifying pen-up and pen-down events . . . . .	78
5.3	Compensating for broken lines (spurious disconnections) . . . . .	80
5.4	The hidden state sequence (estimated pen trajectory) . . . . .	83
5.5	Summary . . . . .	87
<b>6</b>	<b>Experiments</b>	<b>89</b>
6.1	Evaluation protocol . . . . .	89
6.1.1	Existing evaluation protocols . . . . .	90
6.1.2	Using a left-to-right HMM to establish an evaluation protocol . . . . .	93



6.2	Database for experimental evaluation . . . . .	96
6.3	Experiments . . . . .	98
6.3.1	Overview of experiments . . . . .	98
6.3.2	Experimental results for different skeletonisation schemes . . . . .	101
6.3.3	Experimental results for different orientation normalisation schemes . . . . .	102
6.3.4	Experimental results for different resampling schemes . . . . .	105
6.3.5	Experimental results for different training schemes . . . . .	107
6.3.6	Typical errors . . . . .	108
6.4	Comparison with existing approaches . . . . .	111
6.5	Summary . . . . .	113
<b>7</b>	<b>Conclusions and future work</b>	<b>115</b>
7.1	Conclusions . . . . .	115
7.2	Future work . . . . .	118
7.2.1	Reducing the computational complexity . . . . .	118
7.2.2	Exploiting pressure information . . . . .	119
7.2.3	Extending the training schemes . . . . .	120
7.2.4	Applications: Signature verification and character recognition . . . . .	120
7.2.5	Recording more data and conducting more experiments . . . . .	122
	<b>Bibliography</b>	<b>124</b>
<b>A</b>	<b>The static scripts in US-SIGBASE: Animation examples.</b>	<b>132</b>



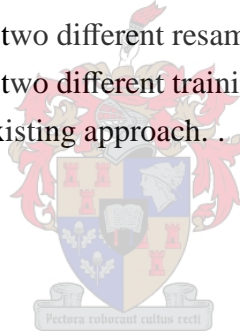
# List of Figures

1.1	A problematic signature to unravel . . . . .	3
1.2	A model for statistical pattern recognition from [36]. . . . .	6
1.3	A high-level diagram for our approach. . . . .	7
2.1	Presenting an “x” pattern as a 2D image and in a 3D space. . . . .	19
2.2	Illustration of graph-theoretical terminology . . . . .	21
2.3	Computing (a) the Voronoi diagram and (b) the Delaunay triangulation (straight-line dual of (a)) of a set of points (filled dots.) . . . . .	23
3.1	The skeleton of a static signature containing regions of multiple crossings . . .	31
3.2	Removal of skeleton artifacts. . . . .	33
3.3	Smoothing static image boundaries and calculating skeletons from the boundaries. .	36
3.4	Removing peripheral artifacts. . . . .	38
3.5	Identifying complicated intersections . . . . .	38
3.6	Calculating crosspoints to identify complicated intersections. . . . .	39
3.7	Correcting intersection artifacts . . . . .	41
3.8	Removing an intersection artifact using an extension of Step 4. . . . .	41
3.9	Intersection regions are united if their skeleton points are associated with the same short ribbon . . . . .	42
3.10	Merging three J-Ts. . . . .	42
3.11	Removing the last spurs . . . . .	43
3.12	Examples of our pseudo skeletonisation . . . . .	45
3.13	Aligning signatures with (a) PCA and (b) the Radon transform. . . . .	47
3.14	A projection of a static signature in the Radon domain . . . . .	48
3.15	Selecting high curvature points from a parametric curve. . . . .	51
4.1	Extracting dynamic information from a simple, single-path static signature . . .	53
4.2	Deriving the first-order HMM from a static signature skeleton . . . . .	55
4.3	Including an unambiguous velocity feature in a static image. . . . .	56
4.4	Calculating the first-order equivalents of second-order HMMs using the ORED algorithm. . . . .	58

4.5	The second-order HMM topology for a line segment . . . . .	61
4.6	The enlarged version of the right-hand column of Figure 4.5 illustrating the development of the HMM for a line segment. . . . .	62
4.7	The HMM topology for endpoints and crosspoints . . . . .	66
4.8	Deriving the first-order HMM from a dynamic exemplar. . . . .	71
5.1	A typical segmentation to identify disconnected parts in static scripts that correspond to single-path trajectories. . . . .	75
5.2	The HHMM $\lambda$ for the static character “i”. . . . .	78
5.3	Manipulating the HHMM $\lambda$ for the static character “i” to identify pen-up and pen-down events. . . . .	79
5.4	The locations of two disconnected endpoints in a broken line. . . . .	81
5.5	Compensating for broken lines . . . . .	82
5.6	Estimating the pen trajectory of a static “i” . . . . .	85
5.7	Estimating the pen trajectory of a multi-path signature . . . . .	87
6.1	Segmentation of a static skeleton at endpoints. . . . .	91
6.2	Using DP to calculate the optimal Levenshtein distance between two sequences. . . . .	92
6.3	The left-to-right HMM from our evaluation protocol for the string INTEREST . . . . .	95
6.4	An example of a typical scanned document in US-SIGBASE . . . . .	97
6.5	Measuring the accuracy of Radon-based versus PCA-based orientation normalisation. . . . .	104
6.6	A typical example from US-SIGBASE to explain experimental results for different resampling schemes. . . . .	107
6.7	Typical errors that are encountered in estimated pen trajectories . . . . .	109
6.8	Extracting dynamic information from a complicated signature . . . . .	111
7.1	Illustration of our system’s ability to separate test and training patterns from different classes. . . . .	122
A.1	The static scripts in US-SIGBASE that were unravelled to generate experimental results. . . . .	134

# List of Tables

2.1	A summary of related work . . . . .	17
6.1	A summary of all the experimental configurations. . . . .	99
6.2	Experimental results for two different skeletonisation schemes. . . . .	101
6.3	Experimental results, showing the average accuracy of recovered pen trajectories for single-path static scripts expressed as the correct percentages of the ground-truth path lengths that were extracted. . . . .	102
6.4	Experimental results for two different orientation normalisation schemes. . . . .	103
6.5	Experimental results for two different resampling schemes. . . . .	106
6.6	Experimental results for two different training schemes. . . . .	108
6.7	A comparison with an existing approach. . . . .	112



# List of symbols

All symbols are used according to the following standard:

1. Constant values, units and labels are not italic and not bold.
2. Scalars are italic and not bold.
3. Vectors are bold and not italic.
4. Matrices are bold and italic.
5. Functions are italic, not bold and include bracketed expressions, e.g.,  $f(\cdot)$ .

This section lists frequently recurring symbols that are defined in this dissertation.

## Symbols defined in Chapter 2.

$G$	The graph constructed from a static script consisting of vertices and edges.
$L(G)$	The line graph of $G$ , where the nodes of $L(G)$ correspond to the edges in $G$ .

## Symbols defined in Chapter 3.

$\alpha$	The ratio between the width $w$ and length $\ell$ of a ribbon.
$\mathbf{p}_i$	The skeleton point (2D coordinate) of $J-T_i$ .
$\mathbf{C}$	The covariance matrix $\mathbf{C}$ of the data that represent a shape.
$\mathbf{E}$	The matrix containing the eigenvalues of $\mathbf{C}$ on its main diagonal.
$\beta_j$	The eigenvalues of $\mathbf{E}$ sorted in descending order so that $\beta_j \geq \beta_{j+1}$ for $j = 1, 2, \dots, n - 1$ , where $n$ is the dimension of the data (see (3.3)).

## Symbols defined in Chapter 4.

$\lambda$	The shorthand notation for a first-order HMM, as defined by (4.2).
$\lambda'$	The second-order HMM derived from $\lambda$ .
$\lambda''$	The first-order equivalent of $\lambda'$ .
$N$	The number of emitting states in $\lambda$ .

$N'$	The number of emitting states in $\lambda''$ .
$M$	The number of skeleton samples in a static script.
$\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_M\}$	Matrix of unordered skeleton samples, where $\mathbf{p}_x$ is the 2D coordinate of sample $x$ .
$\mathbf{q} = \{q_1, q_2, \dots, q_N\}$	HMM emitting states.
$q_0$	Non-emitting initial HMM state.
$q_{N+1}$	Non-emitting terminating HMM state.
$f(\mathbf{x})$	HMM state observation likelihood defined by (4.1), where $\mathbf{x}$ is a $d$ -dimensional vector that must be matched to the PDF.
$\zeta_{ij}$	The index of the pair $ij$ so that $\zeta_{ij} \in \{1, \dots, N'\}$ , where $i, j \in \{1, \dots, N\}$ and $N$ is the number of states in $\lambda$ .
$q_{ij}$	First-order HMM state in $\lambda''$ , where the label $ij$ defines the state uniquely. The skeleton sample $\mathbf{p}_j$ is associated with $q_{ij}$ (indicated by the rightmost index $j$ ) and $q_{ij}$ is preceded by all states that share $\mathbf{p}_i$ (indicated by the leftmost index $i$ .)
$\mathcal{N}(\mathbf{u}_{ij}, \sigma)$	Spherical Gaussian PDF associated with $q_{ij}$ with mean $\mathbf{u}_{ij}$ and standard deviation $\sigma$ , so that $\mathcal{N}(\mathbf{u}_{ij}^P, \sigma_P)$ is the Gaussian PDF component that reflects pen position and $\mathcal{N}(\mathbf{u}_{ij}^V, \sigma_V)$ is the Gaussian PDF component that reflects pen direction at $q_{ij}$ ; see (4.5).
$\sigma'_P$ and $\sigma'_V$	Trained writer-specific standard deviations, estimated from $\sigma_P$ and $\sigma_V$ .
$\mathbf{A}$	The matrix representing the transition links of an HMM, where $a_{ij} = P(s_{t+1} = q_j   s_t = q_i)$ for a first-order HMM and $a_{ijk} = P(s_{t+1} = q_k   s_{t-1} = q_i, s_t = q_j)$ for a second-order HMM.
$\cos(\theta_{hij})$	The angle between the two straight lines connecting points $h$ to $i$ and $i$ to $j$ , as described by (4.3).
$T$	Number of samples in a dynamic exemplar.
$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T]$	$\mathbf{x}_t$ denotes a $d$ -dimensional feature vector at discrete-time instant $t$ , and $T$ is the number of feature vectors that represent a dynamic exemplar.
$\mathbf{s} = [s_1, s_2, \dots, s_T]$	The hidden state sequence $\mathbf{s} = [s_1, s_2, \dots, s_T]$ that results when $\mathbf{X}$ is matched to an HMM.

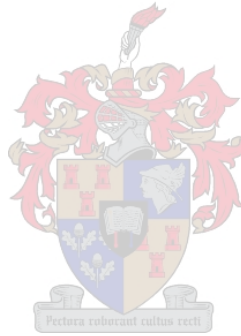
### Symbols defined in Chapter 5.

$N$	The number of sub-images that constitute a static script.
$\mathbf{P}_h = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{M_h}\}$	Matrix of unordered skeleton samples that constitute sub-image $h$ in a static script, where $\mathbf{p}_x$ is the 2D coordinate of sample $x$ .

$\lambda$	The bi-level HHMM of a static script constructed from $\mathbf{P}$ .
$\lambda'$	The single-level HMM representation of $\lambda$ .
$\lambda_h$	The HMM for a sub-image of a static script constructed from $\mathbf{P}_h$ .
$\mathbf{q} = \{q_1, q_2, \dots, q_{N+1}\}$	The higher-level emitting states of $\lambda$ .
$\mathbf{q}^h = \{q_1^h, q_2^h, \dots, q_{N_h}^h\}$	The lower-level emitting states of $\lambda_h$ .
$q_0$ and $q_{N+2}$	The higher-level non-emitting initial and terminating states of $\lambda$ .
$q_0^h$ and $q_{N_h+1}^h$	The lower-level non-emitting initial and terminating states of $\lambda_h$ .
$\mathcal{N}(\boldsymbol{\mu}_i, \sigma')$	Spherical Gaussian PDF associated with $q_i$ with mean $\boldsymbol{\mu}_i$ and standard deviation $\sigma$ , so that $\mathcal{N}(\boldsymbol{\mu}_i^{\text{P}}, \sigma_{\text{p}}')$ is the Gaussian PDF component that reflects pen position and $\mathcal{N}(\boldsymbol{\mu}_i^{\text{V}}, \sigma_{\text{v}}')$ is the Gaussian PDF component that reflects pen direction at $q_i$ ; see (4.5). Likewise, is $\mathcal{N}(\boldsymbol{\mu}_{i,h}, \sigma)$ the Gaussian PDF component associated with $q_i^h$ .
$\mathcal{U}_i(a, b)$	Uniform PDF (described by (5.1)) component that reflects pen pressure similarities at $q_i$ . Likewise, is $\mathcal{U}_{i,h}(a, b)$ associated with $q_i^h$ .
$f_i^{\text{P}}(\mathbf{x}_t^{1,2})$	The positional PDF component $\mathcal{N}(\boldsymbol{\mu}_i^{\text{P}}, \sigma_{\text{p}}')$ evaluated at $\mathbf{x}_t^{1,2}$ . Likewise, is $f_{i,h}^{\text{P}}(\mathbf{x}_t^{1,2})$ the evaluation of $\mathbf{x}_t^{1,2}$ at $\mathcal{N}(\boldsymbol{\mu}_{i,h}^{\text{P}}, \sigma_{\text{p}}')$ .
$f_i^{\text{V}}(\mathbf{x}_t^{3,4})$	The directional PDF component $\mathcal{N}(\boldsymbol{\mu}_i^{\text{V}}, \sigma_{\text{v}}')$ evaluated at $\mathbf{x}_t^{3,4}$ . Likewise, is $f_{i,h}^{\text{V}}(\mathbf{x}_t^{3,4})$ the evaluation of $\mathbf{x}_t^{3,4}$ at $\mathcal{N}(\boldsymbol{\mu}_{i,h}^{\text{V}}, \sigma_{\text{v}}')$ .
$f_i^{\text{F}}(x_t^5)$	The pen pressure PDF component $\mathcal{U}_i(a, b)$ evaluated at $x_t^5$ . Likewise, is $f_{i,h}^{\text{F}}(x_t^5)$ the evaluation of $x_t^5$ at $\mathcal{U}_{i,h}(a, b)$ .
$f_i(\mathbf{x}_t)$	Observation likelihood at state $i$ evaluated at $\mathbf{x}_t$ , defined by $f_i(\mathbf{x}_t) = f_i^{\text{P}}(\mathbf{x}_t^{1,2})f_i^{\text{V}}(\mathbf{x}_t^{3,4})f_i^{\text{F}}(x_t^5)$ . Likewise, is $f_{i,h}(\mathbf{x}_t) = f_{i,h}^{\text{P}}(\mathbf{x}_t^{1,2})f_{i,h}^{\text{V}}(\mathbf{x}_t^{3,4})f_{i,h}^{\text{F}}(x_t^5)$ at $q_i^h$ .
$\mathbf{A}$	The matrix representing the transition links of $\lambda$ , where $a_{ij} = P(s_{t+1} = q_j   s_t = q_i)$ .
$\mathbf{A}_h$	The matrix representing the transition links of $\lambda_h$ , where $a_{ij}^h$ is the weight for a transition from $q_i^h$ to $q_j^h$ .
$T$	Number of samples in a dynamic exemplar.
$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T]$	$\mathbf{x}_t$ denotes a $d$ -dimensional feature vector at discrete-time instant $t$ , and $T$ is the number of feature vectors that represent a dynamic exemplar.
$\mathbf{s} = [s_1, s_2, \dots, s_T]$	The hidden state sequence $\mathbf{s} = [s_1, s_2, \dots, s_T]$ that results when $\mathbf{X}$ is matched to an HMM. In this case $\mathbf{X}$ is matched to $\lambda'$ , so that $\mathbf{s}$ translates into the estimated pen trajectory of $\mathbf{P}$ .
$\delta$	The likelihood of $\mathbf{s}$ , as described by (5.4).
$\delta_{\text{w}}$	Weighted likelihood of $\mathbf{s}$ , as described by (5.5).

**Symbols defined in Chapter 6.**

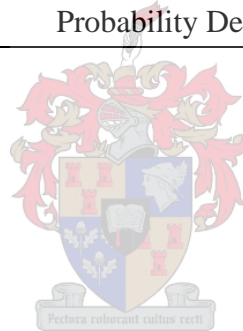
$\varpi_{\text{ground}}$	The ground-truth trajectory of a static script.
$\varpi_{\text{est}}$	The estimated pen trajectory of a static script.
$D$	The matrix from DP containing the values for a locally defined cost function $D(j, i)$ which reflect the similarity at node $(j, i)$ between $\varpi_{\text{est}}(j)$ (sample $j$ of $\varpi_{\text{est}}$ ) and $\varpi_{\text{ground}}(i)$ (sample $i$ of $\varpi_{\text{ground}}$ .)
$C$	The matrix from DP containing the final costs at all the nodes, where the cost $C(j, i)$ is assigned to the node $(j, i)$ , as described by Equation 6.1.
$\lambda_{\text{L2R}}$	The first-order HMM with a left-to-right topology from our evaluation protocol.





# List of acronyms

<b>DP</b>	Dynamic Programming
<b>HHMM</b>	Hierarchical Hidden Markov Model
<b>HMM</b>	Hidden Markov Model
<b>MAP</b>	Maximum A Posteriori
<b>ML</b>	Maximum Likelihood
<b>ORED</b>	Order Reducing
<b>PCA</b>	Principle Component Analysis
<b>PDF</b>	Probability Density Function



# Glossary

This section presents an abbreviated glossary for terms that occur frequently in this dissertation. A more detailed index of terms with page references are provided at the end of this dissertation, after the appendices.

<b>Allographic variations</b>	Variations of the same handwritten character or word due to different writer populations.
<b>Biometric measurement</b>	Quantification of the attributes of an individual that helps to identify a person uniquely.
<b>Chinese postman problem</b>	The search for a Eulerian cycle in a graph.
<b>Critical point resampled curve</b>	The resampled curve that results when selecting the most important points (critical points) from an original parametric curve.
<b>Crosspoint</b>	A skeleton sample connected to more than two adjacent skeleton samples.
<b>Delaunay triangulation</b>	An angle-optimal triangulation from a set of points, where the minimum angle over all the constructed triangles are maximised.
<b>Deletion</b>	A sample that occurs in a static script's ground-truth pen trajectory and not in the script's estimated pen trajectory.
<b>Dynamic counterpart</b>	The on-line version of a static script recorded while the handwriting was generated on the document.
<b>Dynamic exemplar</b>	A dynamic representation (not a dynamic copy) of a static handwritten script recorded at the time of registration.
<b>Edge</b>	A line that connects two successive control points.
<b>Endpoint</b>	A skeleton sample connected to only one adjacent skeleton sample.
<b>Euclidean resampled curve</b>	A parametric curve where the distance between any two successive samples is approximately the same.
<b>Feature vectors</b>	A sequence of $d$ -dimensional quantifiable characteristics describing a pattern.

<b>Geometric variations</b>	Shape variations, e.g., position, orientation, size and slant variations.
<b>Graph-theoretical approaches</b>	Methods that construct graphs from static scripts. The Chinese postman or travelling salesman problems are then typically solved to estimate the pen trajectories of the scripts.
<b>Ground-truth trajectory</b>	The pen trajectory derived by matching the dynamic counterpart to the HMM of a static script.
<b>Insertion</b>	A sample that occurs in a static script's estimated pen trajectory and not in the script's ground-truth pen trajectory.
<b>Intersection artifacts</b>	Skeleton artifacts where two or more lines that should intersect fail to cross each other in a single point.
<b>Levenshtein distance</b>	The smallest number of elementary operations required to transform one sequence into another sequence.
<b>Line segment</b>	A sequence of connected segment points.
<b>Multi-path static script</b>	A static handwritten script that consists of one or more single-path trajectories.
<b>Neighbouring states</b>	States that are associated with adjacent skeleton samples.
<b>Off-line handwriting</b>	A static 2D image of handwriting usually recorded with a scanner.
<b>On-line handwriting</b>	Dynamic handwriting captured using an electronic device, e.g. a digitising tablet, that is able to record the pen's positions, pressure and tilt as it moves across the surface of the tablet.
<b>Orientation of a script</b>	The specific overall or average direction relative to the horizontal axis in which the handwriting is generated.
<b>Path</b>	A list of successive control points, e.g., skeleton samples or vertices in ( $G$ ), where successive control points are connected by edges.
<b>Peripheral artifacts</b>	Spurs attached to the skeleton of an image.
<b>Rule-based methods</b>	Methods that estimate the pen trajectories of static scripts using a prior set of heuristic rules that try to mimic the underlying temporal principles for generating handwriting.
<b>Segment point</b>	A skeleton sample having only two adjacent skeleton neighbours.

<b>Self-loop</b>	An HMM transition link that connects a state back to itself.
<b>Sequence variations</b>	Variations in the order in which pen positions may be produced.
<b>Single-path trajectory</b>	An on-line handwritten curve created with uninterrupted, non-zero pressure.
<b>Skeleton</b>	A collection of thin lines that mostly coincides with the centreline of the original image.
<b>Skip-link</b>	An HMM transition link connecting two states that are separated by a neighbour common to both.
<b>Spurious disconnections</b>	Unexpected broken lines in a static script.
<b>Standard skeletons</b>	Skeletons from skeletonisation or thinning techniques that do not attempt to skeleton artifacts.
<b>Static script</b>	A 2D image of handwriting, e.g., cursive handwriting and signatures.
<b>Sub-image</b>	A set of contiguous samples that represent a shape.
<b>Substitution</b>	A sample from a static script's estimated pen trajectory that is erroneously mapped to a sample from the script's ground-truth pen trajectory.
<b>Travelling salesman problem</b>	The search for the shortest Hamilton cycle in a weighted complete graph.
<b>Writer-specific training</b>	Our HMM training scheme that estimates a unique $\sigma'_p$ and $\sigma'_v$ for each individual.
<b>Zero-pressure state</b>	An additional emitting state in our HMM that enables us to identify where an individual lifted the pen.

# Chapter 1

## Introduction

### 1.1 Problem statement and motivation

Producing cursive writing or handwritten signatures on documents involves a dynamic process: the pen's position, pressure, tilt and angle are functions of time. The end result, however, is a static image with little, if any, dynamic information encoded in it. This dissertation investigates the problem of extracting the pen trajectories that created a static handwritten script, i.e., the paths that the pen followed over the document. Thus, the problem is to unravel the script and present it as a chronological collection of parametric curves.

A *biometric measurement* quantifies attributes of an individual that help to identify a person uniquely. Biometric measurements can be either physiological or behavioural. *Physiological measurements* relate to the inherent physiological characteristics of an individual, e.g., iris patterns and fingerprints. *Behavioural measurements* relate to spontaneous or learned acts that are carried out by an individual, e.g., cursive handwriting and signatures [24]. In general, behavioural measurements are less intrusive than physiological measurements. Nevertheless, the choice of biometric measurement depends on the application domain, e.g., Plamondon and Srihari [64] note that signatures are still the most widely accepted means of identification, socially and legally.

Handwriting can be either on-line or off-line. *On-line* handwriting is captured using an electronic device, e.g., a digitising tablet, that is able to record the pen's position, pressure and tilt as it moves across the surface of the tablet. *Off-line* handwriting is typically recorded with a scanner to present the document as a 2D static image. Behavioural measurements of an individual can be extracted from on-line and off-line handwriting. These measurements are useful for a wide range of applications. Although on-line systems are mostly more reliable than their off-

line versions, as a means of personal identification, off-line systems are, in many cases, more economically viable and sufficiently accurate for the required application. Off-line systems are, e.g., sufficient for the automatic interpretation of handwritten postal addresses on envelopes and reading courtesy amounts on bank cheques [64].

Plamondon and Srihari [64] endorse the relevance of the research topic with the following statement: “The success of on-line systems makes it attractive to consider developing off-line systems that first estimate the trajectory of the writing from off-line data and then use on-line recognition algorithms. However, the difficulty of recreating the temporal data has led to few such feature extraction systems so far.” Munich and Perona [56] have also shown that the pen trajectories of signatures contribute to an effective on-line signature verifier. Thus, it is concluded that estimated pen trajectories of static scripts are particularly useful for automatic handwritten character or word recognition, or for the verification of signatures.

The question is therefore to what extent is it possible to extract dynamic information from static handwritten scripts. Since one must deal with dynamic information loss incurred in static images, Park [60] relates this problem to the recovery of 3D depth information from single 2D images.

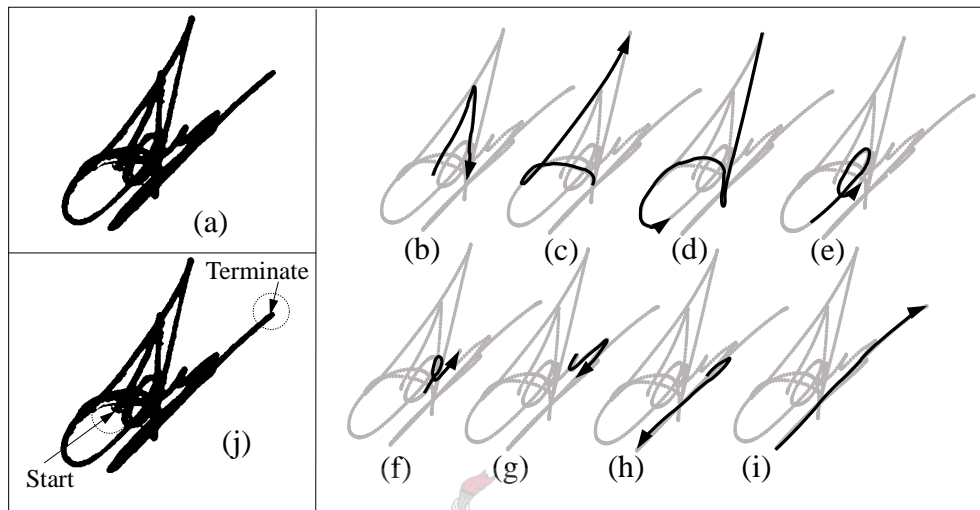
Literature on methods that do not specialise their trajectory estimation algorithms to cursive or language-specific handwriting is sparse. It should be noted that it is not compulsory in South Africa (or in Europe for that matter) for a person’s signature to be readable. Signatures therefore tend to be unpredictable. There are many examples of signatures containing so many regions of self-intersection that even humans find these signatures difficult to unravel. It is therefore challenging to create a robust heuristic framework that can deal with almost any type of handwritten script.

## 1.2 Literature overview

This section discusses typical problems encountered when estimating the pen trajectory of static handwritten scripts. A summary of how existing literature deals with these problems is also presented. Chapter 2 elaborates on the related techniques mentioned in this section.

There are several difficulties that need to be overcome when recovering the pen trajectory from a static handwritten script. These difficulties are compounded when the line densities and line widths at intersection regions are high. An example of a problematic signature containing such regions is shown in Figure 1.1(a). When handwriting is simultaneously recorded on a digitising tablet and on paper, both the static script and the *dynamic counterpart* of the handwriting are

available. The dynamic counterpart of the static signature in Figure 1.1(a) is rendered as grey lines in Figures 1.1(b)-(i). The pen positions that generated the dynamic counterpart are animated using solid arrows. It should be noted that such a dynamic counterpart is not available when unravelling a static script. The dynamic counterpart of Figure 1.1(a) is, in this case, shown only to illustrate typical difficulties arising when dealing with such a complicated signature.



**Figure 1.1:** A problematic signature to unravel. (a) A static signature containing intersection regions with high line densities and thick line widths. (b)-(i) Animation of the dynamic pen positions (solid arrows) that generated the dynamic counterpart (grey lines) of (a). (j) Identifying the starting and terminating positions (labelled arrows) of the static signature in (a).

The first difficulty is to find the starting and terminating positions of the static script—these positions are often hidden inside the image (especially where signatures are concerned) and not visible at all. Due to this ambiguity, strict constraints are normally required. Typically, it is assumed that the pen trajectory must start and terminate at distinct positions [33, 40, 50]. Thus, characters such as “o”, cannot be successfully unravelled. Without prior knowledge, it is almost impossible to determine where the signature in Figure 1.1(a) starts and terminates. It is, however, easy to approximate the starting and terminating positions (dotted circles in Figure 1.1(j)) from the dynamic counterpart in Figure 1.1(b)-(i).

The problem of finding the starting and terminating positions of a static script is more challenging if the script consists of multiple single-path trajectories, where a *single-path trajectory* refers to a single curve created with uninterrupted, non-zero pen pressure. A static script that consists of one single-path trajectory is referred to as a *single-path static script*, whereas one that consists of one or more single-path trajectories is called a *multi-path static script*. Pressure information is vital to determine where the writer lifts the pen. Wirotius et al. [84], e.g., note that the grey-levels within handwritten text are linked to pressure and writing speed when text is

produced. This information is, however, unreliable if, e.g., the script becomes indistinct due to multiple crossings. In general, it is therefore difficult to extend techniques that trace single-path handwritten scripts to deal with multi-path scripts if no prior on-line pressure information is available. Note, e.g., that it is almost impossible to determine how many single-path trajectories constitute the static signature in Figure 1.1(a). However, the pen pressure of the dynamic counterpart in Figure 1.1(b)-(i) reveals that the static signature in Figure 1.1(a) consists of one single-path trajectory. Because of these difficulties some studies deal only with single-path static scripts [58, 40].

Signatures often have complicated regions consisting of many intersections making it difficult to track a particular path through those regions. One possibility is to assume that the direction of a line is maintained when entering and leaving an intersection. A choice between the different possibilities at the intersection is then typically based on some local smoothness criteria, as in [54, 9, 44, 11, 34]. This approach is, however, insufficient to resolve ambiguities completely—if the script becomes indistinct due to a large number of intersections in a small area, local information is not sufficient to find the correct path. Additional assumptions may then be necessary, e.g., restricting the number of lines that can cross at an intersection [33, 40]. It is evident from Figure 1.1(a) that such a restriction is not necessarily valid in cases where signatures are concerned. In general, methods that make local choices at intersections have difficulty taking context into account. Several studies therefore include global information by modelling the pen trajectory estimation problem as a graph-theoretical problem [2, 38, 37, 43, 40, 41, 4, 3].

As a rule, the studies mentioned above, use only the 2D image of the script. Another approach is to record dynamic representatives of the static script captured with a digitising tablet at the time of registration [31, 51]. We refer to such dynamic representatives of the static script as *dynamic exemplars*. The idea is to compare a given static script with the pre-recorded dynamic exemplars. It is important to note that the static image is compared with generic dynamic representatives, and not a dynamic copy of itself. There is a notable advantage to such systems: only a single tablet is required at the registration phase. On-line systems often require a tablet at each signing post, which makes it economically infeasible for many applications.

We have indicated how easy it is to estimate the static script's starting and terminating positions if a dynamic counterpart is available. However, it is more complicated to do the same if only dynamic exemplars are available. When using pre-recorded dynamic exemplars, a problem arises with regard to modelling dynamic exemplar variations. Examples of such variations are geometric, allographic and sequencing variations [64]. *Geometric variations* refer to shape variations, e.g., position, orientation, size and slant variation. *Allographic variations* refer to variations of the same handwritten character or word due to different writer populations. *Sequence variations* refer to variations in the order in which pen positions may be produced. Sequence variations are



increased by the correction of spelling errors, slips of the pen, and letter omissions and insertions. A system, developed using prior dynamic exemplar information, should be able to draw on a comprehensive set of variations so that the additional information from the exemplars can be exploited to partially resolve ambiguities. Nevertheless, some heuristic measures may still be required to resolve the ambiguities completely, e.g., even though Guo et al. [31] and Lau et al. [51] employ pre-recorded dynamic exemplars they rely on local choices at intersections.

A dynamic exemplar is also valuable in resolving another difficulty, namely identifying turning points, where the pen stops and then reverses direction. It should be clear that static scripts retain no information about the return portion of a pen trajectory that stops and then reverses direction, returning along the same path. To simplify the problem, some studies restrict the number of times the pen can revisit a line [33, 40].

We have shown in this section that pre-recorded dynamic exemplars are invaluable in addressing several difficulties when estimating the pen trajectory of a static script. It is evident from existing literature that a lack of prior dynamic information typically necessitates the introduction of several restrictions for simplification. More details of existing approaches are documented in Chapter 2.

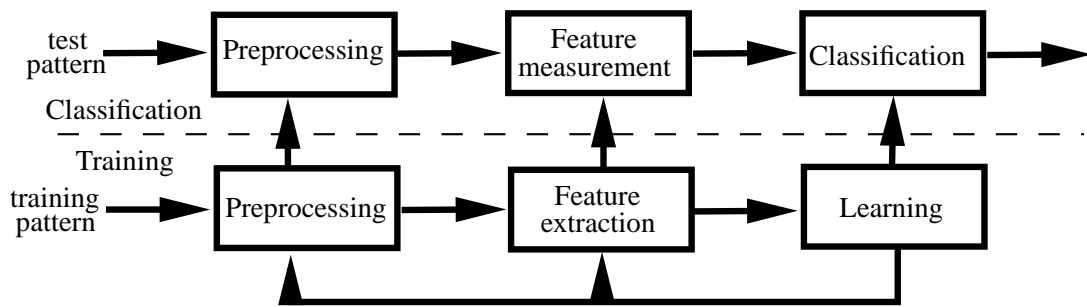


## 1.3 Overview of this dissertation

### 1.3.1 Statistical pattern recognition: A brief background

**A typical statistical pattern recognition system.** In the context of this dissertation, a *pattern* is defined from [6] as “a regular or logical form, order, or arrangement of parts”. In *statistical* pattern recognition, a pattern is described by a sequence of  $d$ -dimensional quantifiable characteristics called *feature vectors*. To distinguish between different patterns, one has to establish suitable decision boundaries. Jain et al. [36] describe a typical statistical pattern recognition system with a chart equivalent to Figure 1.2.

Figure 1.2 illustrates that a statistical pattern recognition system typically operates in two modes: Training and classification. *Training* describes the process in which characteristics of applicable patterns (training patterns) are learned to establish a comprehensive system. *Classification* is the process in which an input pattern (test pattern) must be assigned to a certain class based on the features that are measured from it. If two patterns belong to different classes, a good pattern recognition system would maximise their separability. Likewise, if they belong to the same class, the system must minimise their separability. The system’s ability to calculate



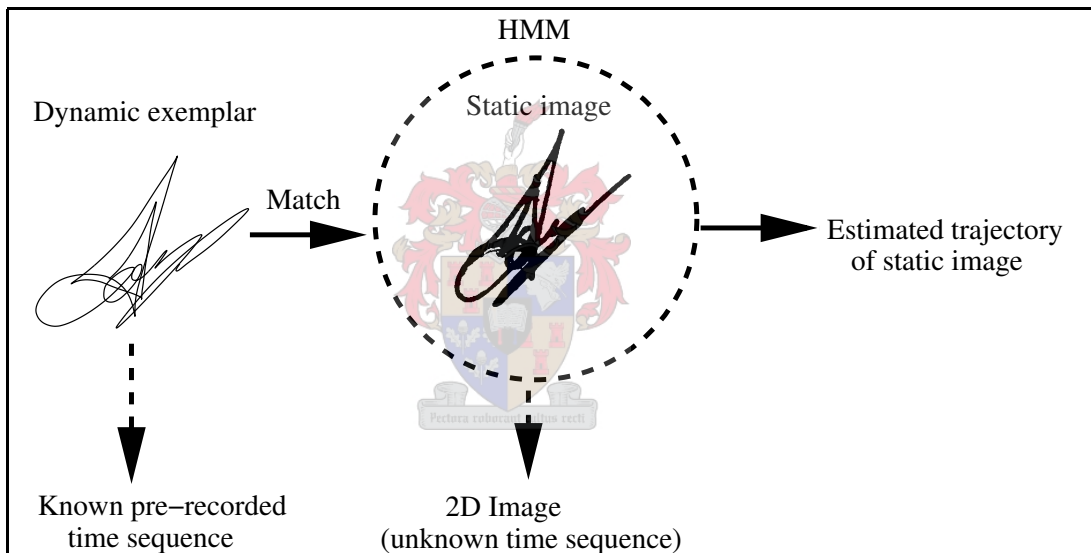
**Figure 1.2:** A model for statistical pattern recognition from [36].

decision boundaries depends on the features selected by the feature extraction module. Increasing the number of features typically leads to more accurate results. The preprocessing module must extract a pattern from its background, remove noise, and normalise it so that the pattern can be represented in a compact form. The feedback path allows the designer to optimise the applicable modules.

**Hidden Markov Models (HMMs).** An HMM is a probabilistic model that models a time dependent sequence of events with a sequence of *states* connected by *transitions links* [68]. An HMM describes a dynamic process that evolves from one state to the next. HMMs have been used successfully in many applications that model sequential data statistically, most notable speech recognition. Jain et al. [36] note that models using the Markov structure in speech compresses the data to what is physically meaningful, thereby simultaneously improving classification accuracy. Each state has an associated Probability Density Function (PDF). HMM observation PDFs reflect similarities between a test pattern and the training data. The HMM *topology* specifies the interconnection of states. Transitions between states are weighted with transition probabilities. The *order* of an HMM determines the number of previous states that can be remembered by the HMM at each state.

An application of HMMs, relevant to this research, is on-line signature verification [53, 75]. It is typically required that a collection of dynamic signatures is recorded for each individual at the registration phase. In the context of Figure 1.2, these dynamic signatures are the training patterns. Training and test signatures are normalised during preprocessing. Such normalisation typically translates, rotates and scales the signatures so that they are aligned. Typical features that are extracted from the normalised signatures are discrete samples of the dynamic pen positions, velocity and pressure. An HMM is then constructed from the feature vectors that represent the training data. The HMM parameters are trained for each individual. Features are then measured from the test signature and matched to the trained HMM. The degree of similarity between the HMM and test signature is quantified so that the test signature can be classified as a forgery or a genuine signature. The success of these systems is primarily due the HMM's ability to model not only the magnitude of the variations but also the *nature* of the variations.

**Our approach within a statistical framework.** To model static images with HMMs poses the problem of modelling 2D data with 1D observation sequences. We make use of pre-recorded dynamic exemplars to estimate the pen trajectory of a static handwritten script. In the context of Figure 1.2, the dynamic exemplars are the training patterns and the test pattern is the static image of the script. The static image is quantified as 2D feature vectors occurring in *no specific sequence*. Thus, a conventional match, as illustrated by the on-line signature verification example above, between a trained HMM and the static image is not applicable. The following solution addresses the problem: An HMM is constructed from the static image, i.e., from the test pattern. The training (pre-recorded) data (dynamic exemplar) is then matched to the HMM in the process to estimate the pen trajectory of the image. These concepts are illustrated in Figure 1.3. A dynamic exemplar, i.e., a known sequence of samples, is matched to the HMM (dashed circle) of a static image. This match enables one to estimate the unknown sequence of samples that constitute the static image.



**Figure 1.3:** A high-level diagram for our approach.

Paradoxically, for this application, the conventional employment of test and training data, specifically for an HMM is *reversed* as follows: Usually an HMM describes a dynamic process and represents the training (pre-recorded) data. The test (newly acquired) data are then matched to the HMM. In this application, however, an HMM represents a static image which forms the test (newly acquired) data. The training (pre-recorded) data is then matched to the HMM in the process to estimate the pen trajectory of the image. Accordingly, the topology for our HMM is not fixed, i.e., it is dependent on the structure of the static image and our training schemes have to be adapted.

In the context of Figure 1.2, the feature measurement module derives an HMM from a static script. The dynamic exemplars are then compared with this HMM to establish a point-wise correspondence between the static script and each dynamic exemplar. A suitable dynamic ex-

emplar is then chosen to reveal the pen trajectory of the static script. Classification, in this case, consists of choosing the most likely pen trajectory, as determined by the HMM and dynamic exemplars. The output of the classification module in Figure 1.2 is therefore the estimated pen trajectory of the static script. On-line techniques can then be applied to the estimated pen trajectory in, e.g., an off-line handwriting recognition system with a restricted library or in an off-line signature verification system. These possible applications are discussed in Section 7.2.4 with some preliminary results. It should be noted that a complete implementation of a handwriting recognition or verification system has not been pursued during this research. Instead, we have developed an evaluation protocol to quantify the accuracy of estimated pen trajectories. The rest of this section describes the different modules of Figure 1.2 in more detail.

### 1.3.2 Preprocessing

Static handwritten scripts must be extracted from the documents on which they were created. Thus, they are not in a form suitable for creating an HMM. They must also resemble on-line data so that they are comparable with dynamic exemplars. A substantial amount of preprocessing is therefore required. Preprocessing is fully treated in Chapter 3. The most important preprocessing steps include:

1. *Orientation normalisation*: A method based on the Radon transform is employed to align the general orientations of a static script and a pre-recorded dynamic exemplar; see Section 3.2.
2. *Skeletonisation*: In order to extract a parametric curve from a static image, a skeleton is derived from the image through a thinning process. A *skeleton*, in the context of this research, is a collection of thin lines that coincides mostly with the centreline of the original image. A number of enhancements particular to this application is introduced for standard skeletonisation/thinning procedures, as described in Section 3.1.
3. *Resampling*: The dynamic exemplars and static skeletons must be parameterised and resampled similarly before they are compared, as discussed in Section 3.3.

### 1.3.3 Deriving an HMM from a static script

**Deriving the HMM.** After preprocessing, an HMM is derived from the skeleton of a static script, as discussed in Chapters 4 and 5. Our HMM, derived from a static skeleton, describes the pen trajectory that created the skeleton. Each state has an associated PDF, embedding geometric shape information of the skeleton. Transitions between states are weighted with tran-

sition probabilities to dictate the choices of pen movements between skeleton samples. HMMs designed specifically for single-path static scripts are discussed in Chapter 4. Chapter 5 shows how to extend these HMMs to deal with multi-path static scripts.

A basic first-order HMM constructed from a single-path static script is described in Section 4.2. However, this HMM is not sufficient to resolve ambiguities in regions with multiple intersections. The problem is due to a loss of context caused by the use of first-order HMMs: state transitions depend only on the current state. Plamondon and Srihari [64] note that any observable signal from a handwritten trajectory is affected by at least both the previous and successive trajectories. Transitions of higher-order HMMs depend not only on the current state, but also on the previous states. Higher-order HMMs are therefore much better equipped to take context into account. Usually, higher-order HMMs tend to be computationally expensive. In this study, however, we use second-order HMMs with sparse transition probability matrices, reducing the computational cost to a manageable level. The suitable second-order HMM that is derived from a basic first-order HMM is described in Section 4.3. Further context is incorporated by comparing not only pen positions but also local line directions. It is shown in Section 5.2 how the pen pressure of the dynamic exemplars can be exploited to extend the HMMs for single-path scripts to deal with multi-path scripts. Normally, both the state observation PDFs and the transition probabilities are obtained through a training process. Data sparseness is a serious problem in our application, which necessitated the adaptation of our training algorithms. This is discussed in Section 4.8.

**Estimating the pen trajectory.** The next step is to compare the constructed HMM with pre-recorded dynamic exemplars of the static image. This is done using the Viterbi algorithm [68]. The result is an optimal state sequence that can be translated into the estimated pen trajectory of the static script, as discussed in Section 5.4.

### 1.3.4 Evaluation protocol and results

**Evaluation protocol.** In general, it is not entirely straightforward to assess the efficacy of an estimated pen trajectory. An obvious solution is to record a static script simultaneously on paper and on a digitising tablet, so that the dynamic counterpart of the static script is available. The dynamic counterpart can then be compared to the estimated pen trajectory (computed from a different dynamic exemplar) of the static script. Due to imperfect recording devices and subsequent processing, the image skeleton may differ from its exact dynamic counterpart. A one-to-one correspondence between the static script and its dynamic counterpart is therefore not available. Hence, a ground-truth trajectory is extracted from the static script. The *ground-truth trajectory* is the estimation of the exact pen trajectory that generated a static script's skele-

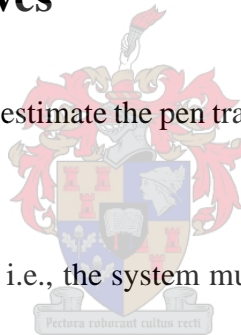
ton, and is calculated by comparing the script’s dynamic counterpart with its skeleton. The ground-truth and the estimated pen trajectories (derived from the same image skeleton) are then compared, as described in Section 6.1. An error measure is calculated from these comparisons to quantify the accuracies of the estimated pen trajectories.

**Results.** Results are generated with US-SIGBASE; see Section 6.2. To the best of our knowledge, a standardised database that contains on-line and off-line versions of signatures does not exist. US-SIGBASE was collected as part of this research, and consists of signatures for 51 individuals that were recorded simultaneously on paper and a digitising tablet. Results are generated by randomly selecting a static image for each individual and estimating pen trajectories from the selected images. The estimated pen trajectories are evaluated as described in Section 6.1. Experimental results show that our HMM is able to estimate approximately 88% of the ground-truth trajectories correctly, as described in Section 6.3.

## 1.4 Research objectives

The objective of this research is to estimate the pen trajectories of static handwritten scripts with the following requirements:

- The system must be *robust*, i.e., the system must not be highly sensitive to variations in static scripts.
- Estimated pen trajectories must be *accurate*. The efficacy of the pen trajectory estimation algorithm must be evaluated objectively in order to produce quantifiable results.



## 1.5 Contributions

- **An original approach.** We have managed to estimate the pen trajectories of static handwritten scripts by using a novel method—to the best of our knowledge, we are the first to use HMMs for this purpose. Guo et al. [31] establish a local correspondence between a static image and a dynamic exemplar. It is shown in Chapter 2, however, that their approach is fundamentally different from our approach. Quantifiable results show that our approach is accurate. Preliminary results show that our pen trajectory estimation algorithm can be especially useful in an off-line signature verification application.
- **Characteristics of our HMM contributing to a robust and accurate system.** By virtue of our HMM’s design, described in Chapters 4 and 5, we have managed to solve the fol-

lowing problems, mentioned in Section 1.2, that are in combination prevalent in existing approaches:

1. The initial/terminating transition probabilities in our HMM allow the estimated pen trajectory to start/terminate at any position, resolving the problem of the *starting/terminating positions*. This is a direct result of our first-order HMMs developed in Section 4.2.
2. *Turning points* are dealt with by specifying appropriate transition probabilities, and no restrictive assumptions are needed, as described in Section 4.5.
3. *Elasticity* is included in the HMM topology so that dynamic exemplars and static scripts with different numbers of samples are comparable, as described in Section 4.4. Corresponding segments are typically allowed to differ with a scale factor of two.
4. The observation PDFs, associated with the states in our HMMs, enable the quantification of similarities between static scripts and dynamic exemplars. Furthermore, the PDF parameters enable us to model the *geometric variations* in different pre-recorded dynamic exemplars. The PDF parameters that are included to model positional variations are described in Section 4.2, whereas the PDF parameters to model directional variations are described in Section 4.6.
5. We are able to model a collection of single-path trajectories constituting a static script, i.e., we are able to deal with *multi-path static scripts*, as described in Sections 5.1 and 5.2.
6. When the ink is not evenly distributed over the pen-tip, it may cause *spurious disconnections* in static scripts. In practice, this problem occurs frequently. Our HMM topology enables us to deal with such spurious disconnections, as shown in Section 5.3.
7. We have mentioned in the previous section that many techniques are limited due to local optimisation. We match a dynamic exemplar to our HMM using the Viterbi algorithm. Since the Viterbi algorithm is a *global optimisation* algorithm, it is particularly useful for resolving local ambiguities due to multiple intersections.
8. Section 5.4 shows that the Viterbi algorithm, the availability of many dynamic exemplars and some further calculations enable us to deal with the *sequence variations* in signatures.
9. Our HMM training schemes calculate a prior set of parameters particular to a specific individual. These parameters can be especially useful in a signature verification system as they are, in fact, biometric measurements of an individual. Section 6.3.5 shows that our system performs only slightly better using this training scheme, indicating that our HMM is rather robust to *allographic variations* in signatures.

- **The necessary preprocessing characteristics to contribute to a robust and accurate system.** The necessary preprocessing steps to enhance the performance of our trajectory estimation algorithm have been thoroughly investigated. Contributions regarding the preprocessing are the following:
  1. A *skeletonisation* algorithm that tends to enhance local line directions, enables us to identify simple crossings with confidence and that enables an accurate resampling scheme has been developed, as discussed in Section 3.1. Specifically, the necessary modifications to the existing techniques described in [86, 87, 69] are introduced, which can also be useful for general off-line handwriting application. In many existing techniques, a collection of skeleton points that must be traversed at least once is selected, making these approaches especially sensitive to artifacts and background noise. Our system has a remarkable robustness to skeleton artifacts, as shown in Section 6.3.2.
  2. The general orientations of static images and dynamic exemplars are aligned with a shape-matching algorithm in the Radon domain, as shown in Section 3.2. This *orientation normalisation* approach is more robust than the general Principle Component Analysis (PCA) approach, especially when aligning shapes with similar principle components, as shown in Sections 3.2 and 6.3.3. Despite the obvious benefits of the Radon-based rotation, there is not a substantial decrease in our system's performance when using PCA-based rotation, as shown in Section 6.3.3. This shows that our HMM contributes to a trajectory estimation algorithm that is robust to rotational variations.
  3. It is shown that the choice of a scheme to resample parametric curves plays an important role in the *accuracy* and *efficiency* of our system. Judicious resampling of parametric curves increases the speed of our system substantially without a significant performance degradation, as shown in Section 6.3.2.
- **Quantifiable results.** Objective methods evaluating the efficacy of estimated pen trajectories are sparse; see Chapter 2. We have developed a sensible evaluation protocol that is applicable to a wide range of pen trajectory estimation algorithms. The evaluation protocol is *straightforward* to implement and *invariant* to parameterisation.
- **Published work.** The sections in this dissertation that describe how to estimate the pen trajectories of single-path static scripts (including the the necessary preprocessing and quantitative results) were condensed into a journal paper. The paper was peer-reviewed and accepted for publication in a journal that specifically publishes work that contributes to the field of pattern recognition [58]. The sections in this dissertation that describes the extensions of the techniques in [58] to multi-path static scripts were condensed into a conference paper. The conference paper was peer-reviewed and accepted for publication in conference proceedings focussing on work that contributes to the field of document



analysis and recognition [57].



# Chapter 2

## Literature study

This chapter documents related literature relevant to the research topic. We focus on prominent studies that estimate the pen trajectories of static scripts. In this dissertation, these studies are divided into rule-based methods, graph-theoretical methods and methods that search for an optimal local correspondence between a static script and a dynamic exemplar. In Chapter 1 we have mentioned that explicit restrictions occur in several existing approaches for the sake of simplification. Section 2.1 provides more detail of these restrictions. In Section 2.2-2.4 each existing system is discussed with attention to the following matters:

- The feature measurement scheme of each system is discussed, i.e., it is investigated how the system under consideration presents a static script.
- Each system's approach to estimating the pen trajectory of a static script is described.
- The database, evaluation protocol and experimental results of each system are reported.
- Chapter 1 has shown that approaches that utilise pre-recorded dynamic exemplars must be especially comprehensive of variations in the dynamic exemplars. Hence, where applicable, notice is taken of a system's performance in this regard.

The discussion on existing approaches is summarised in Section 2.5, where some pertinent conclusions are drawn.

### 2.1 Restrictions

Several existing techniques impose restrictions when estimating the pen trajectories of static scripts for the sake of simplification. As mentioned in Chapter 1, it is important to construct a system that can handle a wide range of static scripts. Restrictions typically restrict the system to

a limited set of scripts, e.g., only characters or cursive words that are straightforward to unravel. The restrictions applicable to existing approaches have been identified and listed. The most common of the listed restrictions, which explicitly occur at some stage in existing algorithms, are categorised as follows:

1. **Starting/terminating positions:** The positions where a single-path trajectory can start (where a pen-down event occurs) or terminate (where a pen-up event occurs), are typically restricted as follows:
  - (a) *Left-to-right assumption:* It is assumed that a static script has been generated by an individual from a specific population, where cursive handwriting proceeds in a top-to-bottom-left-to-right fashion.
  - (b) *End of line assumption:* It is assumed that the starting and terminating positions of a single-path trajectory occur at the end of a line, where traversal can proceed in only one direction.
2. **Intersections:** Section 1.2 has shown that static scripts that contain regions where many lines cross one another in close proximity can be problematic to unravel. Typical restrictions at intersections are:
  - (a) *Local smoothness constraints:* Some methods introduce a local smoothness constraint at intersections, compelling lines that enter an intersection to exit it with approximately the same orientation. Inevitably, this constraint impels local choices at intersections.
  - (b) *Number of intersecting lines assumption:* It is assumed that a maximum of two lines can cross each other at an intersection.
3. **Turning points:** A turning point is defined as a high curvature point on a parametric curve, where the pen stops and reverses its direction. Due to the pen-tip width and digitising effects, it frequently happens that the curve that enters and the curve that exits the turning point are merged. The result is a single curve which must cope with bidirectional traversal. The degree of ambiguity increases even more if the pen revisits the merged curve. Simplifications to deal with ambiguities include:
  - (a) *No turning point assumption:* It is assumed that no segment in the static script can be traversed more than once, i.e., no turning points are allowed.
  - (b) *Double-traced lines assumption:* It is assumed that no segment in the static script can be traversed more than twice.
4. **Single-path static scripts:** Due to the difficulty of identifying pen-up and pen-down events when estimating a pen trajectory, some studies assume that a static script consists of only a single-path trajectory. Hence, pen-up and pen-down events other than the starting and terminating positions of a script cannot be identified.

A summary of all the related work mentioned in this chapter is presented in Table 2.1. The authors, years of publication, and appropriate references are presented. In the third to last columns it is indicated if the assumptions above (indicated by numbers) occur at some stage in the referenced work ( $\checkmark$ ), do not occur ( $\times$ ), or if there is not enough detail to make deductions ( $-$ ).

## 2.2 Rule-based methods

Rule-based methods are some of the earlier approaches used to estimate 1D sequences from 2D images. The first attempts to unravel static scripts tried to understand the temporal principles for generating handwriting. Various mathematical models have been developed to analyse or generate a piece of handwriting; see [62, 64]. *Bottom-up models* are, e.g., concerned with the analysis and synthesis of low-level neuromuscular processes involved in the motor-controlled actions to generate handwriting [62]. One can then model certain curves of a handwritten script as the result of the coactivation of two neuromuscular systems, one agonist and the other antagonist, which control the velocity of the pen-tip. Accordingly, an appropriate mathematical function is chosen to model velocity. Note, however, that measuring such neuromuscular processes and choosing appropriate models are highly dependent on the application and is definitely not trivial (these tasks are also dedicated subjects in the field of psychology, neurology, cognitive science, and graphology [62].)

In this field of study, it is already a difficult task to estimate dynamic information from static images. To calculate indicators of neuromuscular processes from 2D images is even more challenging. In general, it can be concluded that handwriting, especially signatures, is unpredictable, making it difficult to establish a robust set of heuristic rules that are able to mimic the underlying principles that control pen motions. Hence, several rule-based methods aim to estimate 1D observation sequences *consistently* rather than *precisely*, i.e., to extract consistent pseudo-dynamic information from a static script. Although some of these methods are severely restricted by the rules they impose, they provide a useful framework for other approaches. The most important heuristic from rule-based methods, which is also a crucial component of most of the relevant literature on this research topic, is based on continuous handwriting motion. Specifically, it is assumed that muscular movements constrain an individual's hand (holding the pen) to move continuously. Consequently, this *natural* motor-controlled movement leads to a general smoothness criterion, enforcing the pen to maintain its direction of traversal. This smoothness criterion enables one to follow lines through intersections. In the chapters to follow we refer to this criterion as the *continuity criterion of motor-controlled pen motions*.

Authors	Year	1. Start/End		2. Intersect		3. Turn		4. Single-path
		(a)	(b)	(a)	(b)	(a)	(b)	
<b>Rule-based methods</b>								
Lee and Pan [59, 54]	1991, 1992	√	×	√	×	×	√	×
Doermann and Rosenfeld [17, 19, 18]	1993, 1995	√	×	√	–	×	–	×
Boccignone et al. [9]	1993	√	√	√	×	√	√	×
Huang et al. [34]	1995	×	√	√	×	√	√	×
Lallican and Viard-Gaudin [44]	1997	–	–	√	×	√	√	×
Chang and Yan [11]	1999	√	√	√	×	√	√	×
Plamondon and Privitera [66, 63]	1995, 1999	√	√	√	×	×	–	×
Spagnolo et al. [78]	2004	–	–	–	–	–	–	–
<b>Graph-theoretical methods</b>								
Abuhaiba and Ahmed [2]	1993	×	×	×	×	×	√	√
Huang and Yasuhara [33]	1995	–	√	×	√	√	√	√
Allen and Navarro [5]	1997	√	×	√	×	×	√	√
Jäger [38, 37]	1997, 1998	×	×	×	×	×	×	√
Kato and Yasuhara [40, 41]	1999	√	√	√	√	×	√	×
	2000	√	√	√	√	×	√	√
Lallican et al. [43]	2000	√	–	×	×	×	×	×
Al-Ohali et al. [4, 3]	2002	×	√	×	×	×	√	√
Lau et al. [50, 51]	2002	√	√	×	×	–	–	×
	2003	×	√	√	×	–	–	×
Qiao and Yasuhara [67]	2004	–	√	√	×	×	√	√
<b>Local correspondence methods</b>								
Guo et al. [31, 30]	2000, 2001	×	×	√	×	×	×	×

**Table 2.1:** A summary of related work. The authors, years of publication, and appropriate references are presented. In the third to last columns it is indicated if the numbered assumptions of Section 2.1 occur explicitly at some stage in the referenced work (√), do not occur (×), or if there is not enough detail to make deductions (–).

Lee and Pan [59, 54] estimate the pen trajectories of static signature skeletons by using a set of heuristic rules that mimic the writing process of English-speaking, right-handed individuals. Experiments were conducted on 20 static signatures. Subjective evaluation indicated that the invoked rules cannot cope with characters such as “a”, “d”, and “g”, where counterclockwise circular drawing movements occur.

Doermann and Rosenfeld [17, 19, 18] extract a taxonomy of local and global clues from grey-scale images of handwriting that is useful for recovering temporal information from the images. A typical local clue is, e.g., the grey-scale intensity at the end of a line: the authors observe that, for ball-point pens, the intensity is typically significantly lighter than the rest of the script if a pen-down event occurs. The recovered clues are weighted according to their reliability. The weighted clues are then used to estimate the order of certain segments. Hence, temporal information is not recovered in ambiguous parts where no reliable clues are available. Their experiments intend to determine to what extent the recovered clues can be used to deduce the mechanics of the writing instrument and knowledge of the writing process. Subjective evaluation of 1000 handwritten static scripts from U. S. mail pieces indicated that over 90% of the scripts contain clues that can be used to recover temporal information.

Boccignone et al. [9] consider the direction, width and length of curves in a continuity criterion to segment the skeletons of static scripts. During the segmentation process each curve that enters an intersection is either merged with another curve and detached from the rest of the image, or just detached without merging. The calculated segments are then traced according to the invoked heuristics. Experiments were conducted on 10 000 handwritten characters by 20 writers, consisting of uppercase and lowercase letters as well as numerals. Human observers were consulted to determine if the system made the right choices at intersection regions (yes or no.) Accordingly, the system performed with an average accuracy of 97%.

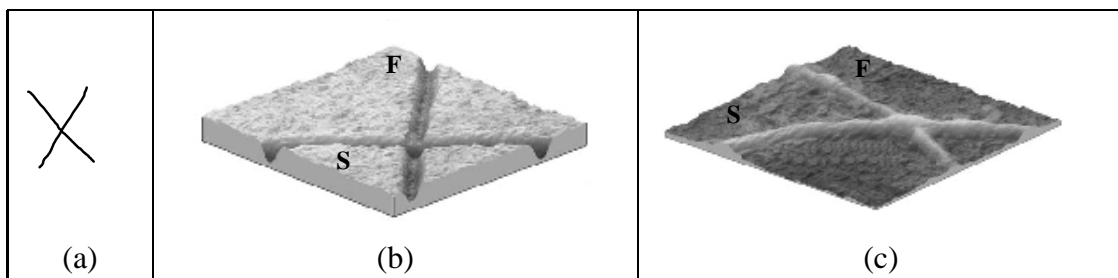
Chang and Yan [11] and Huang et al. [34] partially trace the skeletons of static scripts as part of a segmentation process. Huang et al. [34] divide static signatures into easily traceable (reliable) and ambiguous (unreliable) skeletons. Parametric curves are then extracted from the reliable skeletons by using a local continuity criterion. The authors note that many lines crossing at an intersection cause severe problems. Chang and Yan [11] subdivide Chinese characters into collections of parametric curves. (Note that the substructures of Chinese characters are sometimes referred to as *radicals*.) For each character, the general positions and directions of the separated curves are matched to calculate a further segmentation, i.e., some parametric curves are merged and some are subdivided into smaller segments to compute a new collection of parametric curves. A set of direction rules is then employed to calculate the time sequence of the points that constitute the parametric curves.

Lallican and Viard-Gaudin [44] divide grey-scale images of static scripts into sets of sepa-

rated segments. The pen trajectories of the grey-scale segments are then calculated using a Kalman filter that chiefly detects curvature information. Assorted trajectories, as computed by the Kalman filter, are then merged in accordance with a global cost function to compute the final, smoothest trajectory.

Plamondon and Privitera [63] identify high curvature points and intersection regions in images of handwritten words by employing information available from image contours. Prior rules that mimic the writing process of the Latin alphabet by right-handed individuals, as well as the identified intersection regions and high curvature points are used in conjunction to estimate the pen trajectories of the words while tracing the contours of the words. Results were tested on 200 city names for 6 individuals, containing 1390 intersection regions. Ten human observers were consulted to determine if the system made the right choices at intersection regions (yes or no.) Accordingly, the system performed with an average accuracy of 94%. Additionally, the authors also determined subjectively that the system recovered 89% of the original pen-tip movements.

Spagnolo et al. [78] use a novel opto-electronic device and 3D reconstruction techniques to perform 3D acquisition of documents. Preliminary results show that their 3D presentations of static scripts contain information that is unavailable in conventional 2D presentations. The additional information is especially useful for pen pressure analysis, thereby making it easier to unravel superimposed curves. Figure 2.1(a) shows a typical example of a binarised 2D “x” pattern. Note that there are no clues, whatsoever, to determine the sequence in which the two intersecting curves have been created. However, Figures 2.1(b) and (c) show the 3D presentations from [78] of an “x” pattern viewed from above and below, respectively. Spagnolo et al. [78] observe that the z-coordinates of the curves in the xyz-plane, and the colour intensities of the 3D presentations can be used, among other local clues, to infer which curve was written first (F) and second (S). It is shown in Section 7.2.2 that the pressure information inferred from such 3D presentations of static scripts may complement our approach.



**Figure 2.1:** Presenting an “x” pattern as a 2D image and in a 3D space (taken from [78]), with (a) the 2D image, (b) the 3D presentation viewed from above and (c) the 3D presentation viewed from below. Pressure information is inferred from the 3D presentations, which is used, among other local clues, to detach the two superimposed curves and determine which curve was written first (F) and second (S).

## 2.3 Graph-theoretical methods

The excess of recent literature on the research topic relies on graph-theoretical methods. A brief background of graph theory is provided in Section 2.3.1. Further information regarding graph theory can be found in [72, 38, 55].

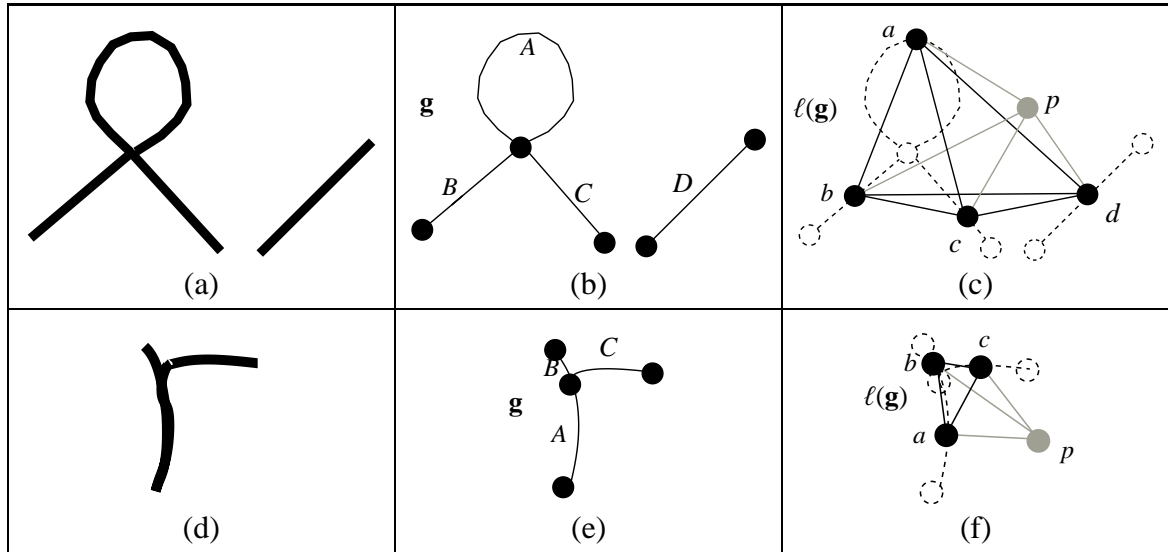
### 2.3.1 Graph-theoretical background

A graph  $\mathbf{g} = (\mathbf{v}, \mathbf{e})$  is a collection of *nodes/vertices*  $\mathbf{v} = \{v_0, \dots, v_n\}$  and edges  $\mathbf{e} = \{e_1, \dots, e_n\}$ , where an *edge*  $e_n$  is a line that connects two control points, in this case the two vertices  $v_{n-1}$  and  $v_n$ . A *directed* graph consists of *ordered* pairs of vertices, whereas an *undirected* graph consists of *unordered* pairs of vertices. Graph-theoretical approaches to this research topic construct undirected graphs from static scripts. In skeleton-based methods, e.g., the nodes typically label the skeleton samples that are connected to one, or more than two skeleton samples. A *cost/weight* is then assigned to each edge to produce a *weighted graph*. In a *complete graph* all the nodes are connected. The nodes in a *line graph*  $\ell(\mathbf{g})$  correspond to the edges of  $\mathbf{g}$  so that two nodes are adjacent in  $\ell(\mathbf{g})$  if the corresponding edges in  $\mathbf{g}$  are adjacent.

General graph-theoretical concepts are illustrated in Figure 2.2. A static script, which is easy to unravel, is shown in Figure 2.2(a). Figure 2.2(b) depicts a typical graph  $\mathbf{g}$  for the skeleton of the static script in Figure 2.2(a), where  $\mathbf{g}$  consists of four edges  $\{A, B, C, D\}$  and five nodes (filled dots.) The complete line graph  $\ell(\mathbf{g})$  of Figure 2.2(b) is the collection of black solid lines and black filled dots in Figure 2.2(c). A *path*  $\mathbf{p} = [v_0, e_1, v_1, \dots, v_{n-1}, e_n, v_n]$  in a graph is an alternating sequence of nodes and edges, beginning and ending with nodes. It joins nodes  $v_0$  and  $v_n$ , passes through the nodes  $\mathbf{v} = [v_0, v_1, \dots, v_{n-1}, v_n]$  and traverses the edges  $\mathbf{e} = [e_0, e_1, \dots, e_{n-1}, e_n]$ . A path is *elementary* if all the edges are distinct, whereas it is *simple* if all the nodes are distinct. A *cycle* is a path in a graph with identical start/end nodes, i.e.,  $v_0 = v_n$ . A *tree* is a graph that contains no cycles. A *spanning tree* of  $\mathbf{g}$  is a subgraph of  $\mathbf{g}$  that contains all the vertices but only enough of the edges to form a tree. Thus, in Figure 2.2(b), the set of edges  $\{B, C, D\}$  and the nodes that are connected to these edges form a spanning tree of  $\mathbf{g}$ .

An important graph-theoretical problem to address is to calculate the shortest path from a source vertex  $s \in \mathbf{v}$  to a destination vertex  $d \in \mathbf{v}$ . In a weighted graph, this path corresponds to the path  $s \rightarrow d$  with the smallest total weight. This problem is called the *single-pair shortest-path* problem; see [32]. For any graph, there is a number of variants to this problem. In the *single-source shortest-path* problem, the shortest path from a single source  $s \in \mathbf{v}$  to *every* other vertex  $v \in \mathbf{v}$  must be calculated. This problem can be solved using Dijkstra's algorithm with





**Figure 2.2:** Illustration of graph-theoretical terminology. (a) A static script with (b) a typical graph  $g$  for the script's skeleton. (c) Black solid lines and filled dots render the complete line graph  $\ell(g)$  of (b). A “virtual” node  $p$  is included to search for an optimum Hamilton cycle. (d) A static “r” with (e) the graph of its skeleton. (f) The complete line graph of (e) which is connected to a “virtual” node  $p$ .

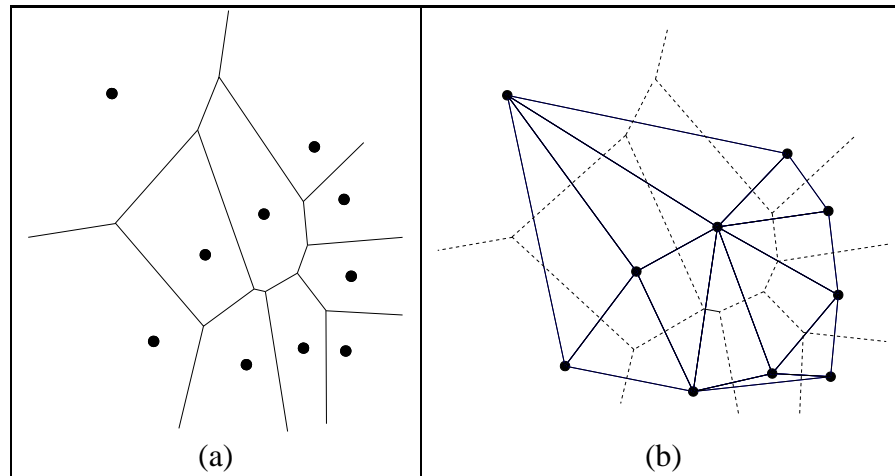
complexity  $O(N^3)$ , where  $N$  is the number of nodes [72, 32, 38]. In the *all-pairs shortest-path* problem the shortest path between every pair of vertices is calculated. Although this problem can be solved by solving the single-source shortest-path problem on every vertex, there are more efficient ways, e.g., in [72] a dynamic-programming approach is described. In the context of the research topic, the path that minimises a predefined cost function (depending on the application) maps to the pen trajectory of the static script. The well-known *Chinese postman problem* is the search for a Eulerian cycle in a graph, where the *Eulerian path* is the shortest path containing every edge in the graph exactly once and the *Eulerian cycle* is the Eulerian path with identical start/end nodes. However, not all graphs have Eulerian paths. To solve the Chinese postman problem, some edges may have to be duplicated so that the graph has a Eulerian path. Edges are, however, duplicated at most once, so that a segment in a static script can be traversed at most twice in approaches that solve the Chinese postman problem.

The *Hamilton path* is a path that contains every node of a given graph exactly once. The search for the shortest Hamilton cycle in a weighted complete graph is called the *travelling salesman problem* [38], where a *Hamilton cycle* is a Hamilton path with identical start/end nodes. To solve the travelling salesman problem, a “virtual node” or *pseudo-node*  $p \in \mathbf{v}$  is typically included in the complete line graph of  $g$ . The new graph is called the *extension* of the complete line graph of  $g$ . *Pseudo-edges* are then included to connect all the other nodes to  $p$ , as illustrated by the grey lines in Figure 2.2(c). The shortest Hamilton cycle is then calculated.

If the shortest Hamilton cycle corresponds, e.g., to the sequence of nodes  $[p, b, a, c, d, p]$  in Figure 2.2(c), the node  $p$  is removed to calculate the final sequence of nodes. The shortest Hamilton cycle may yield discontinuous paths. In such cases, the shortest continuous path to bridge the discontinuity is typically calculated. In Figure 2.2(d) a static script of the character “r” is shown. The graphs  $\mathbf{g}$  and  $\ell(\mathbf{g})$  for the skeleton of (d) are shown in Figures 2.2(e) and (f), respectively. A realistic Hamilton cycle from Figure 2.2(f) is  $[p, b, a, c, p]$ . This sequence of nodes then translates to the sequence of edges  $[B, A, C]$  in Figure 2.2(e). The shortest path to continuously reach edge  $C$  from  $A$  in Figure 2.2(e), after  $A$  has already been traversed from edge  $B$ , is to retrace edge  $A$ . Hence, the final sequence of edges in Figure 2.2(e) is  $[B, A, A, C]$ . In travelling salesman approaches, similar extensions are made for lines that are traversed more than twice.

The travelling salesman problem and Chinese postman problem are *proximity problems*, i.e., these problems can be reduced to geometric problems that deal with the proximity of points in metric space [55]. A powerful approach to deal with these problems effectively is to utilise a *Voronoi diagram* and its straight-line dual, the *Delaunay triangulation*. The Voronoi diagram  $V$  of a set of points  $P = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_N\}$  is a uniquely defined decomposition or tessellation of the plane into a set of  $N$  polygonal cells, referred to as Voronoi polygons [73]. Each polygon contains exactly one sample  $\mathbf{p}_i \in P$  and delineates the locus of all points in that plane that are closer to  $\mathbf{p}_i$  than any other point in  $P$ . The edges defining a Voronoi polygon are generated from the intersections of perpendicular bisectors of the line segments connecting any one point to all its nearest neighbours in  $P$  [73]. A typical Voronoi diagram  $V$  of a set of points (black dots) is shown in Figure 2.3(a). The Delaunay triangulation can now be computed from  $V$  by connecting all pairs of points that share the same edge [55]. The Delaunay triangulation of Figure 2.3(a) is shown in Figure 2.3(b). It is shown in Section 3.1 how a Delaunay triangulation can be derived from a static script to calculate the skeleton of the script from the computed tessellation.

A few points must be considered when deciding whether the Chinese postman or the travelling salesman problem is applicable to a problem. That is, one must determine when to calculate the shortest path directly from  $\mathbf{g}$  or when to compute extension of the complete line graph of  $\mathbf{g}$  and then calculate the shortest path from  $\ell(\mathbf{g})$ . Both solutions calculate an optimal trajectory, based on global optimisation. Global optimisation is especially useful to resolve ambiguous intersections in static scripts. A wider range of scripts can be dealt with when solving the travelling salesman as opposed to the Chinese postman problem, e.g., lines that are traversed more than twice can, by definition, be identified for the travelling salesman problem, which is not the case for the Chinese postman problem. Unfortunately, the computational cost of the Chinese postman and travelling salesman problems have to be accounted for. Solutions to problems are regarded as *efficient* if they can be solved in polynomial time, i.e., the number



**Figure 2.3:** Computing (a) the Voronoi diagram and (b) the Delaunay triangulation (straight-line dual of (a)) of a set of points (filled dots.)

of operations is proportional to some polynomial in the number of input bits [72]. As graph-theoretical methods based on the travelling salesman problem and certain cases of the Chinese postman problem are NP-complete, these methods belong to a family of problems for which no efficient solution can be found [72, 38]. Since a complete graph of a static script has to be constructed to solve the travelling salesman problem, the travelling salesman problem is in general computationally more expensive than the Chinese postman problem. Generally, when solving the travelling salesman problem, all the permutations of the vertices in a weighted complete graph must be calculated. Hence, in the worst case  $O(N!)$  operations are required, where  $N$  is the number of edges in the graph [72]. There are, however, good approximations of efficient solutions to all cases of the Chinese postman problem; see [38]. Thus, one often has to rely on sub-optimal solutions to the travelling salesman problem, or, by imposing more restrictions, revert to the Chinese postman problem and approximations of it. When solving the travelling salesman problem for this application, one can also introduce heuristic constraints to reduce the computational complexity and thereby ensure that the algorithm is not NP-complete. This is, however, at the cost of a lower accuracy.

### 2.3.2 Trajectory estimation algorithms that solve the Chinese postman and travelling salesman problems

Methods that rely on solving either the Chinese postman or travelling salesman problem differ primarily as follows:

1. Different cost functions are minimised to find the shortest path in the graph that presents a static script.

2. Different processing steps are applied to the graphs before the search for the shortest path is conducted, e.g., in some methods the graphs are divided into smaller sub-graphs in order to reduce the computational complexity.
3. Graphs are derived from the skeleton or grey-scale image of the static script.

Attention is afforded to the above differences in the discussion of the graph-theoretical methods that follows, as these differences influence a method's flexibility, computational complexity and sensitivity to artifacts.

The methods in [2, 33, 5] are some of the first approaches to estimate the pen trajectories of static script skeletons by solving the Chinese postman problem. Abuhaiba and Ahmed [2] introduce a prior set of heuristic rules that comply with Arabic handwriting, e.g., they search for a starting position at the right side of the script, as Arabic handwriting proceeds from right to left. After calculating the starting position of the pen trajectory, the rest of the trajectory corresponds to the path with the shortest Euclidean distance. 1605 Arabic scripts by two writers were traced, and a subjective evaluation indicated that 92% of the actual temporal information was recovered. Huang and Yasuhara [33] subdivide the skeleton graph into smaller sub-graphs to reduce the computational complexity, and introduce a cost function which is minimised to find the smoothest path in the graph. Allen and Navarro [5] apply a local continuity criterion to merge certain edges in the graphs presenting the skeletons of Roman characters. The Eulerian paths with minimum Euclidean distances are then calculated.

Jäger [38, 37] constructs graphs from the skeletons of static scripts and estimates pen trajectories from the graphs by using two systems. The first system finds the path with minimum Euclidean distance by solving the Chinese postman problem. The second system computes the angles between intersecting edges and solves the travelling salesman problem in order to minimise curvature globally. In [38], 6934 on-line words by 88 German students were converted into off-line words to generate results, where the average length of the ground-truth trajectories is 17.7 (expressed as the number of edges in  $\mathbf{g}$ .) It should be noted that edges do not necessarily have equal pixel lengths (nodes in  $\mathbf{g}$  are constructed for only certain skeleton samples, as illustrated in Figure 2.2(b).) *Levenshtein distances* are used to calculate error rates; see Section 6.4 for more detail. The second system, which solves the travelling salesman problem, performs best with an average Levenshtein distance of approximately 3.9. If the graph of a word consists of too many edges, i.e., when the computational complexity is too high, the word is segmented. The optimal trajectories from the separate graphs of the segments are then combined to calculate the final trajectory of a static script.

Kato and Yasuhara [40] construct a graph from the skeleton of a single-path static script and label the kind of vertices and edges in the graph, e.g., heuristic measures are invoked to identify

and label the edges that are allowed to be traversed twice. The pen trajectory of the static script is then estimated from the graph by combining the label information and the search for the shortest Eulerian path. Although restrictions are imposed on the starting and terminating positions of a pen trajectory, the authors are able to identify lines that have been traversed twice. Hence, in this regard, they improve on techniques that are unable to identify retraced lines, e.g. [33]. Although their method is not as flexible as methods that solve the travelling salesman problem, e.g. [38], not all the possible paths are enumerated, thereby making their approach computationally more effective. A subjective evaluation was conducted on more than 100 static handwritten scripts, and it has been observed by the authors that their method is successful on scripts that comply with their assumptions. In [41], the same authors extend their approach to multi-path static scripts.

Lallican et al. [43] segment the grey-scale images of static scripts at critical points, e.g., high curvature and intersection points. A graph is then calculated with nodes that correspond to the calculated critical points. The position and direction between edges in the graph are minimised by solving the travelling salesman problem. The authors note that, compared to Jäger [38, 37], their graphs are refined so that they are able to account more accurately for lines that are revisited multiple times, as well as pen-up and pen-down events. Pen trajectories were estimated from characters and words in the IRONOFF database, where the IRONOFF database consists of on-line and off-line versions of 30 000 words and 25 000 characters for 700 individuals (see [45] for more detail.) The estimated pen trajectories were implemented in a character and word recogniser. Approximately 80% of the estimated trajectories produce likelihoods in the same range as their dynamic counterparts when words and characters are classified. Hence, the authors infer that approximately 80% of the estimated pen trajectories are correct.

Al-Ohali et al. [4, 3] transform the graph  $\mathbf{g}$  of a static script's skeleton into a spanning tree  $\mathbf{t}$  by removing all the cycles from  $\mathbf{g}$ . Some edges in  $\mathbf{t}$  are allowed to be retraced when calculating the path with the shortest Euclidean distance. The removed cycles are reinstated after the calculation of the shortest path in  $\mathbf{t}$ . The algorithm is specialised for Arabic letters and words. In [3], estimated trajectories are applied to recognise Arabic words, where a recognition system has been trained on 19 813 samples and tested on 8172 samples. All samples were extracted from real-world bank cheques. The authors observe that 40% of the classifier errors are due to noise on the cheques, digitising effects, and scripts that are hard to unravel. A further 8% of the classifier errors are due to skeleton artifacts. (Other errors are caused by the classification module.)

Qiao and Yasuhara [67] identify lines that are allowed to be traversed twice in the skeleton of a static script using a probabilistic approach. Accordingly, they duplicate, merge and separate certain edges in the graph that presents the skeleton. The altered graph is subsequently divided

into smaller sub-graphs to improve the efficiency of the search for an optimal path. The same cost function as in [33] is minimised during the search for the smoothest path. The authors observe that their approach is more efficient and flexible than the methods in [40, 33].

The approaches of Lau et al. [50, 51] are similar to methods that solve the Chinese postman problem. Curves are extracted from the skeleton of a script. In [50], the extracted curves are matched to empirically determined cost functions, which are minimised to find an optimal path using dynamic programming. In [51], the cost functions are expressed as PDFs which are trained from a set of on-line handwritten scripts, i.e., they utilise dynamic exemplars in this regard. Training makes the system in [51] more robust to allographic variations than the system in [50]. Note, e.g., in Table 2.1 that the *left-to-right assumption* occurs in [50] and not in [51]. The authors note that, regarding geometric variations, their PDFs are scale and translation invariant, but not rotationally invariant. No quantitative results are provided.

## 2.4 Local correspondences with dynamic exemplars

Guo et al. [31, 30] estimate the pen trajectories of static signatures to detect forgeries in an off-line signature verification system. Dynamic exemplars are recorded using a digital mouse. A search algorithm is directed that finds a corresponding point in the grey-scale image of a static script for each point in the dynamic exemplar. For each point in the dynamic exemplar, the corresponding point in the static image must have a grey-scale intensity above a certain threshold, it must be nearby and within an angular sector of  $\pm 45^\circ$ . The search for the most likely pen trajectory is conducted up to a certain depth, where the search is terminated and started again. The authors note that this local optimisation causes errors as well as local choices at noisy, ambiguous intersections. It is also indicated that, regarding geometric variations, their system is not scale invariant. It is, however, rotationally invariant, within an angular sector of  $\pm 45^\circ$ . No explicit provision is made for sequence or allographic variations, i.e., no training schemes are invoked. No quantitative results are provided.

## 2.5 Summary

Some characteristics of existing techniques are identified that are comparable to our approach and that can be used to evaluate the performance of pen trajectory estimation algorithms. These characteristics are summarised as follows:

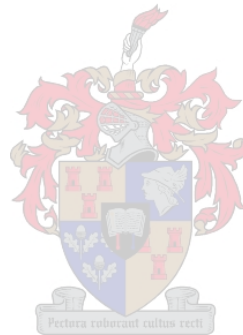
1. It is evident from Table 2.1 that rules are typically required to find the starting and ter-

minating positions of a static script in methods that do not make use of prior knowledge. Guo et al. [31] and Lau et al. [51], e.g., do not impose rules to calculate these positions, but invoke prior information available from dynamic exemplars. Likewise, additional information from dynamic exemplars, or prior rules, are typically required to identify pen-up and pen-down events, i.e., to deal with multi-path static scripts.

2. A comparison between graph-theoretical and rule-based methods indicate that local choices at intersections are frequently invoked in rule-based methods, whereas graph-theoretical methods evade local choices by minimising predefined cost functions globally. Hence, graph-theoretical methods generally have a better ability to resolve ambiguities in static scripts than rule-based methods have. The most flexible graph-theoretical methods are employed by Jäger [38] and Lallican et al. [43], where the travelling salesman problem is solved. However, no efficient solution exists to solve the travelling salesman problem. Hence, many graph-theoretical techniques are impelled to revert to sub-optimal or more restrictive solutions.
3. Table 2.1 indicates that techniques that can explicitly deal with lines that must be traversed more than twice are sparse. Hence, most techniques deal with handwritten words, letters or numerals which, in general, do not contain lines that are traversed more than twice. Few techniques unravel static signatures, which are usually much more unpredictable and difficult to unravel.
4. Only Jäger [38] and Lau et al. [52] propose methods to quantify the efficacy of an estimated pen trajectory. Although the method in [38] is not invariant to parameterisation, it can be applied to a wide range of scripts. The method proposed in [52] is only applicable to a restricted set of scripts, as discussed in Section 6.1.1.
5. In general, methods that extract all the skeleton samples of a static script are more sensitive to noise than methods that extract only a selection of skeleton samples, e.g., Guo et al. [31]. If a scanned-in document contains noticeable spurious lines near the handwritten script, due to external noise, errors are typically introduced if the system enforces traversal of all lines. Al-Ohali et al. [3], e.g., take notice of the errors introduced in their classifier due to external noise and skeletonisation artifacts.
6. Recent work by Lau et al. [51] and Guo et al. [31] rely on prior knowledge from dynamic exemplars. Guo et al. establish a local correspondence between a static image and dynamic exemplar. Lau et al. do not establish such a local correspondence but compute cost functions from dynamic exemplars which are used to compute the path with minimum cost in the skeleton of a static script. The following remarks can be made regarding existing systems that employ prior dynamic information:
  - (a) Even when prior dynamic exemplars are available, heuristic measures are typically required to resolve ambiguities completely. Lau et al. [51] and Guo et al. [31], e.g., invoke a local smoothness criterion at intersections.

- (b) Existing methods that utilise prior dynamic information are not completely robust against geometric variations; the approach followed by Lau et al. [51] is not rotationally invariant while the approach followed by Guo et al. [31] is not scale invariant.
- (c) The approach followed by Lau et al. [51] train PDFs to include writer-specific information, and is therefore more robust to allographic variations than the approach followed by Guo et al. [31].
- (d) Sequence variations do not explicitly influence the method of Lau et al. [51], whereas the method of Guo et al. [31] is highly sensitive to substitutions and deletions (corresponding points in the static image must be found for all points in the dynamic exemplar.)

The above characteristics are taken into consideration in Section 6.4, where the performance of our system is evaluated in comparison with existing techniques.





# Chapter 3

## Preprocessing

Some basic preprocessing steps are applied before estimating the pen trajectory of a static script. To estimate a pen trajectory, a static script is compared with a dynamic exemplar. Hence, any prior alignment of a dynamic exemplar and a static script assists our HMM to establish an accurate match and therefore improves the performance of our trajectory estimation algorithm. The most significant preprocessing steps are:

1. A dynamic exemplar is presented as a smooth parametric curve. Accordingly, the skeleton of a static script is computed by reducing the script to a collection of parametric curves. Section 3.1 pursues an adequate skeletonisation scheme.
2. The next step renders prior preprocessing to make our approach invariant to geometric variations. First, the centroids of a dynamic exemplar and a static skeleton are aligned to ensure translation invariance. Next, the dynamic exemplar is scaled so that it has the same standard deviation in the xy-plane as the static skeleton [53]. The orientations of the dynamic exemplar and static image are aligned using a Radon-based approach, as described in Section 3.2.
3. Finally, the parametric curves that constitute the dynamic exemplars and static skeletons are similarly resampled, as described in Section 3.3, in order to obtain a more efficient representation.

## 3.1 Skeletonisation

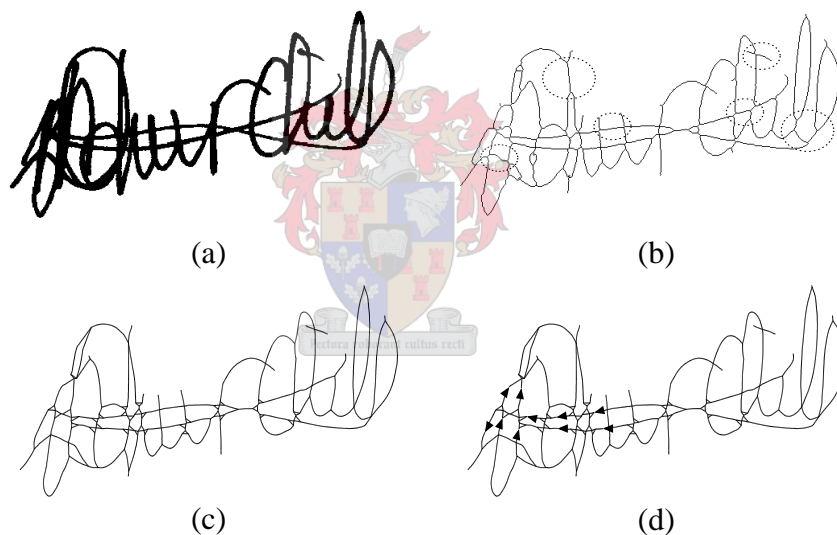
### 3.1.1 Introduction

Since static signatures appear as 2D images on documents, little, if any, dynamic information about the actual process of creating the signature is retained. In order to extract a parametric curve from a static image, we first extract a *skeleton* from the image through a thinning process, where the skeleton follows the centreline of the original image. Note that *skeletonisation* algorithms compute the centrelines from image boundaries, whereas *thinning* algorithms remove outer layers of an image while simultaneously preserving the image connectivity. In general, a strict requirement of good skeletonisation is the preservation of the topological and geometrical properties of the original object. There is a vast collection of existing literature available on thinning and skeletonisation techniques. Some of these techniques are discussed in Section 3.1.2. It is, however, important to bear the following in mind regarding the skeletonisation for this application:

- Many existing techniques that estimate the pen trajectories of static scripts from the script skeletons are sensitive to skeletonisation artifacts. It is therefore necessary to determine our HMM's robustness to artifacts. This section shows how artifacts are removed, whereas the effect of artifact removal is measured in Section 6.3.2.
- The local directions and the positions of the lines that constitute the skeletons of static scripts are embedded in our HMM PDFs, as shown in Chapter 4. The embedded script characteristics are then matched to the dynamic exemplars. To facilitate an accurate match, it is important to skeletonise static scripts so that they resemble their dynamic counterparts as accurately as possible. The efficacy of our algorithm in this regard is qualitatively measured in Section 3.1.6, whereas quantitative performance measures are presented in Section 6.3.2.
- For this application, it is important to distinguish between parts in a handwritten static script that are easy and parts that are difficult to unravel. Section 3.1.5 develops heuristics to establish such distinctions.
- Section 4.5 shows that smooth and accurate line directions allow the simplification of our HMM for intersections that are easy to unravel. We therefore focus on various smoothing techniques during the development of our skeletonisation algorithm in Sections 3.1.3-3.1.5.
- A skeleton that segments a handwritten script into smooth parametric curves complements our resampling scheme that identifies high curvature points; see Section 3.3 for details.

### 3.1.2 Literature synopsis

When an image is scanned as a grey-scale image and binarised, noise is inevitably introduced. If the skeleton is then derived from this binarised image, following the image centreline exactly, the skeleton is bound to contain artifacts. We refer to skeletons resulting from skeletonisation or thinning techniques that do not attempt to remove such artifacts as *standard skeletons*. Examples of such techniques can be found in [48, 28, 73, 29]. Artifacts are categorised as *peripheral artifacts*, such as spurs attached to the skeleton of an image and *intersection artifacts*, where two or more lines that should intersect fail to cross each other in a single point. Figure 3.1(a) shows an example of a reasonably difficult signature to skeletonise—one that even the eye finds difficult to unravel. Figure 3.1(b) is an example of a standard skeleton from [29] for Figure 3.1(a). Some, but not all, artifacts are encircled with dotted lines. Since we are attempting to extract the pen trajectory from the image, artifacts can affect our trajectory extraction algorithm—the exact effect of such artifacts is investigated in Chapter 6.



**Figure 3.1:** *The skeleton of a static signature containing regions of multiple crossings. (a) A binarised signature that is difficult to unravel. (b) Examples of artifacts that can occur in skeletons. (c) The final skeleton of (a), specific to our application (note the web-like structures.) (d) Examples of trajectories that can be extracted from (c).*

To remove skeletonisation artifacts and improve local line directions, so that static scripts closely resemble their dynamic counterparts, more sophisticated algorithms are required; see, e.g., [86, 87, 69, 80, 42, 13, 47, 79]. To choose an appropriate scheme is unfortunately rather difficult, as the quality of skeletonisation and thinning algorithms are mostly quantitatively measured by their computation time and their ability to preserve the topology and geometric properties of the original object; see [83, 46]. For this application, additional considerations should be taken into account, as outlined in Section 3.1.1. It is important, e.g., that the connectivity of

lines through complicated regions should be preserved.

For our application the rather sophisticated algorithm by Zou and Yan [86], as improved by Rocha [69] is highly suitable, with modifications specific to our application. Our skeletonisation scheme is primarily based on the algorithm by Zou and Yan [86], which is from now on referred to as the Zou-Yan algorithm. The Zou-Yan algorithm is based on *Delaunay triangulation* [14]—an angle-optimal triangulation given a set of points. Specifically, any Delaunay triangulation from a set of points maximises the minimum angle over all the constructed triangles; see [14] for further detail. The Zou-Yan algorithm first identifies the edges that represent the boundaries of the original image, where the edges, in this case, are lines connecting successive boundary samples. By constructing Delaunay triangles from the control points representing these edges and some further basic steps, one computes a skeleton that follows the centreline of the image [86]. Additionally, the triangles that comprise artifacts are identified, resulting in a powerful technique to identify and remove skeletonisation artifacts.

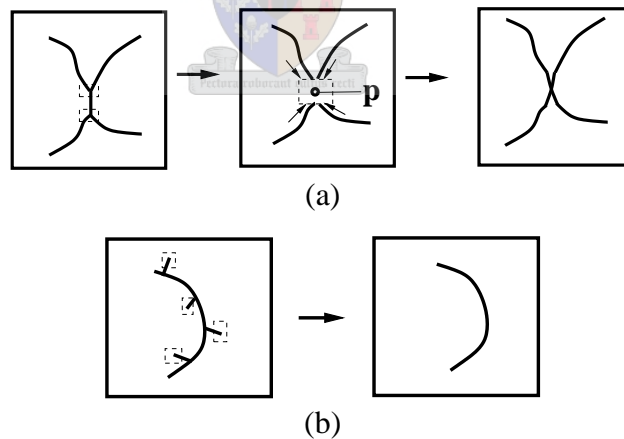
The most important modification for our application is with regard to the skeletonisation in complicated regions. The Zou-Yan and Rocha [69] algorithms assume that lines do not change their orientation after entering an intersection. Due to the nature of human handwriting, especially signatures, this is not always true. When an image becomes indistinct due to multiple thick-lined crossings in a small region, it is not clear which curves should be connected. If the skeletonisation algorithm follows a dominant curve and strives to maintain its direction, the wrong curves may be connected, with the result that actual trajectories become irretrievably lost. In situations like these, we are careful to maintain all possible connections, while smoothing transitions at intersections as much as possible. This often results in a visually unappealing web of connected lines (see Figure 3.1(c).) Although visually unappealing, these *web-like structures* are tailored for our proposed method. The HMM is able to find the appropriate connections, thereby reconstructing the pen trajectory accurately. Due to the web-like structures in complicated regions we do not necessarily preserve the topology of the original image. We therefore refer to our skeletonisation algorithm as a *pseudo skeletonisation* algorithm resulting in the *pseudo skeleton* of the original image.

### 3.1.3 Overview of our pseudo skeletonisation algorithm

This section presents a brief overview of the application-specific modifications to the Zou-Yan and Rocha approaches. The key idea of these approaches is to partition an image into smaller regions so that regional information can be exploited to identify artifacts. These regions assume a wide variety of shapes. *End regions* are defined as regions that contain skeleton lines between endpoints and crosspoints. An *endpoint* is a skeleton sample connected to only

one adjacent skeleton sample, whereas a *crosspoint* is a skeleton sample connected to more than two adjacent skeleton samples. Typically, end regions are likely to contain peripheral artifacts if they are short in length in comparison with their width. Spurious end regions are simply removed. *Intersection regions* contain crosspoints. Different crosspoints are joined in a single point by merging their intersection regions, thereby removing intersection artifacts, where *merging* describes the process that unites two or more intersection regions. Typically, the directions of skeleton lines that enter intersection regions are used as basis for calculating whether nearby intersection regions should be united.

Two simple examples are shown in Figure 3.2 to illustrate the basic steps for artifact removal. The first bounding box in Figure 3.2(a) depicts the skeleton of an image containing spurious intersection regions (dashed boxes.) Line directions (arrows) are used as basis for merging the two intersection regions. As mentioned above, the merging of the two intersection regions results in their unification, effectively expanding the separate intersection regions and removing the connected sub-shapes between them. Thus, the two regions are united into a single intersection region, as depicted by the big dashed box containing  $\mathbf{p}$  in the second bounding box of Figure 3.2(a). The lines that enter the new intersection region (dashed box) are joined at a crosspoint, where the crosspoint is the 2D skeleton sample  $\mathbf{p}$ . Figure 3.2(b) shows spurious end regions (dashed boxes) which are removed to compute the final skeleton, as shown in the last bounding box.



**Figure 3.2:** *Removal of skeleton artifacts. (a) Removing intersection artifacts by uniting the appropriate intersection regions (dashed boxes.) The directions (arrows) of the lines that enter the new intersection region are computed to calculate the crosspoint  $\mathbf{p}$  where the lines should join. (b) Peripheral artifacts are removed by removing spurious end regions (dashed boxes.)*

Problems are typically encountered in complicated areas where many intersection regions are located within close proximity, e.g. the left-hand side of Figure 3.1(a). In such cases, lines entering the intersection regions are too short to make accurate estimates of their directions. Inaccurate line direction estimates can result in the merging unrelated shapes, thereby remov-

ing skeleton lines that are not intersection artifacts, but important image features. The removal of such features can lead to a performance degradation of our HMM. It is therefore necessary to introduce further refinements to the basic algorithm. In short, one has to avoid merging of unrelated intersection regions. Accordingly, we introduce additional constraints, relying mainly on local line width. In the previous section, we alluded to the generation of additional lines in complicated parts forming web-like appearances. Since there is a direct relationship between the noise in the boundaries of an image and the number of artifacts in the image skeleton, a smoothing procedure is applied to the original boundaries as well as the final skeletons. Using the Zou-Yan algorithm as the basis for our pseudo skeletonisation algorithm, our implementation effects the following modifications for handwritten signatures:

- Image boundaries, lines that estimate the directions of connected sub-shapes and final skeletons are smoothed with appropriate smoothing techniques.
- Complicated parts of a static handwritten script, where it is difficult to estimate line directions, are identified and handled separately.
- Several constraints are set, based on line width.
- Iterative merging of intersection regions is prevented and the criteria of the Zou-Yan algorithm are extended to decide whether two intersection regions should be merged.

The details of the algorithm are described in subsequent sections of this chapter. Section 3.1.4 explains how an image is partitioned into sub-shapes and how a standard skeleton is derived from these sub-shapes. Section 3.1.5 explains how sub-shapes are manipulated to remove artifacts. Section 3.1.6 presents results and a summary of our algorithm, and some final conclusions are drawn in Section 3.1.7.

### 3.1.4 Shape partitioning

The first steps are straightforward: Image boundaries are extracted from static handwritten images. These boundaries comprise polygons so that the set of polygons, referred to as the *approximating polygon* of the static image, now represents the image. Since local line directions are not well defined as a result of noisy boundaries, boundary samples are processed as follows:

1. High curvature points are selected from the boundaries following the polygonalisation approach in [73].
2. The resulting boundaries are smoothed using a low-pass filter [28]. Excessive smoothing, however, will remove regions of high curvature. This can be particularly problematic for

thinner signatures, as excessive smoothing allows outer boundaries (surrounding the image) to cross inner boundaries (surrounding holes.) To overcome this problem, the mean  $\mu$  and standard deviation  $\sigma$  of line thickness are computed. After each smoothing iteration, the areas of the polygons enclosed by smooth boundaries are calculated as percentages of the original polygon areas. The area loss/gain  $a_{\text{polygon}}$  after each smoothing iteration is compared with the following empirically determined threshold  $a_{\text{min}}$ , where

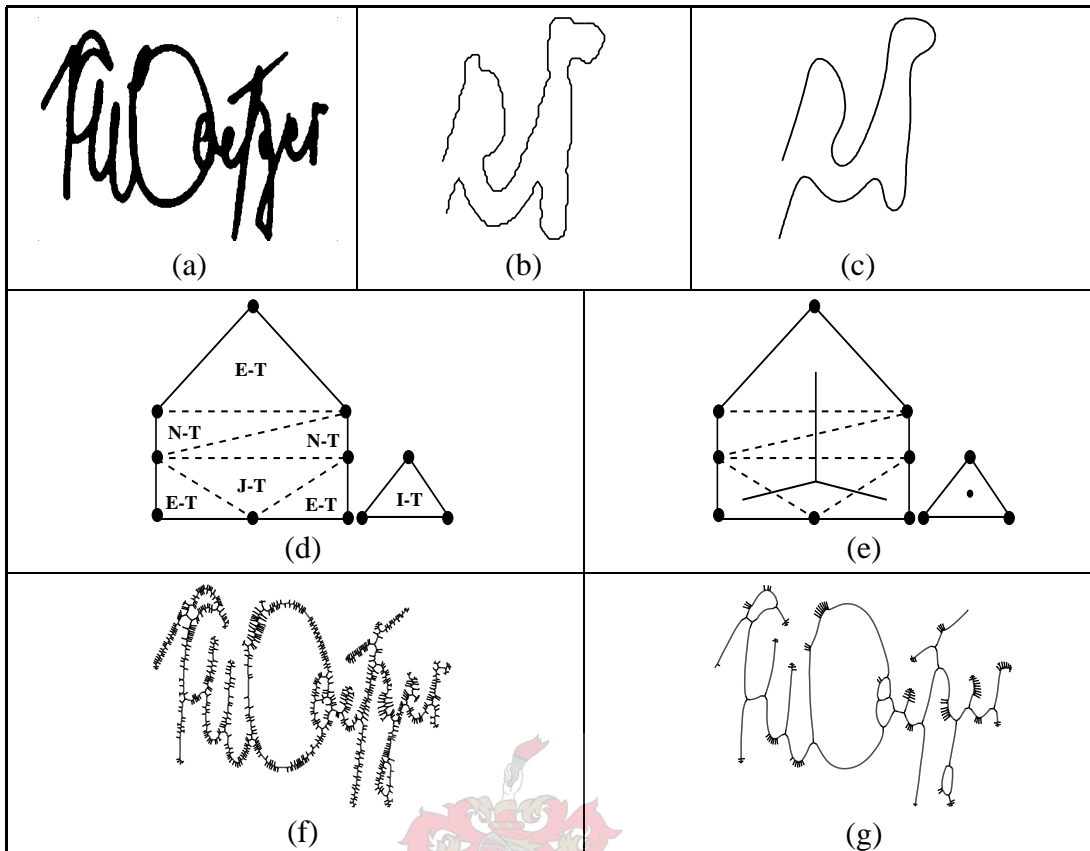
$$a_{\text{min}} = 1.5 * \mu^{\frac{1}{2}} \sigma^{\frac{1}{3}}. \quad (3.1)$$

Hence, if  $a_{\text{polygon}} > a_{\text{min}}$  the smoothing of the iteration is not included. According to (3.1), thick-lined signatures varying more in line width (an indication of noisy boundaries) are smoothed more than thin-lined signatures with less boundary noise.

3. Image boundaries that enclose three or less connected pixels are considered insignificant and removed immediately.
4. For the sake of simplicity, image boundaries are resampled so that the distance between any two successive samples is approximately one pixel.

Figure 3.3(a) shows a static signature to be skeletonised. Figure 3.3(b) shows a part of the signature's noisy boundary as extracted from Figure 3.3(a). Figure 3.3(c) illustrates the smoothing effect on the boundary of Figure 3.3(b) after processing the parametric curve that presents the boundary according to the steps above. The smoothed boundary samples are used as control points to divide the original shape into a set of non-overlapping triangles using Delaunay triangulation [86, 65, 14]; also see Section 2.3.1. In order to proceed we need to recall some concepts of [86, 87, 69]:

- *External triangles* occur because the Delaunay triangles are situated inside the the convex hull of the object. This can generate triangles outside the approximating polygon that represents an image. External triangles are simply removed.
- *Internal triangles* are the Delaunay triangles that are situated inside the approximating polygon of a static image. Internal triangles are identified by shooting a ray (half-line) from the centroid of a particular triangle in any direction, so that the ray does not directly hit any vertices of the approximating polygon. The ray originates inside an internal triangle if the ray intersects the edges of the approximating polygon an odd number of times (see [72].)
- *External edges* are the sides of internal triangles that coincide with the image boundaries.
- *Internal edges* are internal triangle sides inside the approximating polygon of the image. Note that two adjacent internal triangles have a common internal edge so that the internal edges connect the internal triangles that partition the approximating polygon of an image.



**Figure 3.3:** (a) A scanned signature with (b) a part of its noisy boundary. (c) The filtered version of (b). (d) Vertices (filled dots), external edges (solid lines) and internal edges (dashed lines) are used to classify the internal triangles. (e) The primary skeleton (solid centrelines) of (c). (f)-(g) The primary skeleton of (a) without and with prior smoothing.

- Internal triangles having zero, one, two, or three internal edges are labelled *isolated-triangles (I-Ts)*, *end-triangles (E-Ts)*, *normal-triangles (N-Ts)* and *junction-triangles (J-Ts)*, respectively. Figure 3.3(d) shows examples of each triangle type, where the black dots represent control points (vertices) of the approximating polygon that represents the original image. The control points also form the vertices of the Delaunay triangles. External edges are rendered as solid lines, whereas internal edges are rendered as dashed lines.

A *primary skeleton* is obtained as follows: for N-Ts, lines connecting the midpoints of their internal edges are computed. For E-Ts and J-Ts, the skeletons are straight lines connecting their centroids to the midpoints of their internal edges, whereas the skeletons for I-Ts are their centroids. The appropriate skeletons for the internal triangles from Figure 3.3(d) are shown in Figure 3.3(e). The primary skeleton for the signature in Figure 3.3(a), if no prior smoothing is applied to the image boundaries, is shown in Figure 3.3(f). The result when smoothing



is applied, is shown in Figure 3.3(g). Note the reduction of spurs as a result of a smoother boundary. The few remaining spurs must now be removed and intersection artifacts corrected.

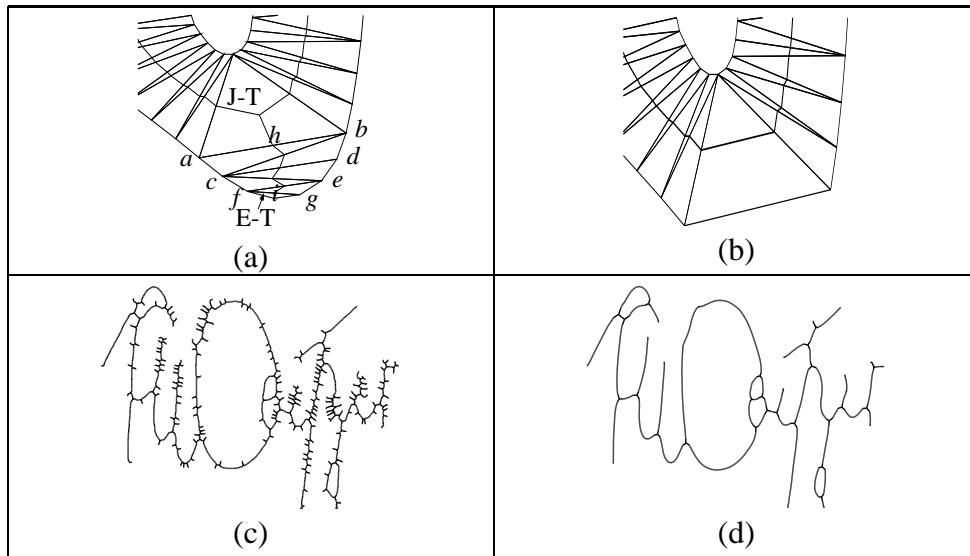
### 3.1.5 Removing artifacts

Parts of a handwritten static script that are difficult to unravel, as well as intersection and peripheral artifacts, are identified by means of a parameter  $\alpha$ , which is the ratio between the width  $w$  and length  $\ell$  of a ribbon, i.e.,  $\alpha = \frac{w}{\ell}$ , where a *ribbon* is a set of connected N-Ts between two J-Ts, or between a J-T and an E-T. A *long ribbon* is identified when  $\alpha$  is smaller than a threshold value, whereas a *short ribbon* is identified when  $\alpha$  is larger than a threshold value. The width  $w$  of a ribbon is taken as the trimean length over all internal edges that constitute the ribbon, i.e., the weighted average of the 25th percentile, twice the 50th percentile and the 75th percentile. The length  $\ell$  is the path length of the connected N-T skeleton lines that constitute the ribbon. Figure 3.4(a) depicts a typical ribbon between an E-T and a J-T. The length of the ribbon is computed as the path length of the skeleton line that connects the midpoints of the internal edges from  $h$  to  $i$ . The width  $w$  of the ribbon is given by the trimean of  $\{\|\mathbf{ab}\|, \|\mathbf{bc}\|, \|\mathbf{cd}\|, \|\mathbf{ce}\|, \|\mathbf{ef}\|, \|\mathbf{fg}\|\}$ , where  $\mathbf{xy} = \mathbf{y} - \mathbf{x}$  and  $\mathbf{x}, \mathbf{y}$  are both 2D boundary coordinates. The algorithm proceeds in several steps:

**Step 1: Removing spurs.** The first step in the skeletonisation is to remove all peripheral artifacts remaining after boundary smoothing. Following [87], short spurs belong to sets of connected triangles that are short in comparison with their width; they are removed. If  $\alpha \geq 2$ , the ribbon is identified to be a short ribbon and removed, so that the J-T becomes an N-T as shown in Figure 3.4(b).

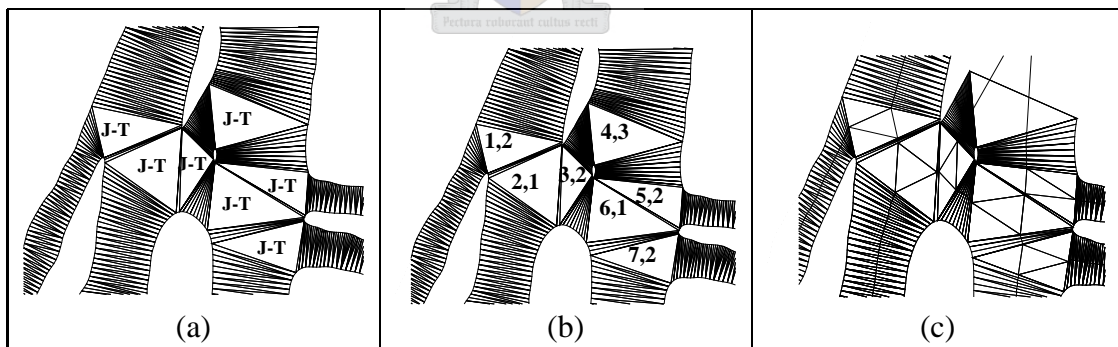
The threshold for  $\alpha$  depends on the boundary noise—less boundary noise results in shorter spur lengths. Thus, the threshold for  $\alpha$  is increased as the boundary noise decreases. Figures 3.4(c) and (d) show the result after *Step 1* is applied to Figures 3.3(f) and (g) with  $\alpha \geq 2$ . Note that most of the important image features from Figure 3.3(a) are preserved in Figure 3.4(d), whereas it becomes difficult to calculate a threshold for  $\alpha$  that removes spurs from Figure 3.3(f) without removing important image features. Clearly, smoothing significantly improves spur removal as spurs are shortened in a natural way, making it easier to compute a robust value for  $\alpha$ .

**Step 2: Identifying complicated intersections.** Figure 3.5(a) indicates the typical locations of J-Ts, as derived from a complicated part in a signature. If so many lines cross in a small area, it is difficult, if not impossible, to maintain the integrity of lines, i.e., it is difficult to follow individual ribbons through intersections. The Delaunay triangles enable us to identify such complexities as parts of images where many J-Ts are within



**Figure 3.4:** *Removing peripheral artifacts. (a) An illustration of the parameters involved to determine if the ribbon between the J-T and E-T is spurious. Vertices that are used to compute the width of the ribbon are labelled from a to g, whereas the length of the ribbon is the path length of the skeleton line between h and i. (b) Removing the spurious end region from (a). (c) Removing spurs from Figure 3.3(f). (d) Removing spurs from Figure 3.3(g).*

close proximity, as shown in Figure 3.5(a). Instead of forcing poor decisions in such complicated parts, web-like structures are introduced, including additional skeleton lines to preserve all possible connections.



**Figure 3.5:** *Identifying complicated intersections. (a) Cluttered J-Ts extracted from a complicated part of a signature. (b) Illustration of Step 2, where J-Ts are numbered, followed by the number of long ribbons that are connected to them. (c) Final skeleton for (b) containing web-like structures superimposed on the internal Delaunay triangles.*

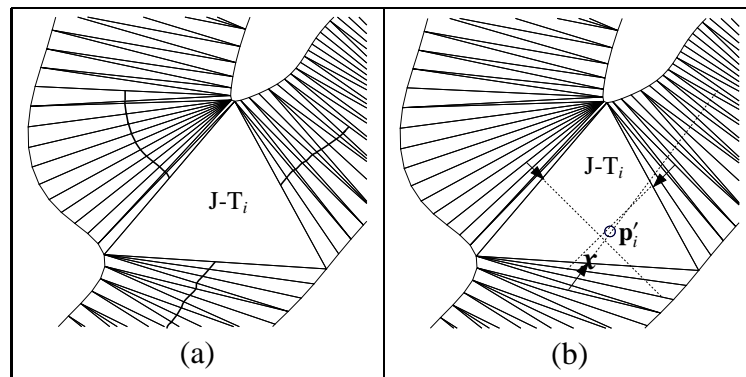
Recall that during the primary skeletonisation, the centroids of all J-Ts become skeleton points. As mentioned above, it is important to avoid forcing crucial decisions in complicated parts of a static script. Hence, for complicated intersections, the primary skeleton points of the J-Ts are removed, and the lines that enter the J-Ts are directly connected. The

resulting web-like structures contribute to smoother transitions than the original primary skeleton points in complicated parts of the image. We proceed to discuss the heuristic measures employed to identify J-Ts that belong to such complicated intersections.

First, J-Ts that are connected to two or three short ribbons ( $\alpha \geq 2.5$ ), are labelled *complicated J-Ts*. The primary skeleton points of complicated J-Ts are replaced by lines connecting the midpoints of the J-T internal edges. The same is done for other J-Ts that are connected to complicated J-Ts through short ribbons. This is illustrated in Figure 3.5(b), where the J-Ts from Figure 3.5(a) are numbered, followed by the number of long ribbons connected to the J-Ts. Although J-Ts 1 and 7 are connected to two long ribbons, they are connected to complicated J-Ts through short ribbons, so that their primary skeleton points are also replaced with web-like structures, connecting the midpoints of their internal edges, as shown in Figure 3.5(c). Note that our HMM extracts the appropriate connections, as described in Chapter 5.

**Step 3: Characterising skeleton points.** The remaining uncomplicated J-Ts are either connected to two or three long ribbons. The skeleton points of such uncomplicated J-Ts (recall that the primary skeleton selected the centroid) are recalculated following a similar approach to [86].

**Recalculating the skeleton points of uncomplicated intersections.** The midpoints of internal edges belonging to the first few triangles (we use thirteen) in all three ribbons connected to a J-T are connected and smoothed using a smoothing cubic spline, as shown in Figure 3.6(a). The average directions of these curves are then calculated and extended in the direction of the J-T, as illustrated by the dashed lines in Figure 3.6(b). Let the skeleton point of  $J-T_i$  be  $\mathbf{p}_i$ , i.e., at this stage  $J-T_i$  does not belong to a web-like structure so that  $\mathbf{p}_i$  is the centroid of  $J-T_i$ , where  $i \in \{1, \dots, n\}$  and  $n$  is the number of uncomplicated J-Ts. The new skeleton point  $\mathbf{p}'_i$  of  $J-T_i$  is computed by calculating the centroid of the intersections between the extended lines, as indicated by a circle in Figure 3.6(b).



**Figure 3.6:** Calculating crosspoints to identify complicated intersections. (a) Cubic splines to estimate the local directions of the ribbons that enter  $J-T_i$ . (b) The local ribbon directions (arrows) are extended (dashed lines) to compute the skeleton point  $\mathbf{p}'_i$  for  $J-T_i$  in (a).

**Recalculation of skeleton points that are out of bounds.** In some cases  $\mathbf{p}'_i$  falls completely outside J-T<sub>*i*</sub> and all the ribbons connected to J-T<sub>*i*</sub>, e.g., in the image background. To preserve local line directions,  $\mathbf{p}'_i$  is relocated to an appropriate triangle closest to it. Specifically, for each ribbon *j* connected to J-T<sub>*i*</sub>, the nearest triangle  $T_{i,j}$  to  $\mathbf{p}'_i$  is computed. Thus,  $T_{i,j}$  can be any triangle that partially constitutes the *j*th ribbon connected to J-T<sub>*i*</sub> for  $j \in \{1, 2, 3\}$  and  $i \in \{1, \dots, n\}$ . For each  $T_{i,j}$ , the angle  $\theta_{ij}$  is computed, where

$$\theta_{ij} = \cos^{-1} \left( \frac{(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i - \mathbf{p}'_i)}{\|(\mathbf{p}_i - \mathbf{p}_j)\| \cdot \|(\mathbf{p}_i - \mathbf{p}'_i)\|} \right), \quad (3.2)$$

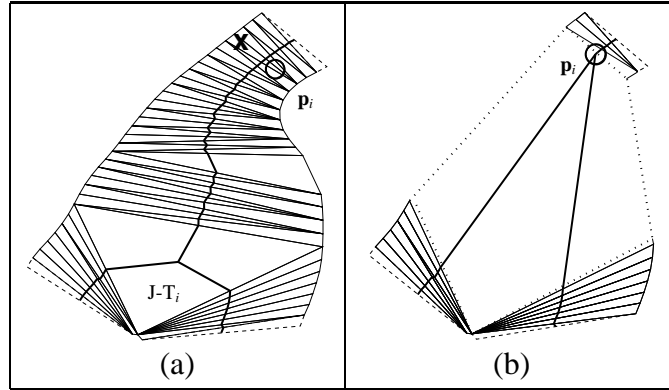
where  $\mathbf{p}_j$  is the centroid of  $T_{i,j}$ ,  $\mathbf{p}_i$  is the centroid of J-T<sub>*i*</sub>, and  $\mathbf{p}'_i$  is the recalculated value for  $\mathbf{p}_i$  which must be recalculated again. The triangle  $T_{(i,j)\min}$  corresponding to the minimum  $\theta_{(i,j)}$  is chosen as the triangle that should contain  $\mathbf{p}'_i$ . If  $T_{(i,j)\min}$  is an E-T,  $\mathbf{p}''_i$ , the new value for  $\mathbf{p}'_i$ , is the centroid of the E-T. If  $T_{(i,j)\min}$  is an N-T,  $\mathbf{p}''_i$  is the centroid of the midpoints of the N-T's two internal edges. Finally, the skeleton point  $\mathbf{p}_i$  for each uncomplicated J-T<sub>*i*</sub> is recalculated as  $\mathbf{p}_i = \mathbf{p}'_i$ , or  $\mathbf{p}_i = \mathbf{p}''_i$  if  $\mathbf{p}_i$  is out of bounds.

**Associating a ribbon with each crosspoint.** We now associate a single ribbon *j* with a crosspoint  $\mathbf{p}_i$ , where ribbon *j* is selected from the three ribbons that are connected to the uncomplicated J-T<sub>*i*</sub>. Specifically, the distances between  $\mathbf{p}_i$  and the midpoints of J-T<sub>*i*</sub>'s internal edges (each corresponding to a specific ribbon) are calculated. The midpoint of edge *j* closest to  $\mathbf{p}_i$  defines the ribbon *j*, associated with  $\mathbf{p}_i$ , e.g., the ribbon *x* is associated with  $\mathbf{p}_i$  in Figure 3.6(b). All the crosspoints along with their associated ribbons are stored to be used at a later stage of the algorithm.

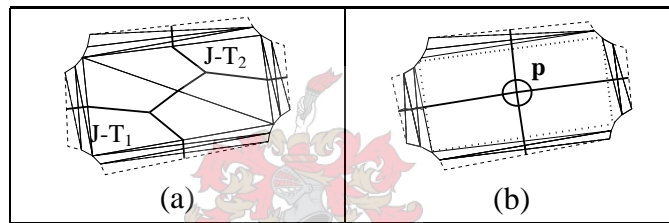
**Step 4: Removing intersection artifacts (criterion 1).** This step identifies uncomplicated J-Ts (excluded from web-like structures) that contribute to intersection artifacts so that artifacts can be removed by adapting some of the criteria used by [86]. A J-T<sub>*i*</sub> is labelled *unstable*, i.e., contributing to an artifact, if its skeleton point  $\mathbf{p}_i$  lies outside it. In this case, the sequence of connected triangles from J-T<sub>*i*</sub> up to the triangle in which  $\mathbf{p}_i$  falls are removed, thereby merging them into a single polygon. Figure 3.7(a) indicates  $T_{(i,j)\min}$  for J-T<sub>*i*</sub> using an x-shaped marker. The intersection region resulting from the removal of all the triangles up to  $\mathbf{p}_i$  is depicted in Figure 3.7(b). Note that  $\mathbf{p}_i$  is now associated with a pentagon (five-sided polygon rendered as dotted lines.)

An extension of Step 4 is illustrated in Figure 3.8. The primary skeletons of J-T<sub>1</sub> and J-T<sub>2</sub> from Figure 3.8(a) must be joined to remove the intersection artifact (solid line between J-T<sub>1</sub> and J-T<sub>2</sub>). In this case, the skeleton point  $\mathbf{p}_2$  of J-T<sub>2</sub> falls inside J-T<sub>1</sub>, so that the two J-Ts are united into a four-sided polygon (dotted rectangle) with skeleton point  $\mathbf{p}$  (circle), as shown in Figure 3.8(b), where  $\mathbf{p} = (\mathbf{p}_1 + \mathbf{p}_2)/2$ .

More intersection artifacts are identified and removed during the next step.



**Figure 3.7:** Correcting intersection artifacts. (a) Recalculating the skeleton point  $\mathbf{p}_i$  (circle) for  $J-T_i$ , where the ribbon  $j$  that is associated with  $J-T_i$  is indicated by an x-shaped marker. (b) All the triangles from  $J-T_i$  to  $\mathbf{p}_i$  are removed from (a) to calculate a new intersection region (dotted polygon) containing the crosspoint  $\mathbf{p}_i$ , thereby preserving the line directions of the y-shaped pattern (thick solid line) better than in (a).

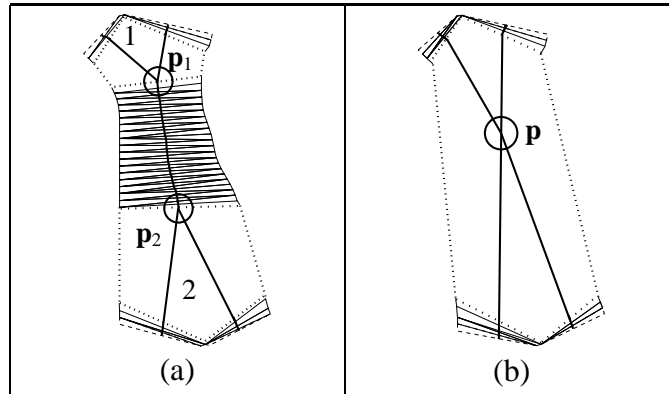


**Figure 3.8:** Removing an intersection artifact using an extension of Step 4. (a) An intersection artifact (solid line) between  $J-T_1$  and  $J-T_2$  and (b) removal thereof by uniting  $J-T_1$  and  $J-T_2$  into a new intersection region (dotted polygon) with skeleton point  $\mathbf{p}$ .

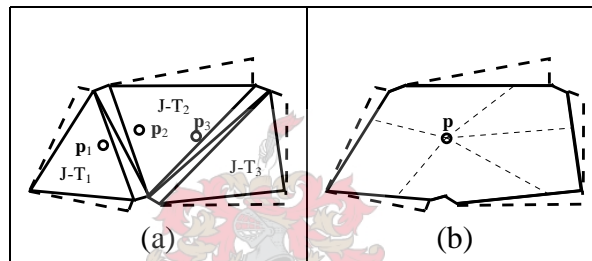
**Step 5: Removing intersection artifacts (criterion 2).** We now make use of the information about the location of skeleton points and their associated ribbons obtained in *Step 3*. If two crosspoints  $\mathbf{p}_1$  and  $\mathbf{p}_2$  are associated with the same ribbon, as shown in Figure 3.9(a), and  $\alpha \geq 2$  for the ribbon, the two intersection regions and the ribbon between them are united into a new intersection region. Note that after the application of *Step 4* a ribbon connects an intersection region (triangle/polygon) to an intersection region or an E-T. The skeleton point for the new intersection region (dotted polygon) is  $\mathbf{p} = (\mathbf{p}_1 + \mathbf{p}_2)/2$ , as shown in Figure 3.9(b).

In addition, three J-Ts must sometimes be merged, as illustrated in Figure 3.10. Figure 3.10(a) depicts three J-Ts and their skeleton points  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  and  $\mathbf{p}_3$ . Conditions for such a merge occur if according to *Step 4*,  $J-T_2$  and  $J-T_3$  must be united, whereas according to *Step 5*,  $J-T_1$  and  $J-T_2$  must be united. In such cases, a new intersection region is created with a single skeleton point  $\mathbf{p} = (\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3)/3$ , as shown in Figure 3.10(b).

The final step modifies *Step 1* and removes spurs after the application of *Steps 4* and *5*.



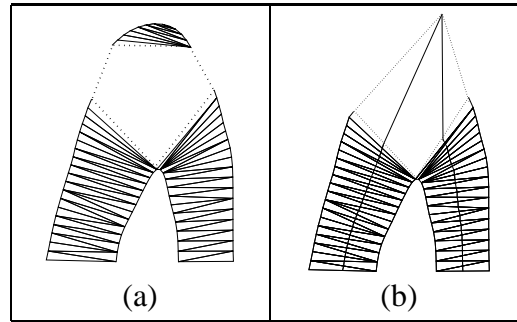
**Figure 3.9:** Intersection regions are united if their skeleton points are associated with the same short ribbon. (a) Skeleton points  $\mathbf{p}_1$  and  $\mathbf{p}_2$  of intersection regions 1 and 2 (numbered dotted polygons) are associated with the same short ribbon. (b) The ribbon and intersection regions from (a) are united into a new intersection region (dotted polygon with skeleton point  $\mathbf{p}$ ).



**Figure 3.10:** Merging three J-Ts, where (a)  $J-T_2$  must merge with  $J-T_3$  and  $J-T_1$  according to the locations of the J-T skeleton points  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  and  $\mathbf{p}_3$  and the criteria imposed by Steps 4 and 5. (b) A new intersection region (solid lines) with skeleton point  $\mathbf{p}$  and skeleton lines connecting the midpoints of its internal edges (thin dashed lines) results after merging the J-Ts from (a).

**Step 6: Removing spurs by modifying Step 1.** If a crosspoint  $\mathbf{p}$  is associated with a short ribbon ( $\alpha \geq 2.5$ ) that is connected to its intersection region and an E-T, the intersection region is united with all the connected triangles up to the E-T. The skeleton point of the new intersection region is the centroid of the E-T. The intersection region (dotted polygon) of Figure 3.11(a) is connected to an E-T through a short ribbon. A new intersection region is therefore computed (dotted polygon) resulting in a v-shaped pattern (solid line), as shown in Figure 3.11(b).

Although Steps 1 and 6 appear similar, there are subtle differences. Step 1 measures the length of all ribbons that are connected to E-Ts and J-Ts and serves to remove the excessive artifacts that can affect the rest of the algorithm. Step 6 removes peripheral artifacts by relocating crosspoints to the centroids of E-Ts, i.e., Step 6 transforms y-shaped patterns into v-shaped patterns.



**Figure 3.11:** *Removing the last spurs. (a) An intersection region (dotted polygon) connected to an E-T through a short ribbon. (b) Merging connected triangles from the intersection region up to the E-T from (a) in accordance with Step 6 to compute a new intersection region (dotted polygon) and skeleton (thick solid line.)*

### 3.1.6 Results: The final skeletons

Final skeletons are smoothed using Chaikin’s corner-cutting subdivision method [10, 49]. This smoothing scheme treats the samples that constitute a parametric curve as control points of a polygon and iteratively “cuts” the corners of the polygon while doubling the numbers of samples that constitute the curve. It is shown by Lane and Riesenfeld [49] that Chaikin’s curve is equivalent to a quadratic B-spline curve (a piecewise quadratic Bézier curve.) Due to Chaikin’s *geometric* approach to smooth curves, a wide variety of shapes can be handled easily and efficiently, e.g., straight lines and closed curves are treated the same, making it an appropriate smoothing scheme for this application. This smoothing scheme is applied to all curves connected to endpoints and crosspoints, as well as closed curves (e.g., the character “o”.) Further resampling before deriving an HMM from the skeletons is described in Section 3.3.

On-line signatures from the Doling database [20, 82] (converted into thin off-line signatures) and the off-line signatures from the Stellenbosch dataset developed by Coetzer [12] were used to optimise the threshold values for  $\alpha$  at each step; see the previous section. The average line thickness of the static signatures that were skeletonised varied between 1.7 and 8.1 pixels with average standard deviations between 0.3 and 5.3. Although the signatures vary considerably as far as line thickness and boundary noise levels are concerned, the same threshold values for  $\alpha$ , as presented in the previous section, are used in all cases. Examples of final skeletonised signatures are shown in Figure 3.12. Figure 3.12(a) shows the original signatures, while their application-specific and general-purpose skeletons are presented in Figure 3.12(b)-(c). The *application-specific* skeletons are our pseudo skeletons containing web-like structures. The *general-purpose* skeletons are our pseudo skeletons that do not contain web-like structures, i.e., the skeleton points of complicated J-Ts are their primary skeleton points (see *Step 2*.) Thus, except for the web-like structures, the general-purpose skeletons are the same as the

application-specific skeletons. The general-purpose skeletons can be used for other off-line handwriting applications where improvements of standard skeletons are required to compute visually appealing skeletons.

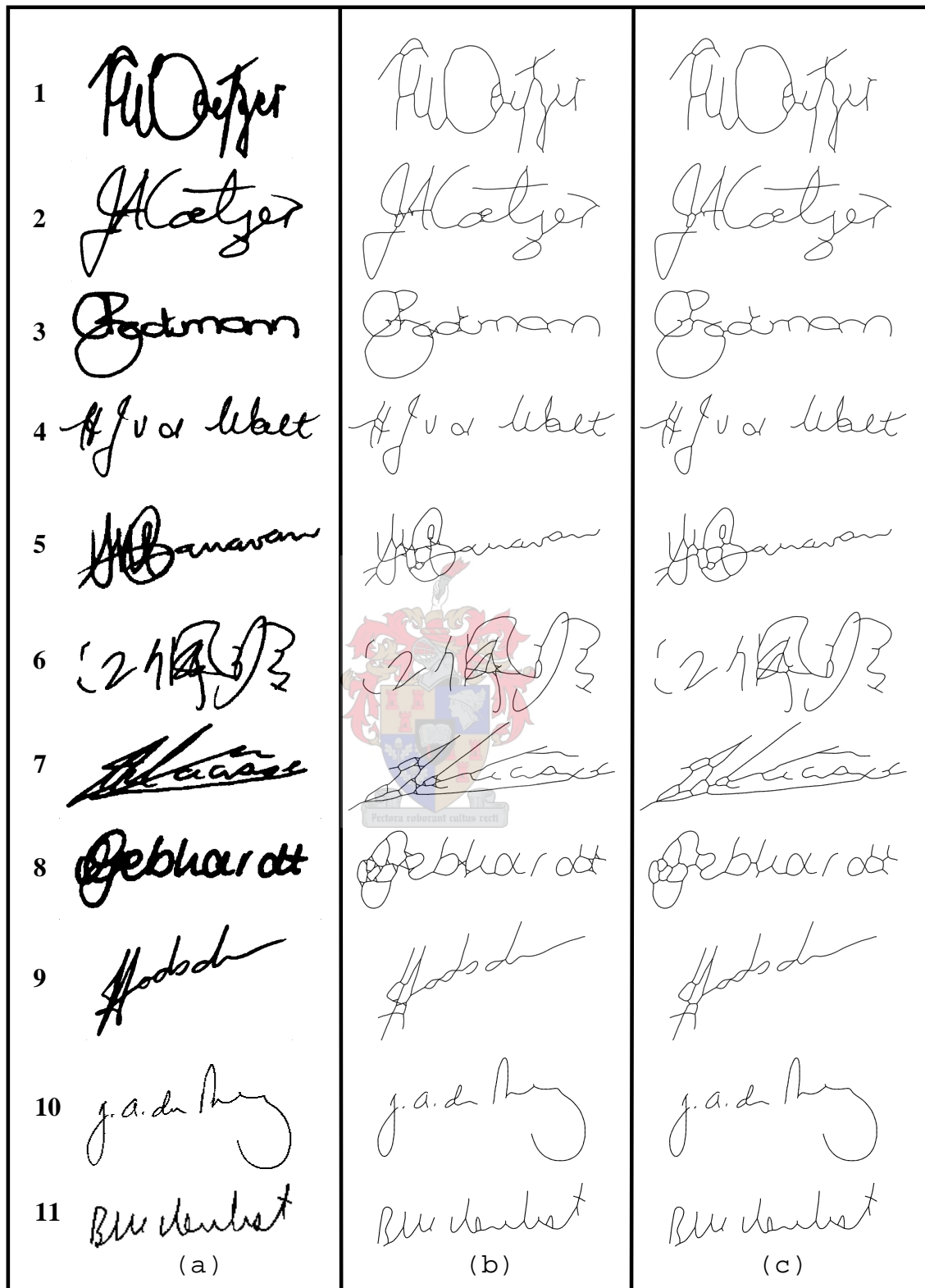
Signatures 1, 10 and 11 are examples of signatures that are relatively straightforward for the eye to unravel. Their application-specific and general-purpose skeletons are therefore the same. Note how local line directions are improved in the skeleton of the first signature after the application of *Steps 2 to 6* as compared to Figure 3.4(d). Furthermore, web-like structures retain all possible connections while smoothing transitions at intersections slightly in parts that are difficult to unravel. The application-specific skeletons of signatures 5 and 8 illustrate that our pseudo skeletonisation algorithm is able to identify difficult parts (evident from the webs on the left-hand parts), whereas intersection and peripheral artifacts are corrected in parts that are relatively straightforward to unravel (right-hand parts.) Quantitative measurements to determine the efficacy of our skeletonisation scheme for this application are presented in Chapter 6.

### 3.1.7 Summary and conclusions

Before drawing some conclusions, our skeletonisation scheme is briefly summarised as follows:

- First, the boundaries of static handwritten images are extracted, smoothed and resampled. Small polygons are removed, while the rest of the polygons are subdivided into non-overlapping Delaunay triangles. These triangles are used as the foundation to calculate the primary skeletons of static images.
- Peripheral artifacts are removed by removing short ribbons ( $\alpha \geq 2$ ) connected between J-Ts and E-Ts.
- Parts of the signature that are difficult to unravel are identified and web-like structures are introduced. Here  $\alpha \geq 2.5$ . (For general purposes, the primary skeletons in complicated parts are retained.)
- Unstable J-Ts and other intersection regions that contribute to intersection artifacts are identified. Intersection artifacts are corrected using two criteria. The first criterion merges a J-T with a connected set of triangles up to its estimated skeleton point. The second criterion unites two intersection regions if their skeleton points are associated with the same short ribbon ( $\alpha \geq 2$ .)
- The last peripheral artifacts are identified after the recalculation of crosspoints in the steps above. Here  $\alpha \geq 2.5$ .
- Skeletons are smoothed using a corner-cutting subdivision scheme.





**Figure 3.12:** Examples of our pseudo skeletonisation, where the scanned static signatures are shown in (a) with (b) their application-specific skeletons and (c) the general-purpose skeletons derived from (a).

Apart from the specific application to static signatures, the above skeletonisation procedure is also very suitable for arbitrary handwritten shapes. This was achieved by introducing a number of detailed modifications to the basic skeletonisation technique in [86] and developed further by [69]. In particular we find that smoothing of boundaries reduces the number of artifacts significantly and enhances local line directions. An important feature of the algorithm is our exploitation of line width. This allows us to identify the complicated parts of the signature where difficulties might be encountered. It is of interest to note that the parameters depending on the line width can be fixed for a wide class of handwritten images, possibly for all handwritten images.

To conclude, our skeletonisation approach has the following beneficial characteristics for this application:

- In order to be able to extract the time sequence of the lines comprising static signatures, estimating local line directions is crucial. The challenge is to design a skeletonisation algorithm that preserves local line directions and maintains the correct connections between incoming and outgoing curves, especially in complicated parts of the signature. Our pseudo skeletonisation identifies complicated parts in static handwritten scripts while enhancing local line directions using various smoothing techniques. An adaption of our pseudo skeletons for general purposes is also proposed for applications that require artifact removal and visually appealing skeletons.
- Due to the useful shape partitioning of static scripts using Delaunay triangulation, our pseudo skeletonisation enables us to identify simple intersections that are easy to unravel, as shown in Section 4.5.
- The segmentation of a static handwritten script into a selection of smooth parametric curves with few artifacts enable us to identify high curvature points accurately, as described in Section 3.3.

The effect of skeletonisation artifacts on our system is investigated in Section 6.3.2.

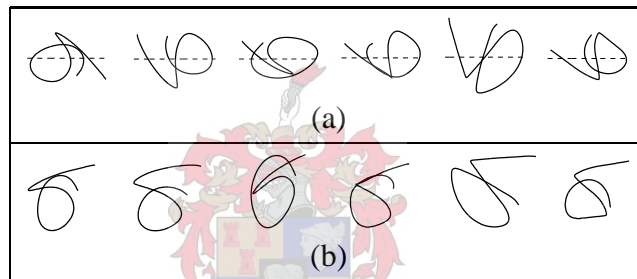
## 3.2 Orientation normalisation

Any form of handwriting is generated with a specific general direction relative to the horizontal axis, which we refer to as the *orientation* of the handwriting. Since our algorithm relies on local line directions, it is important that the static image and the dynamic exemplar have the same orientation.

Principal component analysis (PCA) is often used to align different shapes [28, 54, 56]. Unfortunately, this simple procedure is not reliable for signatures. To calculate the principle axes of a shape, the covariance matrix  $\mathbf{C}$  of the data, representing the shape, is computed. The matrix  $\mathbf{E}$  containing the eigenvalues of  $\mathbf{C}$  on its main diagonal is then computed, where

$$\mathbf{E} = \begin{bmatrix} \beta_1 & 0 & \dots & 0 \\ 0 & \beta_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \beta_n \end{bmatrix}, \quad (3.3)$$

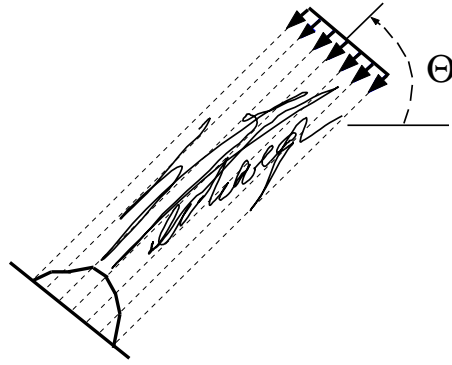
where  $n$  is the dimension of the data and the eigenvalues are sorted in descending order so that  $\beta_j \geq \beta_{j+1}$  for  $j = 1, 2, \dots, n - 1$ . The eigenvector corresponding to  $\beta_1$  is then the axis of maximum variation and therefore the principle axis. For our application  $n = 2$ . Hence, problems are encountered with shapes that do not display a clear “direction”, i.e., when  $\beta_1 \approx \beta_2$  which causes mismatched principle axes of dynamic exemplars and static skeletons.



**Figure 3.13:** Aligning signatures with (a) PCA and (b) the Radon transform.

A more robust approach is provided by shape matching algorithms in the Radon and Hough domains [26, 81, 12]. The Radon and Hough transforms are frequently used to detect straight lines in an image. The estimated equations of the straight lines enable one to detect italic (*slanted*) handwritten characters or to determine the general orientation of a document (*document skew*); see, e.g., [70, 39]. The Radon transform consists of projections at different angles  $\theta$ , where  $0^\circ \leq \theta < 180^\circ$ , as shown in Figure 3.14, so that all the original image information is contained in the projections for  $0^\circ \leq \theta < 180^\circ$ . The Radon transform is computed over  $n$  angles, where  $n = 360$  for this application. The Radon transform is periodic and rotation becomes a linear shift in the Radon domain. At a specific angle  $\theta_j$ , an image is presented by  $m$  line integrals (projections over  $m$  beams), where  $j \in \{1, \dots, n\}$ .

Since a rotation of an image corresponds to a linear shift in the Radon and Hough domains (see, e.g., [35]), it is straightforward to calculate the optimal match between a dynamic exemplar and a static script. We use the Radon transform in a general shape matching algorithm, which is very similar to the algorithms described in [26, 81, 12]. The dynamic exemplar is then converted into a static image. Since the Radon transform is sensitive to the line width of the images, we thicken [28] the static image skeleton as well as the image derived from the dynamic exemplar to



**Figure 3.14:** A projection of a static signature in the Radon domain, where the dashed lines depict the projection at an angle  $\theta$  relative to the  $x$ -axis.

a line width of approximately five pixels. This results in the following orientation normalisation algorithm:

1. The relative angle between a reference image (originally the static script) and test image (originally the dynamic exemplar) that produces the closest match in the Radon domain is computed. The Euclidean distance  $d_i$  is computed for all possible rotations so that

$$d_i = \sqrt{\sum_{j=1}^{360} \|\mathbf{t}_j - \mathbf{r}_{i+j}\|^2}, \quad (3.4)$$

where  $\mathbf{t}_j$  and  $\mathbf{r}_{i+j}$  are the Radon transforms ( $m$  dimensional vectors) of the test and reference images at  $\theta_j$  and  $\theta_{i+j}$ , respectively, and  $i \in \{1, \dots, 360\}$ . The value of  $k$  that produces the minimum value to (3.4), i.e.,  $k = \arg(\min(\mathbf{d}))$ , where  $\mathbf{d} = [d_1, d_2, \dots, d_n]$ , is computed.

2. The dynamic exemplar is rotated with  $\theta_k$ , where  $k$  is computed in the previous step, to align it with the skeleton of a static script.

As far as we are aware, only Coetzer [12] employs this Radon-based scheme in a signature verification application. However, existing techniques that quantify the proficiency of PCA-based versus Radon-based orientation normalisation could not be found. A direct comparison between these two normalisation schemes is therefore presented in Section 6.3.3, where the effect of these normalisation schemes on our system is also measured.

### 3.2.1 Summary

This section has shown how to match a static image and a dynamic exemplar in the Radon domain in order to calculate their relative angle of rotation. The dynamic exemplar is then rotated

with this angle to align its orientation with the orientation of a static skeleton. This approach is applicable to any method that requires rotation normalisation between two shapes. Many signature verification approaches use PCA to align the orientations of the signatures. Local shape similarities form the foundation for our Radon-based rotation, whereas global characteristics with less shape information form the foundation for PCA-based rotation. The effect of PCA-based and Radon-based orientation normalisation on our trajectory estimation algorithm and a more quantitative comparison between the two approaches are investigated in Section 6.3.3.

In addition to the alignment of two shapes, a similarity measure between the shapes is also obtained that can be useful in shape recognition or signature verification systems. This is endorsed by Terrades and Valveny [81] who note that: “Symbols with a common structure share maxima location in the Radon transform, although the Radon transform also reflects differences between them.”

### 3.3 Resampling

The performance of the HMM is significantly improved by employing a suitable sampling scheme. An adequate sampling scheme is also required to establish a comparison between standard skeletons and our pseudo skeletons from Section 3.1.

A dynamic exemplar sample  $\mathbf{x}_t$  at time instance  $t$  is a 3D vector consisting of  $\mathbf{x}_t^{1,2}$ , the 2D position coordinate in the first two entries, and  $x_t^3$  the pen pressure in the last entry, where  $x_t^3 \in \{0, \dots, 255\}$ . Since the final skeletons are smoothed using Chaikin’s method (see Section 3.1.6), the dynamic exemplars are also smoothed using the same method. For reasons that become more apparent in Chapter 5,  $x_t^3$  is normalised so that  $x_t^3 = 1$  if the pressure at instance  $t$  is non-zero, and  $\mathbf{x}_t^{1,2} = (0, 0)$  and  $x_t^3 = 0$  in cases where the pen pressure is zero at  $t$ .

Subsequent resampling proceeds in two steps:

1. Any two successive samples that form part of a standard pixel-based skeleton of a static image are within a distance of 1 or  $\sqrt{2}$  pixels from each other. To obtain a comparative resampling, all smoothed curves are resampled so that  $\|\mathbf{x}_t - \mathbf{x}_{t+1}\| \approx 1$ , where  $\mathbf{x}_t$  is the 2D position coordinate at time instance  $t$ . The result is called *Euclidean resampling*, as the distance between any two successive samples is approximately the same. This resampling scheme is applied to all the position coordinates of the dynamic exemplar curves created with non-zero pressure. It is also applied to static skeleton curves connected to endpoints and crosspoints, as well as circular shapes.
2. During the next step of the resampling, the most important samples, called *critical points*

are selected from the dynamic exemplars and static scripts after the Euclidean resampling, as described in Section 3.3.1. The resulting parametric curve is called a *critical point resampled curve*.

### 3.3.1 A critical point resampling scheme

It is shown in Section 6.3.4 that the computation time of our trajectory algorithm depends on the numbers of samples that constitute the static skeletons and dynamic exemplars. Thus, for this application, it is desirable to reduce the number of Euclidean samples without degrading the performance of our HMM. Hence, the Euclidean resampled curves that represent the dynamic exemplars and static scripts are resampled by selecting only the critical points from the curves.

In general, when a piecewise linear interpolation is employed between the successive 1D samples  $x_t$  and  $x_{t+1}$ , the error  $e(x_t)$  of the linear interpolant between these successive samples is given by

$$e(x_t) = \frac{1}{2}|x_t - x_{t+1}|^2 f''(\xi_t), \quad (3.5)$$

where  $f''(\xi_t)$  is the second derivative at an unknown value  $\xi \in (x_t, x_{t+1})$  for  $t = [1, \dots, T - 1]$ . Thus, the maximum error of the linear interpolant  $e_{\max}(x_t) = \max(e(x_t))$  satisfies

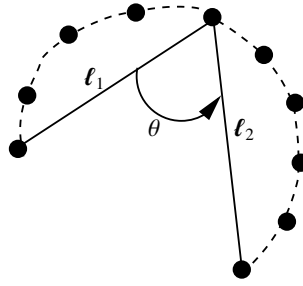
$$e_{\max}(x_t) \leq \frac{1}{8}h_t^2 M_t, \quad (3.6)$$

where  $|f''(\xi_t)| \leq M_t$ ,  $\xi_t \in (x_t, x_{t+1})$  and  $h_t = |x_{t+1} - x_t|$ . Thus, provided that  $|f''(\xi_t)|$  is bounded, it is natural to try and reduce the overall error by placing interpolation samples so that

$$e(x_t) = h_t^2 f''(\xi_t) \approx \text{constant}, \quad (3.7)$$

for  $\xi_t \in (x_t, x_{t+1})$  and  $t = [1, \dots, T - 1]$ . Thus, from the equations above, one can decrease the overall error by increasing the number samples where the curvature (second derivative) is high, reducing samples where the curvatures is low (see [15] for more detail on approximating splines.)

High curvature points are identified using the critical-point detection described in [73]. This technique computes the angular difference  $\theta$  between the slopes of two lines that are fitted to portions of a parametric curve at each sample, as shown in Figure 3.15. The longer the line segments, the more noise are smoothed out and the more samples constitute high curvature portions. It is noted in [73] that it becomes problematic to choose a single set of parameters in noisy images containing a wide range of image features, i.e., coexisting sharp and gentle curves in the presence of noise. Thus, our pseudo skeletonisation, which removes artifacts and performs smoothing, holds a great advantage over noisy standard skeletons in this regard.



**Figure 3.15:** *Selecting high curvature points from a set of control points (filled dots) that constitute a parametric curve (dashed line) using two straight line segments (solid lines) to estimate curvature ( $\theta$ ).*

For this application, our line segments that estimate the slopes at dynamic exemplar or skeleton samples have a length of 10, i.e., in Figure 3.15  $\|\ell_1\| = \|\ell_2\| = 10$ . Furthermore, if  $|\theta| \leq 170^\circ$ ,  $\mathbf{x}_t$  is selected as a critical point, otherwise  $\mathbf{x}_t$  is identified to be part of a straight line, where  $\mathbf{x}_t$  is a 2D position coordinate of the sample at instance  $t$ . Samples on straight lines are reduced so that the distance between any two successive samples on the straight lines is approximately five pixels. Skeleton crosspoints and endpoints are treated differently from other skeleton samples by our HMM (as explained in the chapters to follow) and are therefore also chosen as critical points. Due to the pen-tip width and digitising effects, it frequently happens that the curve that enters and the curve that exits a high curvature point are merged. Thus, starting at an endpoint, the first 10 connected points (excluding crosspoints) are selected as critical points. In accordance, the first and last 10 non-zero pressure samples of the single-path trajectories that constitute the dynamic exemplars are also chosen as critical points.

### 3.3.2 Summary

This section has described how the Euclidean resampled curves for static skeletons and dynamic exemplars are computed, so that so that results are comparative to results from standard skeletonisation schemes. It is also shown how critical points are chosen from the Euclidean resampled curves to reduce the computation time without degrading the performance of our pen trajectory estimation algorithm. This critical point resampling scheme favours good skeletons, i.e., smooth curves without noise. The effect of the different resampling schemes on our system is evaluated in Section 6.3.4.

# Chapter 4

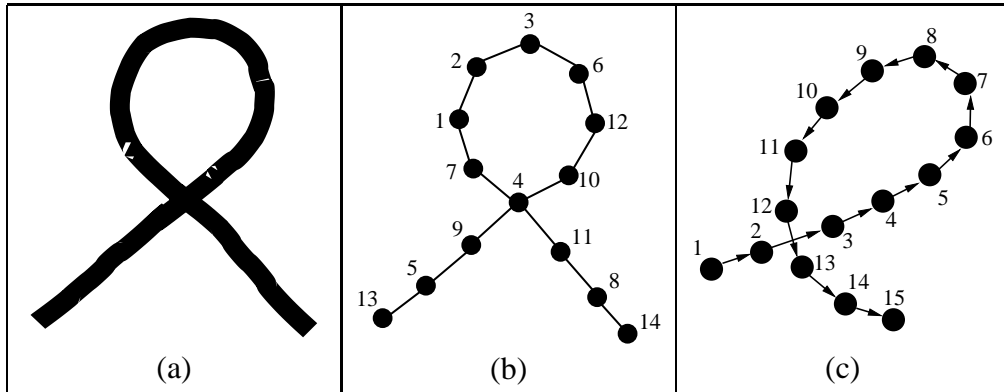
## The HMM for a single-path static script

The technique we develop for extracting the pen trajectory from a static, normalised image is based on an HMM. An HMM is a probabilistic model describing a dynamic process that evolves from one state to the next. In our application, the sequence of states describes the sequence of pen positions as the image is produced. An HMM is constructed from the static image skeleton. Using the HMM, a dynamic exemplar is matched to the static image. The matching algorithm results in the most likely pen trajectory of the static skeleton, given the model. In addition to the pen trajectory, one also obtains a quantitative correspondence between the static image and dynamic exemplar.

We explain the main ideas by means of the simple example shown in Figure 4.1. The static image of Figure 4.1(a) is skeletonised, as described earlier. In Chapters 4-5 skeleton samples that constitute a static image are always rendered as filled dots, whereas HMM states are rendered as unfilled numbered circles. The order of skeleton samples is unknown; a typical numbering is shown in Figure 4.1(b). Figure 4.1(c) shows a dynamic exemplar that must be matched to the static image. Note the shape differences between the two. Possible pen trajectories must be estimated from Figure 4.1(b) and compared with the known exemplar sequence of Figure 4.1(c). Since we do not know the optimal sequence of samples in Figure 4.1(b), or even the starting point for that matter, a very large number of possible sequences need to be compared—far too many for an exhaustive search. The use of an HMM, however, makes the calculation of the optimal pen trajectory computationally feasible.

To estimate the pen trajectory of the static image, two basic issues are addressed. First, a probabilistic model of the static script is created. More specifically, an HMM is created which describes the geometric shape of the script and restricts the choices of possible pen movements. Second, the optimal pen trajectory is calculated by matching the known dynamic exemplar to the HMM. This chapter presents our HMM for a single-path static script. The necessary





**Figure 4.1:** *Extracting dynamic information from a simple, single-path static signature. (a) A straightforward single-path static signature, with (b) its unordered skeleton samples (filled dots) and (c) a dynamic exemplar that can be used to extract the pen trajectory of (b).*

HMM background is provided in Section 4.1. A basic first-order HMM is then expanded and adapted for this application in Sections 4.2-4.6. All the HMM parameters are summarised in Section 4.7. Our HMM training scheme that estimates specific HMM parameters for each individual is described in Section 4.8. This chapter treats all static images as single-path static scripts. In Chapter 5 it is shown how to extend our approach so that multi-path static scripts can be handled.

## 4.1 HMM background

An HMM has  $N$  emitting states  $\{q_1, q_2, \dots, q_N\}$  that have observation PDFs associated with them. The two states  $q_0$  and  $q_{N+1}$ , without associated PDFs, are called *non-emitting* states. These two additional non-emitting states serve as initial and terminating states, thus eliminating the need for separate initial and terminating probabilities (see [22] for more detail.)

All state observation PDFs in this chapter are spherical Gaussian PDFs, described by

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{D}{2}} \sigma} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{x} - \boldsymbol{\mu})^T(\mathbf{x} - \boldsymbol{\mu})\right), \quad (4.1)$$

where  $\mathbf{x}$  is a  $D$ -dimensional vector that must be matched to the PDF and  $\boldsymbol{\mu}$  is the  $D$ -dimensional mean of the Gaussian. The standard deviation  $\sigma$  is preset for this application. For brevity, the PDF associated with state  $i$  having mean  $\boldsymbol{\mu}_i$  and standard deviation  $\sigma$  will be referred to as  $\mathcal{N}(\boldsymbol{\mu}_i, \sigma)$ . Geometric shape information of the static image is embedded in the PDF parameters  $\boldsymbol{\mu}_i$  and  $\sigma$ , as described in Section 4.2.

States are connected by transition links that dictate the possible pen movements. All transitions between states are weighted with transition probabilities. The *order* of the HMM specifies the

number of previous states the HMM considers when transiting to a next state. Sections 4.2–4.6 describe how the order of our HMM is increased to take context into account.

In order to match a static image and a dynamic exemplar, the dynamic exemplar is presented as a sequence of quantifiable characteristics called *feature vectors*. The sequence is given by  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T]$ , where  $\mathbf{x}_t$  denotes a  $D$ -dimensional feature vector at discrete-time instant  $t$ , and  $T$  is the number of feature vectors (number of samples in the dynamic exemplar.) Using the Viterbi algorithm,  $\mathbf{X}$  is matched to our HMM to produce a hidden state sequence  $\mathbf{s} = [s_1, s_2, \dots, s_T]$ . This state sequence is then mapped to the desired sequence of skeleton samples, as described in Section 5.4.

## 4.2 First-order HMMs

The shorthand notation for an HMM  $\lambda$  is

$$\lambda = \{A, \{\mathcal{N}(\boldsymbol{\mu}_i, \sigma), i = 1, \dots, N\}\}, \quad (4.2)$$

where  $A$  is a matrix representing the transition links and  $\mathcal{N}(\boldsymbol{\mu}_i, \sigma)$ , as described by (4.1), is the observation PDF of state  $i$  for  $i \in \{1, \dots, N\}$ .

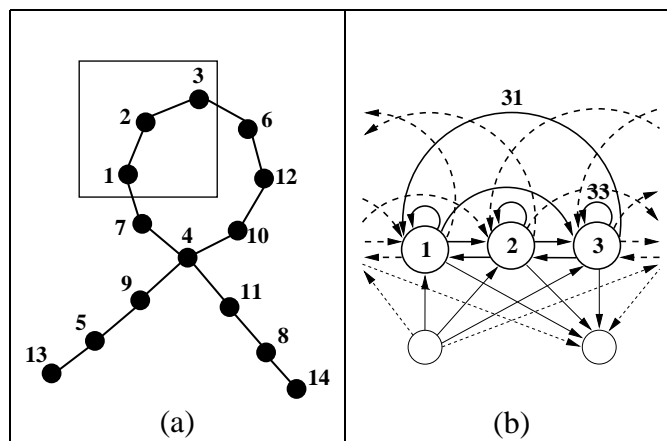
We begin by constructing a first-order HMM from the skeleton of the static image. The skeleton consists of  $M$  unordered samples  $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_M\}$ , where  $\mathbf{p}_x$  is the 2D coordinate of sample  $x$ . Each emitting state  $i$  is associated with a skeleton sample  $i$  by letting  $\boldsymbol{\mu}_i = \mathbf{p}_i$ . Thus, the observation PDF  $i$  of state  $i$  embeds the position coordinate of the skeleton sample. For a first-order HMM, we therefore have  $N = M$ . Our first-order HMM matches only 2D feature vectors, in this case, the pen positions of the dynamic exemplar. We choose  $\sigma = 0.7$  pixels in (4.2) for all states, in order to relate the match between the position coordinates of the dynamic exemplar and static image to Euclidean distance.

The HMM topology is crucial to our algorithm, as it constrains the range of possible pen movements that could generate the static image. For our first-order HMM, the probability of reaching the next state depends only on the current state, so that the transition probability matrix  $A = [a_{ij}]$ , where  $a_{ij} = P(s_{t+1} = q_j | s_t = q_i)$  is the probability of a transition from  $q_i$  to  $q_j$  at instance  $t + 1$ , with  $i, j \in \{0, 1, \dots, N + 1\}$  and  $t \in \{1, 2, \dots, T - 1\}$ . HMM states are called *neighbours* if their associated skeleton samples are adjacent. All emitting states are linked to their neighbours, to allow the pen to move to an adjacent skeleton point on a transition. However, this only takes local information into account, and not context. Context is incorporated by using second-order HMMs, which allow us to include a directional feature, as described in Section 4.3.

Since we have no prior knowledge of where the pen trajectory of the static image may start or end, the non-emitting initial state can enter any emitting state. Also, each emitting state is directly connected to the non-emitting terminating state.

One also needs elasticity in the model, to allow the static image and dynamic exemplar to have different numbers of samples. This is accomplished by including skip-links and self-loops in the HMM. A *skip-link* is a transition between two states separated by a neighbour common to both. A *self-loop* connects a state back to itself. Self-loops are added to the emitting states. In this dissertation, we use skip-links to skip states with only two neighbours. Equal transition probabilities are assigned to all transition links leaving a state, normalised to sum to one.

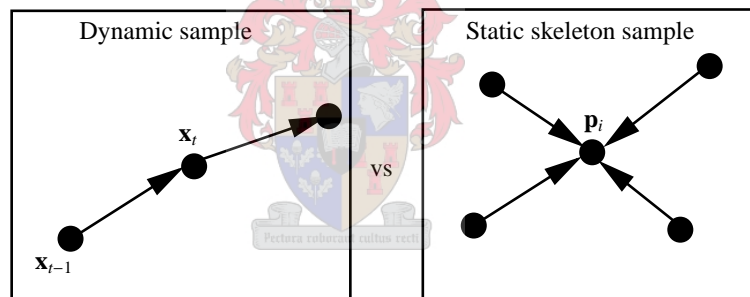
These ideas are illustrated in Figure 4.2. The first-order HMM for the isolated fragment in Figure 4.2(a) is shown in Figure 4.2(b), where the three states indicated by the larger circles are emitting states. Each state is labelled with a single number representing the index of the state, the state's PDF index (and therefore also the index of the skeleton sample associated with the state which is embedded in the PDF.) The dashed lines indicate transition links to and from states outside the rectangular box in Figures 4.2(a). The smaller blank circles indicate the non-emitting initial and terminating states. All states are connected to these non-emitting states so that the pen trajectory can start and end at any skeleton sample. Skip-link 31 and self-loop 33 are also indicated. State 1 and its neighbours each have two neighbours. Thus, state 1 has six transition links leaving it: two to its neighbours, two skip-links, one self-loop and one to the non-emitting terminating state. The associated six transition probabilities are therefore all specified as  $\frac{1}{6}$ .



**Figure 4.2:** Deriving the first-order HMM from a static signature skeleton. (a) Isolated unordered skeleton samples (within rectangle) in a signature. (b) The first-order HMM for the skeleton samples contained within the rectangle in (a). Non-emitting states (small circles) and emitting states (big circles) are connected with transition links (arrows), where dashed lines indicate links to states that correspond to skeleton samples outside the rectangle in (a).

We emphasise an important feature of Figure 4.2(b): Any two neighbouring emitting states and any two emitting states connected by a skip-link are connected both ways—if one enters the HMM at an emitting state with more than one neighbour it is not possible to determine locally in which direction one should move next, and therefore more than two directions are allowed. Since all transition links are assigned the same probability, all skeleton samples are potential turning points. It is therefore entirely possible, and it indeed happens in practice, that the extracted pen trajectory may incorrectly reverse direction.

One way to model turning points is to include more context. On-line handwriting systems typically include more context by extending the feature vectors. Specifically, the inclusion of a velocity/delta component  $\mathbf{x}_t - \mathbf{x}_{t-1}$  contributes substantially to the good results of these systems. As the time sequence of a handwritten script is known in an on-line system, each sample  $\mathbf{x}_t$  can be preceded by only *one* sample  $\mathbf{x}_{t-1}$ , permitting the inclusion of this velocity component. As indicated above, our static images consist of *random-order* skeleton samples. Thus, a skeleton sample can be preceded by multiple other skeleton samples, as illustrated in Figure 4.3. In Figure 4.3 it is specifically indicated that a dynamic exemplar sample  $\mathbf{x}_t$  can be preceded by only  $\mathbf{x}_{t-1}$ , whereas a static skeleton sample  $\mathbf{p}_i$  can be preceded by multiple skeleton samples.



**Figure 4.3:** Including an unambiguous velocity feature in a static image is difficult as skeleton samples with more than one neighbour can be preceded by multiple skeleton samples.

It is therefore not possible to include an unambiguous velocity component at each skeleton sample in a static image (without introducing heuristic constraints.) As our HMM topology is directly computed from the connectivity of the static skeleton samples, it is not possible to include unambiguous velocity components in our first-order models either. (Recall that pre-recorded dynamic exemplars are matched to the HMM of a static image to compute the trajectory of the image. Thus, although we are able to include velocity components in the dynamic exemplar feature vectors, corresponding components must exist in our HMMs.) Hence, to include velocity/delta components in our HMMs we must find a way to model long-term dependencies.

Bengio and Frasconi [8], supported by the experiments of Abou-Moustafa et al. [1], investigated the effect of an HMM's topology on the ability of the HMM to learn context. They showed that the addition of hidden states with a sparse connectivity can increase the ability of a Markov

model to learn long-term dependencies and reduce the diffusion of context. The topology of our first-order HMM is ergodic with a sparse connectivity. When using second-order HMMs, we include extra states and the connectivity becomes even sparser in a natural way, as discussed in the next section. Thus, in accordance with Bengio and Frasconi, we improve the ability of the HMM to model context. The second-order models specifically allows us to include velocity components in our feature vectors.

## 4.3 Second-order HMMs and their first-order equivalents

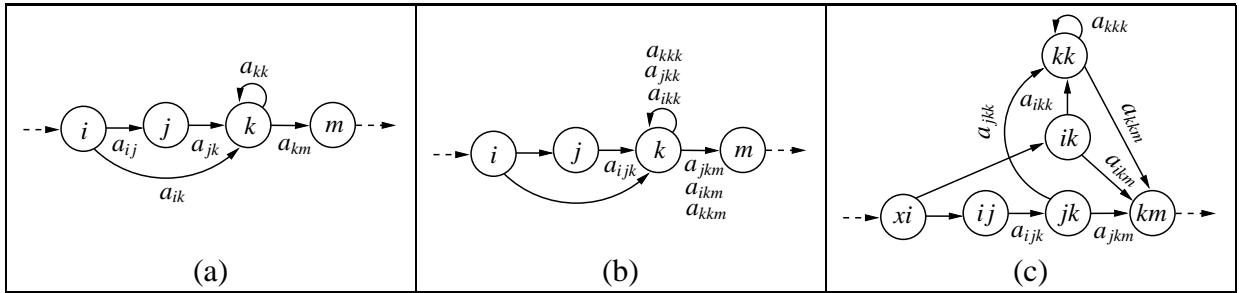
This section describes the most important characteristics of higher-order HMMs that can be exploited for our application. In Section 4.3.1, some fundamental concepts of higher-order HMMs are discussed. Section 4.3.2 provides a broad overview of the implication of higher-order HMMs for our application.

### 4.3.1 General higher-order HMM theory

In order to take past context into account, we use second-order HMMs. It has been shown that the transition probabilities of first-order HMMs only depend on the current state, so that  $a_{ij} = P(s_{t+1} = q_j | s_t = q_i)$ . The transition probabilities of second-order HMMs depend on the current and previous states. The probability of a transition from state  $j$  to state  $k$ , given that state  $j$  is preceded by state  $i$ , becomes  $a_{ijk} = P(s_{t+1} = q_k | s_{t-1} = q_i, s_t = q_j)$ . Second-order HMMs can then be reduced to first-order equivalents to simplify their implementation, by using the Order Reducing (ORED) algorithm [22, 23].

We illustrate these ideas with a hypothetical HMM in Figure 4.4. The HMM fragment in the figure forms part of a larger HMM. We only consider the transitions between the visible states. The second-order HMM in Figure 4.4(b) is formed by extending all transitions of the first-order HMM in Figure 4.4(a) to second-order connections (the order of the transitions is encoded in the subscripts of the transition probabilities.) Second-order connections depending on states outside of the HMM fragment are not shown.

The principle behind the ORED algorithm is to reduce an  $R$ th-order HMM to its  $(R - 1)$ th-order equivalent, by creating states for all pairs of connected states in the  $R$ th-order HMM. Applying this procedure recursively, an HMM of arbitrary order is reduced to its first-order equivalent [22, 23]. The first-order equivalent of the second-order HMM of Figure 4.4(b) is shown in Figure 4.4(c). In general, each state is now uniquely defined by its label, where each



**Figure 4.4:** Calculating the first-order equivalents of second-order HMMs using the ORED algorithm. (a) A first-order HMM expanded to (b) a second-order HMM, and (c) the first-order equivalent of (b).

state  $ab$  in Figure 4.4(c) was created from the connected pair  $ab$  in Figure 4.4(b). If numerical values are assigned to  $a$  and  $b$ , we refer to state  $ab$  as state  $(a,b)$  for the sake of simplicity, e.g. if  $a = 1$  and  $b = 10$ , we refer to state  $ab$  as state  $(1,10)$ . All the second-order transition weights can now be interpreted as first-order weights. The first-order states  $jk$  and  $km$ , e.g., are connected with transition weight  $a_{jkm}$ , so that  $a_{jkm}$  can be interpreted as a *first-order* transition probability. Likewise, the pairs  $kk, ik, ij, km$  and  $xi$  in Figure 4.4(b) are connected and become states  $kk, ik, ij, km$  and  $xi$  in Figure 4.4(c), respectively, where  $x$  can be any state connected to  $i$  via a dashed line. Thus, in general,  $M$  different pairs of connected states in the  $R$ th-order model result in approximately  $M$  states in the  $(R - 1)$ th model.

Higher-order HMMs result from enlarging the state context in the Markov order assumption (one of the two fundamental assumptions utilised in HMMs [22].) Strictly speaking, this leaves the state PDFs unaffected by the Markov order, i.e., each higher-order HMM state still has only one PDF. Reducing such an HMM to its first-order equivalent therefore also does not change the total number of PDFs. Note that the rightmost label  $b$  in  $ab$  is the PDF index of state  $ab$  so that some states *share* the same PDF. We refer to PDFs that are shared by more than one state as *tied PDFs*. State  $jk$ , e.g., is created in Figure 4.4(c) from the connected pair  $jk$  Figure 4.4(b). State  $jk$  inherits its PDF from state  $k$  in Figure 4.4(b) and also shares PDF  $k$  with states  $kk$  and  $ik$ . In general, a *predecessor*  $q_h$  of  $q_i$  is any state for which  $a_{hi} > 0$ . Note that by virtue of a second-order HMM's topology, it is guaranteed that each state is preceded only by states that share the same PDF. In general, we now let the leftmost index  $a$  in  $ab$  indicate that all the predecessors of state  $ab$  share PDF  $a$ . State  $kk$ , in Figure 4.4(c), e.g., is preceded only by states that share PDF  $k$ , i.e., states  $ik, kk$  and  $jk$ .

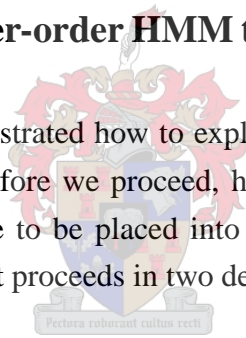
The order reduction significantly increases the number of states. An  $R$ th-order model with  $N$  states reduces to an equivalent first-order model with  $O(N^R)$  states. However, it should be noted that this expansion does not increase the number of free parameters. Tied PDFs are evaluated only once, and only the original number of higher-order transition probabilities need

to be considered. Therefore, the ORED algorithm does not affect processing requirements. It is shown by Du Preez [21] that memory requirements are not affected either. However, for our application we increase the memory requirements somewhat by not allowing the sharing of PDFs after the first-order equivalent models are derived: since the resulting first-order states have the same PDFs than their original higher-order states, they merely represent the higher-order states in different conditions. This *multiplicity* of states allows one to make the state PDFs context dependent, simply by not sharing them with other states. In this research we make fruitful use of the richness in modelling that this option presents. The computational cost depends on the transition probabilities, as discussed in Chapters 6 and 7. The computational cost of our proposed algorithm is manageable, as our transition probability matrix remains sparse. This avoids redundant calculations.

Without loss of generality, all the higher-order HMMs are represented by their first-order equivalents in the sections to follow.

### 4.3.2 Application of higher-order HMM theory to static handwritten scripts

In the sections to follow, it is illustrated how to exploit the flexibility of higher-order HMMs to model handwritten scripts. Before we proceed, however, some of the higher-order HMM concepts from Section 4.3.1 have to be placed into perspective. The derivation of the final HMM of a single-path static script proceeds in two defining steps:



1. **Deriving second-order HMMs.** Firstly, the second-order HMM  $\lambda'$  for our first-order HMM  $\lambda$  of a static script (from Section 4.2) is derived. Each second-order HMM  $\lambda'$  is then represented by its first-order equivalent  $\lambda''$  with  $N'$  states using the ORED algorithm. All the notation developed in Section 4.3.1 is therefore applicable so that an emitting state  $ij$  in  $\lambda''$  is generated from the connected pair  $ij$  in  $\lambda'$ , where  $i, j \in \{1, \dots, N\}$  and  $N$  is the number of states in  $\lambda$ . Let  $\zeta_{ij}$  be the index of the pair  $ij$  so that  $\zeta_{ij} \in \{1, \dots, N'\}$ . Recall from Section 4.2 that a unique PDF exists for each state in  $\lambda$  and that a skeleton sample is embedded in each PDF. Hence, for this application, the following information is available directly from the label  $ij$  that defines state  $ij$  uniquely: The PDF  $j$  is associated with state  $ij$  (as explained in Section 4.3.1), where skeleton sample  $\mathbf{p}_j$  is embedded in PDF  $j$  (as a result of our first-order HMMs.) Thus, it is always known which skeleton sample is inherited from  $\lambda$  by state  $ij$ . We have also explained in Section 4.3.1 that state  $ij$  is preceded only by states that share PDF  $i$ . Since skeleton sample  $i$  is embedded in PDF  $i$ , state  $ij$  can be preceded only by states that share skeleton sample  $i$ . Thus, in general, our second-order HMM  $\lambda''$  guarantees that all predecessors of an emitting state share the

same skeleton sample. Sections 4.4 and 4.5 illustrate the detail development of the first step of our algorithm.

2. **Altering the HMM PDFs.** In the second phase of our algorithm, we add a second component to the PDFs of  $\lambda''$  so that the PDF of each state is unique. We have explained in Section 4.3.1 that the number of states in a second-order HMM is increased when deriving its first-order equivalent. In this step, we exploit the multiplicity of states to make the PDFs context dependent. Recall that a state  $ij$  of  $\lambda''$  is uniquely defined by its double-indexed label, where the rightmost label  $j$  indicates which skeleton sample is associated with state  $ij$  and the leftmost label  $i$  indicates the predecessor skeleton sample of state  $ij$ . This makes it possible to include directional (normalised velocity) information. Hence, in this step the PDF  $j$  of state  $ij$  is relabelled as PDF  $ij$ , thereby defining it uniquely. Section 4.5 elaborates on this step of our algorithm.

## 4.4 HMM topology for line segments

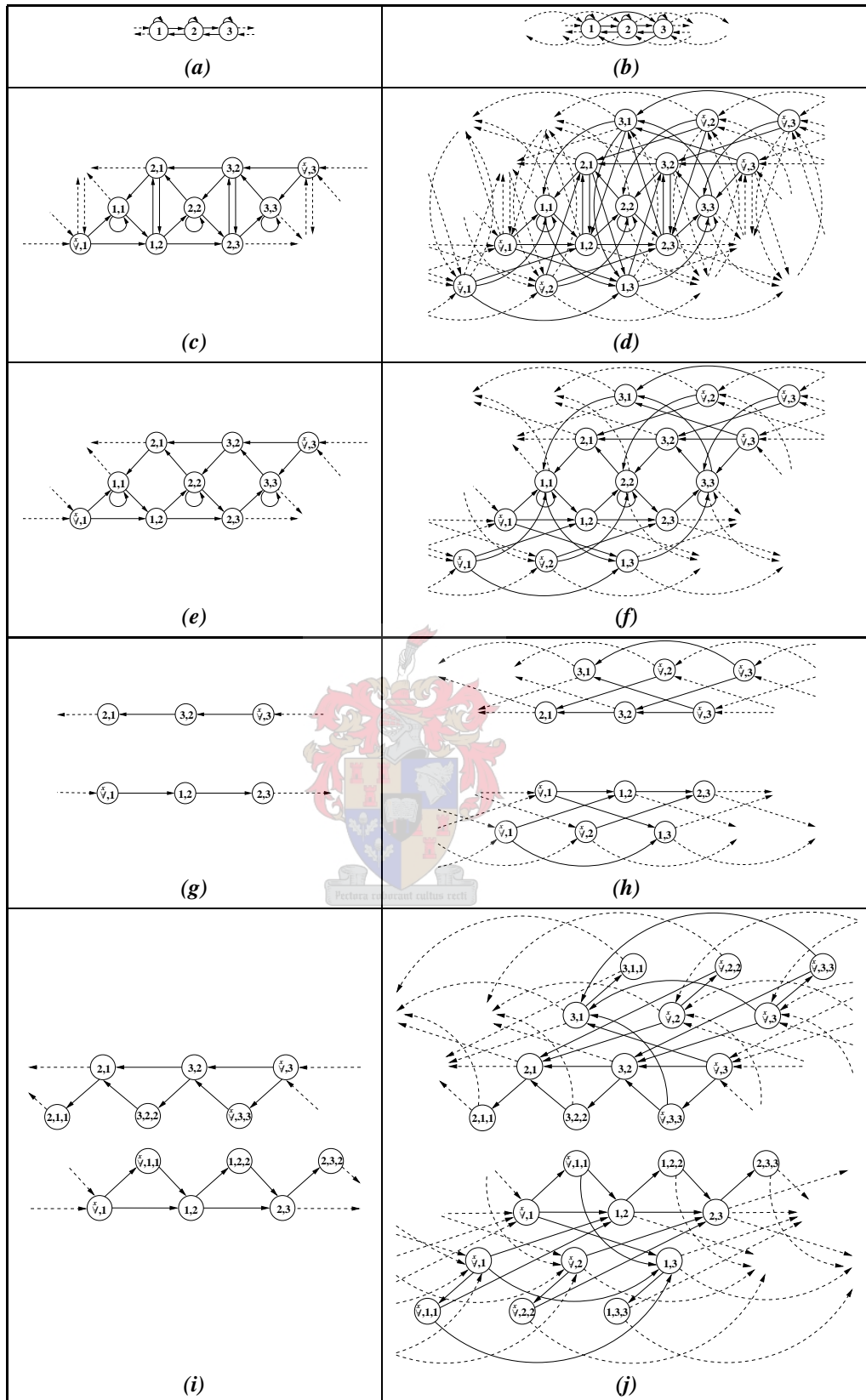
When unravelling a static image, the simplest parts are those without crossings or turning points, referred to as line segments. *Line segments* consist of connected segment points, where a *segment point* is a skeleton sample having only two skeleton neighbours.

For first-order HMMs it is necessary to have transition links connecting neighbouring states in both directions, since the direction of travel on a line segment is not initially available. This creates the problem that the pen's direction of motion can reverse at any segment point. We solve this by extending the first-order HMM of Section 4.2 to a second-order HMM, as described in Section 4.3. This introduces longer state dependencies, which enables the use of directional constraints.

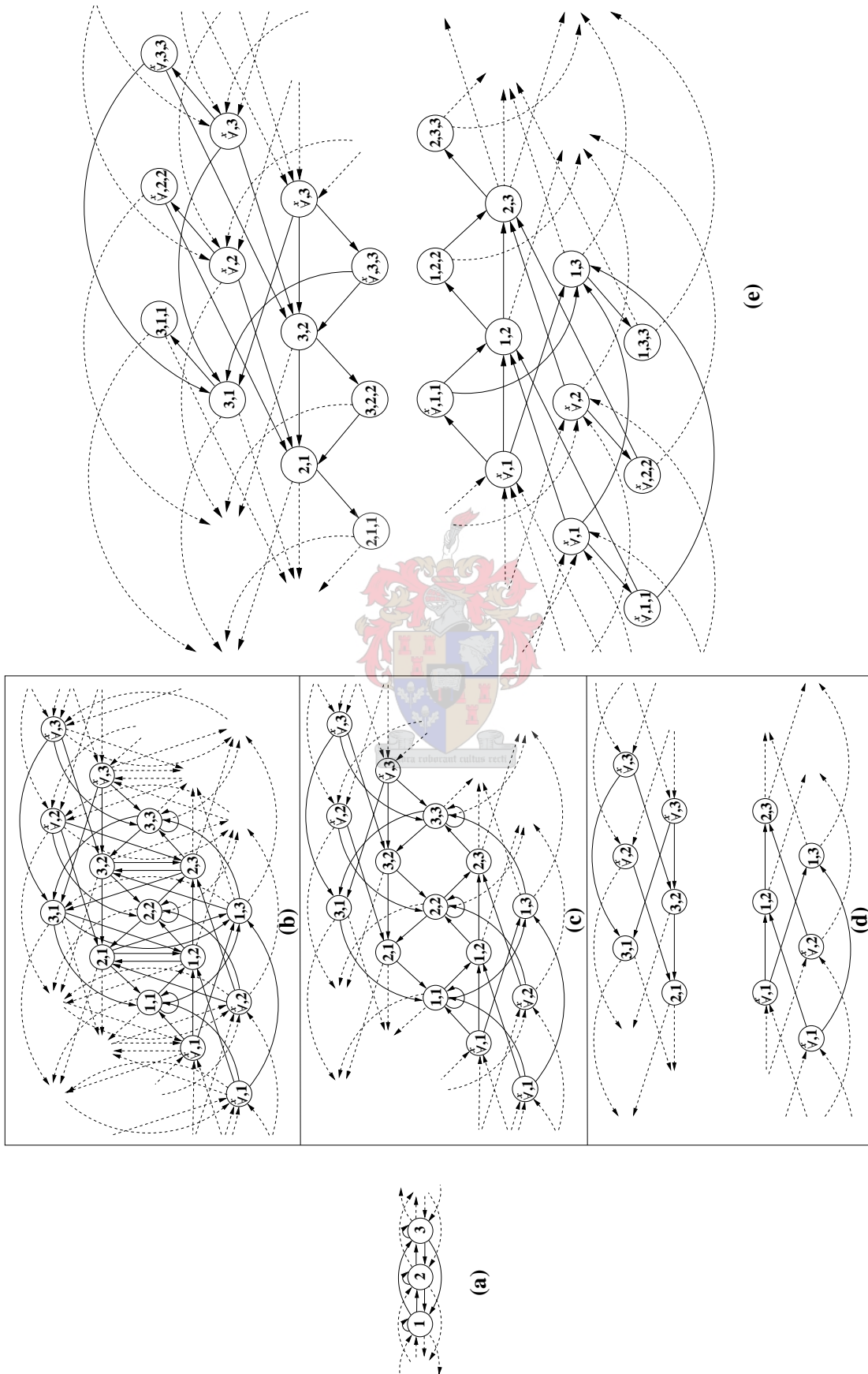
Figure 4.5(a) shows a simplified version of a first-order HMM for a line segment (skip-links, non-emitting states, and transition links connected to the non-emitting states are omitted.) The skip-links are added in Figure 4.5(b). For the sake of clarity, Figure 4.6 presents an enlarged version of the right-hand column of Figure 4.5.

One can now develop second-order HMMs for the topologies of Figures 4.5(a) and (b), as described in the previous section. The first-order equivalents of these second-order systems are shown in Figures 4.5(c) and (d), respectively.





**Figure 4.5:** The simplified HMM topology for a line segment in the left-hand column with a detailed version in the right-hand column when skip-links are added to our first-order HMM. (a)-(b) First-order HMMs. (c)-(d) Second-order HMMs. (e)-(f) Assigning the cost function. (g)-(h) Removal of self-loop states. (i)-(j) Inclusion of duration states.



**Figure 4.6:** The enlarged version of the right-hand column of Figure 4.5 illustrating the development of the HMM for a line segment.

As explained in Section 4.3.2, the first-order HMM has just as many emitting states as skeleton samples, whereas the second-order model has more than one state per sample. Thus,  $N' > M$ , representing different contexts in which a skeleton sample can be found. (Recall that  $N$  is the number of states in the first-order HMM  $\lambda$  of a static script, whereas  $N'$  is the number of states in the first-order equivalent  $\lambda''$  of the second-order HMM for  $\lambda$ .) Specifically, state  $ij$  in the second-order model is associated with skeleton sample  $j$  of the static script and is preceded only by states that share skeleton sample  $i$ . It is also worth noting the significant increase in model complexity in the right-hand column, due to the skip-links added in Figure 4.5(b). The symbol  $\forall, y$  is used in all the figures to indicate that state  $xy$  is preceded by states that share PDF  $x$  outside the figure, where  $x \in \{1, \dots, N\}$  so that the label  $xy$  defines state  $xy$  uniquely. Note that since all the states that precede state  $xy$  share PDF  $x$ , they also share skeleton sample  $x$ .

The next step is to enforce pen movement in one direction. Let  $\theta_{hij}$  be the angle between the two straight lines connecting points  $h$  to  $i$  and  $i$  to  $j$ , respectively. Then

$$\cos(\theta_{hij}) = \frac{(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}_i - \mathbf{p}_h)}{\|\mathbf{p}_j - \mathbf{p}_i\| \|\mathbf{p}_i - \mathbf{p}_h\|}, \quad (4.3)$$

where  $\|\cdot\|$  is the Euclidean distance norm and  $\mathbf{p}_h$ ,  $\mathbf{p}_i$ , and  $\mathbf{p}_j$  are the 2D coordinates of points  $h$ ,  $i$ , and  $j$ , respectively.

In order to encourage the system to follow the same direction along a line segment, the probability of a transition from state  $hi$  to emitting state  $ij$  is chosen as the cost function

$$a_{hij} = \begin{cases} \cos(\theta_{hij}), & \text{for } |\theta_{hij}| \leq 90^\circ \\ 0, & \text{for } |\theta_{hij}| > 90^\circ, \end{cases} \quad (4.4a)$$

where  $\cos(\theta_{hij})$  is defined by (4.3). Equation 4.4 is, however, not applied to links entering or leaving self-loop states, where *self-loop states* are states with self-loops, e.g., state (1,1) in Figure 4.5(d). Figures 4.5(e)-(f) show the HMMs in Figures 4.5(c)-(d) after links with zero probability have been removed, based on the cost function from (4.4).

Since the self-loop states are excluded from the cost function, it is still possible to turn around via them. An example from Figure 4.5(e) is the state sequence [(1,2), (2,3), (3,3), (3,2)] corresponding to the PDF/sample sequence (rightmost indexes) [2, 3, 3, 2]. In order to prevent this, self-loop states and all their connections (both entering and leaving them) are removed. Figures 4.5(g)-(h) show the HMMs in Figures 4.5(e)-(f) after this step.

Introducing skip-links for more elasticity leads to the configuration of Figure 4.5(h). We use the term *skip-link states* when referring to states in the second-order HMM that result from skip-links in the corresponding first-order HMM, e.g., state (3,1) in Figure 4.5(d) resulting from the skip-link leaving state 3 and entering state 1 in Figure 4.5(b). Skip-link states can compensate for situations in which the static image has more samples than the dynamic exemplar. Self-loop

states, on the other hand, can compensate for situations in which the dynamic exemplar has more samples than the static image. Since all self-loop states have been removed all emitting states are duplicated and each emitting state is allowed to enter its duplicated state. We refer to the duplicated states as *duration states*, where the duration states have the same destinations as the states they duplicate.

The above duration state concepts are illustrated in Figures 4.5(i)-(j). State (3,2,2), e.g., is the duration state of state (3,2) (their two leftmost indexes are the same, indicating that they are “partners”). These states also share PDF 2 and therefore skeleton sample 2 (rightmost index) and have the same destinations. Note that the *destinations* of states (3,2,2) and (3,2) are preceded only by states that share PDF 2. It should be noted how the two directions that a pen can follow on a line segment are completely disjoint within the HMM, so that it is not possible to change direction in the middle of a line segment. Additionally any skeleton sample is allowed to be repeated without allowing the pen to turn around abruptly, i.e, without a loss of context.

In general any duration state  $ijj$  is now uniquely defined, where for our application, state  $ijj$  is associated with *and* preceded by skeleton sample  $j$ . Additionally, it is known at state  $ijj$  that state  $ij$  (preceding state  $ijj$ ) is preceded only by states that share skeleton sample  $i$ . The inclusion of this additional information can be interpreted as a third-order occurrence *only* at the duration states of our HMM. This additional knowledge is especially useful when we extend our PDFs, as described in Section 4.6. With the exception of duration states, our HMM employs only past context available from second-order transitions. Thus, we still refer to our HMMs as second-order HMMs in the sections to follow.

In this section we have discussed the topology of states associated with segment points. Specific transition weights are presented at the end of Chapter 5. Next we discuss the topology of states where the pen is allowed to change direction abruptly.

## 4.5 HMM topology for crosspoints and endpoints

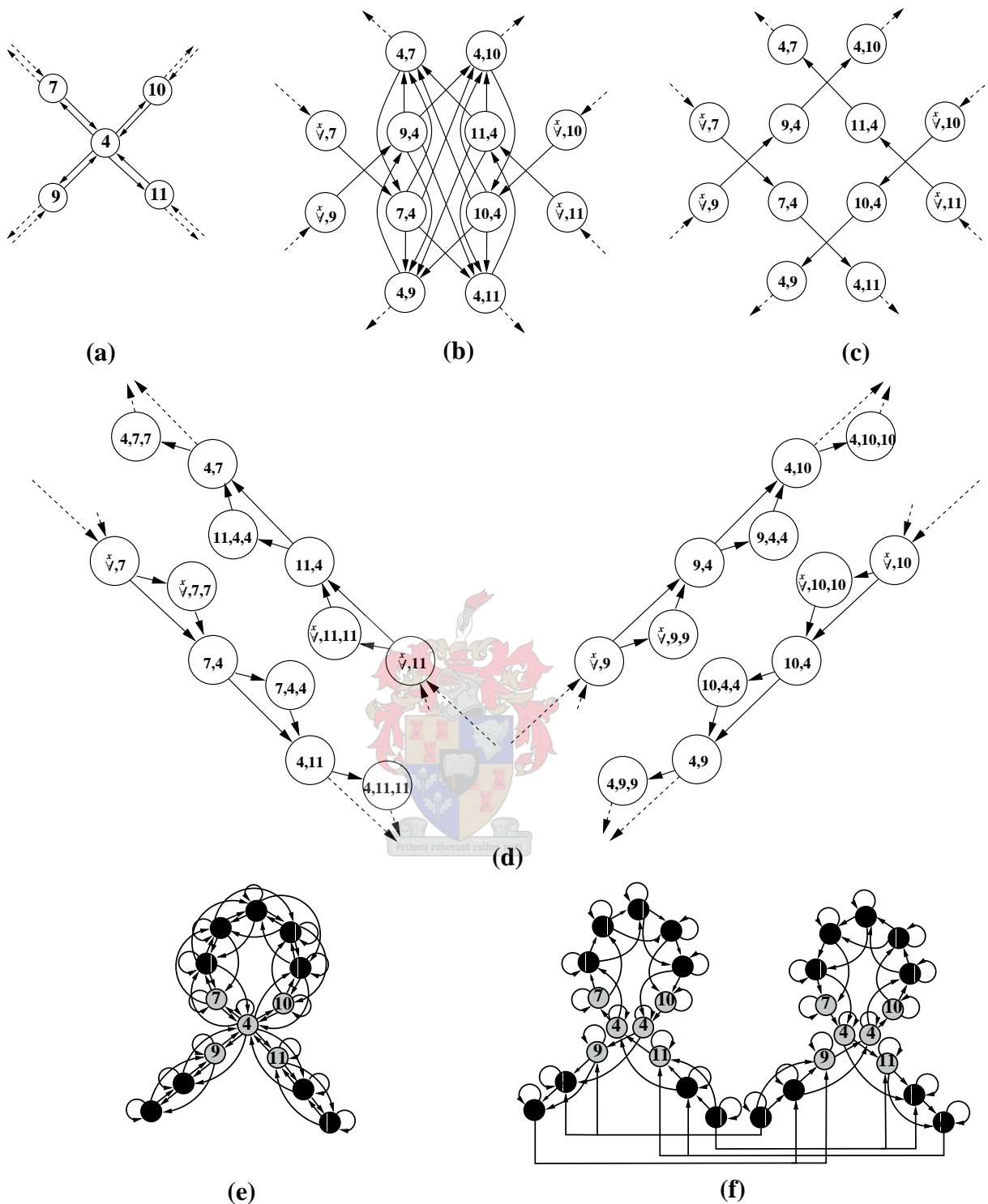
To enable the pen to immediately recross a line or suddenly change direction we allow it to turn around or change direction abruptly at states associated with endpoints and crosspoints. Recall that Section 3.1 has defined *endpoints* as skeleton samples having one neighbour, whereas *crosspoints* are skeleton samples having more than than two neighbours. The main difference between states associated with segment points and states associated with crosspoints and endpoints is that the direction constraint of (4.4) is not enforced for crosspoint and endpoint states. Instead, traversal to any immediate or skip-link state neighbour is allowed from a state associated with a crosspoint or endpoint. This ensures that it is possible to change direction abruptly,

or even to turn around, at these states.

Some situations, however, involve simple crossings, where it is easy to follow line directions that enter and leave the crosspoint. It is straightforward to unravel such crossings. If the line thickness is uniform and line directions are smooth near such a simple crosspoint, and no more than two lines cross each other at a single point, it is unlikely that the pen has passed through that region multiple times. Let us, for the moment, assume that we can identify such a simple crossing and label its associated sample as  $i$ . A simplified first-order model (excluding non-emitting states, skip-links, and self-loops) is shown in Figure 4.7(a), where  $i = 4$ . It should be noted that first-order HMMs are not able to model past context, so that the transition probabilities for state  $i$  have to allow access to and from any of its four neighbouring states. The situation is different with second-order HMMs. The first-order equivalent of the second-order model for Figure 4.7(a) is shown in Figure 4.7(b). As the transition probabilities depend on predecessor states that share the same skeleton sample, it is straightforward to follow lines through the crosspoint. At a simple crossing, one can then detach the two lines that cross, by setting the appropriate transition probabilities to zero, as shown in Figure 4.7(c). To do this, however, we need to be able to identify such simple crossings.

With the crosspoint labelled as  $i$ , we label the four neighbouring coordinates clockwise, in order, as  $a$ ,  $b$ ,  $c$ , and  $d$ . The idea is to identify whether the sequenced samples  $[a, i, c]$  and  $[b, i, d]$  are intersecting lines. In Figure 4.7(b), e.g., the coordinates (rightmost indexes) are labelled as  $a = 7$ ,  $b = 10$ ,  $c = 11$ , and  $d = 9$ . We consider only crossings where  $a$ ,  $b$ ,  $c$ , and  $d$  are all segment points, having only one other skeleton neighbour besides  $i$ . Let  $x$  be the other skeleton neighbour of  $a$  and  $y$  be the other skeleton neighbour of  $c$ . We now calculate three angles,  $\theta_{xai}$ ,  $\theta_{aic}$ , and  $\theta_{icy}$ , using (4.3). If  $|\theta_{aic}| \leq 10^\circ$ ,  $|\theta_{xai}| \leq 30^\circ$ , and  $|\theta_{icy}| \leq 30^\circ$ ,  $[a, i, c]$  is considered a straight line. If, likewise  $[b, i, d]$  also proves to be a straight line, the crossing is considered a simple crossing. The second-order HMM provides the necessary context to extract directions and decouple the two lines  $[a, i, c]$  and  $[b, i, d]$  so that the two intersecting lines can both be traversed in one direction or the other. Direction is now maintained through the crossing, and the inclusion of duration states provides the necessary flexibility, as shown in Figure 4.7(d).

An important point should be brought to light, regarding our skeletonisation and resampling. Final skeletons are smoothed using Chaikin's corner-cutting subdivision method [10, 49]. Since only line segments are smoothed and resampled, endpoints and crosspoints are not affected by this procedure. When testing whether crosspoint  $i$  is a simple crosspoint, the original intersecting lines  $[a, i, c]$  and  $[b, i, d]$ , i.e., the two lines *before* smoothing are used. This allows a more reliable estimate of how close the lines are to being straight. Thus, for each immediate and skip-link neighbour of a simple crosspoint state, two skeleton samples are in reality stored—those before smoothing and those after smoothing.



**Figure 4.7:** The HMM topology for endpoints and crosspoints. (a) The simplified first-order HMM for a crosspoint (excluding non-emitting states, skip-links, and self-loops), with (b) the first-order equivalent of its second-order counterpart; (c) lines decoupled at the crossing by removing links from (b); and (d) duration states included in (c). (e) Possible pen trajectories for Figure 4.1(b) using our first-order HMM, and (f) using the second-order HMM with detachment of the intersecting lines at the crosspoint. It should be noted that graphs (e) and (f) are not intended as HMMs, but as representations of allowed pen trajectories, with skeleton samples as nodes and possible pen motions as arrows.

A more detailed example is shown in Figures 4.7(e)-(f), indicating possible pen trajectories that can be extracted from Figure 4.1(b) if first-order and second-order HMMs are used and detaching the simple crossing at skeleton sample 4 in Figure 4.7(f). For the sake of clarity, Figures 4.7(e)-(f) omit the fact that a pen trajectory can start and end at any sample (filled circle.) Self-loop symbols are used to indicate duration modelling in Figure 4.7(f). The corresponding states for the grey skeleton samples in Figure 4.7(e) are shown in Figure 4.7(a), whereas the corresponding states for the grey skeleton samples in Figure 4.7(f) are shown in Figure 4.7(d). Note that the two lines [7, 4, 11] and [9, 4, 10] are detached in Figure 4.7(f). It is interesting to derive the number of first-order and second-order HMM states and non-zero transition probabilities for the signature in Figure 4.1(b), that allow the choices of possible pen motions shown in Figures 4.7(e) and (f):

1. The signature in Figure 4.1(b) has 16 states in its first-order HMM and 107 states in its final second-order HMM (including non-emitting states.)
2. The signature in Figure 4.1(b) has  $\frac{92}{256}$  (36 percent) non-zero transition probabilities in its first-order HMM. This is reduced to  $\frac{428}{11449}$  (3.7 percent) in its final HMM. It should be noted that transition links leaving the non-emitting initial state and entering the non-emitting terminating state are included in this computation.

We can conclude from the above statistics that the final HMM is notably sparser and has more states than its first-order counterpart. Thus, in accordance with Bengio and Frasconi [8], our second-order HMM has a better ability to model long-term dependencies compared to our first-order HMM, thereby improving our pen trajectory estimation algorithm’s ability to take more context into account; see Section 4.2 for further detail.

## 4.6 HMM PDFs

In our first-order HMM, a skeleton sample is associated with each state. For simplicity, the PDF associated with each state has so far reflected only information about the position variations of the static image. When unravelling a static image, the direction of pen motion at each coordinate is also important. Knowledge of pen direction allows us to match not only position coordinates, but also local directions in a dynamic exemplar, thus providing additional context. At this point, we have designed the second-order HMM  $\lambda''$  of a single-path static script with  $N'$  states. To include pen directional information in our PDFs we have to “untie” the PDFs of  $\lambda''$ , as discussed in Section 4.3.2. Hence, we relabel our PDFs so that the PDF component of state  $ij$  that reflects the pen position is given by  $\mathcal{N}(\boldsymbol{\mu}_{ij}^P, \sigma_P)$ , where we defined  $\boldsymbol{\mu}_{ij}^P = \mathbf{p}_j$ . For a

duration state  $\mathbf{p}_{ijj}^P = \mathbf{p}_j$  in  $\mathcal{N}(\mathbf{\mu}_{ijj}^P, \sigma_P)$ . However, in our second-order HMM, each state  $ij$  has a single predecessor skeleton sample  $i$ , enabling the use of a directional feature. Except for the duration states, the existing (positional) Gaussian PDFs are now extended with an independent directional component. The mean of this component takes the form of (4.1) with

$$\mathbf{\mu}_{ij}^V = \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|}, \quad (4.5)$$

for  $i, j \in \{1, \dots, N\}$  and  $\zeta_{ij} \in \{1, \dots, N'\}$  where  $\zeta_{ij}$  is the index of the pair  $ij$  and  $\mathbf{\mu}_{ij}^V = (0, 0)$  if state  $ij$  is preceded by the non-emitting initial state. The directional PDF is abbreviated as  $\mathcal{N}(\mathbf{\mu}_{ij}^V, \sigma_V)$ . A duration state  $ijj$ , associated with skeleton sample  $j$ , is preceded by state  $ij$  which is also associated with skeleton sample  $j$ , so that we cannot compute the directional feature described by (4.5) for  $\mathcal{N}(\mathbf{\mu}_{ijj}^V)$ . Recall that we have introduced a third-order occurrence at duration states (described in Section 4.4) so that it is known at state  $ijj$  that state  $ij$  is preceded only by states that share skeleton sample  $i$ . We now exploit this additional information and let  $\mathbf{\mu}_{ijj}^V = \mathbf{\mu}_{ij}^V$ .

The two components that constitute each PDF are assumed to be statistically independent [61]. They reflect the typical correspondences between the coordinates (position and direction of pen motion) of the static image and dynamic exemplar. It should be noted that the directional feature described by (4.5) is frequently used in first-order HMMs of on-line character recognition and signature verification applications, where each pen position has a unique previous position. In our application, each coordinate in the skeleton of a static image has one or more neighbours and we have no prior knowledge to choose appropriately. Second-order HMMs can model longer dependencies, effectively enforcing a single previous coordinate for each state. Thus, we are able to include an unambiguous directional feature in each state PDF. All the final HMM parameters for a single-path static script are presented in Sections 4.7 and 4.8.

## 4.7 A summary of empirical HMM parameters

**Key concepts.** Since our final HMMs are in reality first-order HMMs by virtue of the ORED algorithm, we present them hereafter with the first-order HMM notation developed in Section 4.2. Hence, the HMM of a static script consists of  $N$  states  $\mathbf{q} = \{1, \dots, N\}$ . Note that we only revert to this notation for the sake of simplicity (ignoring the double indexing.) This can only be done if the following key concepts are intact: Each state  $i$  in the HMM of a single-path static script is associated with a skeleton sample and is preceded only by states that share the same skeleton sample. Thus, when referring to state  $i$ , we hereafter assume that its associated skeleton sample is known, as well as its unique predecessor skeleton sample. The transition weights of state  $i$  depend on whether the skeleton sample associated with state  $i$  is a segment point, a turning point



or a crosspoint. State  $i$  has its own PDF  $i$  which embeds directional and positional information. These are the only concepts that have to be remembered hereafter to proceed with our HMM development. The HMM parameters are also listed in accordance with these ideas.

All the HMM parameters for a single-path static script are designed specifically for our application. The parameters were optimised on a single signature from a different database (the Dolfing [20, 82] database) and are therefore independent of the test set. The relevant empirically determined values used in our system are listed as follows (all transition probabilities leaving a state are normalised to sum to 1.0 after these values are assigned):

- $a_{ij}^E = 1.0$ : Probability of a transition from endpoint state  $i$  to state  $j$ .
- $a_{ij}^I = 1.0$ : Probability of a transition from non-emitting initial state  $i$  to state  $j$ .
- $a_{ij}^C = 1.0$ : Probability of a transition from crosspoint state  $i$  to state  $j$ .
- $a_{ij}^{SS} = 0.05$ : Probability of a transition from segment point state  $i$  to its duration state  $j$ .
- $a_{ij}^{ST} = 0.05$ : Probability of a transition from segment point state  $i$  to the terminating non-emitting state  $j$ .
- $a_{ij}^S = \cos(\theta_{xyz})$ ,  $|\theta_{xyz}| \leq 90^\circ$ : Probability of any other transition from state  $i$  associated with segment point  $y$ , where skeleton sample  $x$  is the unique predecessor sample of state  $i$  and skeleton sample  $z$  is associated with state  $j$  (see (4.4).) To prevent numerical instabilities,  $a_{ij}^S$  is set to 0.05 if  $a_{ij}^S < 0.05$ . This corresponds to cases where the angle  $|\theta_{xyz}| \geq 87.1^\circ$ .
- $\boldsymbol{\mu}_i^P = \mathbf{p}_y$ : The mean of the position PDF component (a spherical Gaussian PDF described by (4.1)) of state  $i$  associated with skeleton sample  $y$ .
- $\boldsymbol{\mu}_i^V = \frac{\boldsymbol{\mu}_i^P - \mathbf{p}_x}{\|\boldsymbol{\mu}_i^P - \mathbf{p}_x\|}$ : The mean of the directional PDF component (a spherical Gaussian PDF described by (4.1)) of state  $i$  with predecessor skeleton sample  $x$ ; see (4.5). Note that for duration states  $\mathbf{p}_x$  is the predecessor skeleton sample of the state that precedes state  $i$ .
- $\sigma_P = 17$ : Standard deviation (in pixels) quantifying similarities between pen positions. This constrains the distance between points in the static image and dynamic exemplar.
- $\sigma_{VI} = 2$ : Standard deviation that quantifies similarities between local line directions if a state is preceded by the non-emitting initial state.
- $\sigma_V = 0.2$ : A tight standard deviation that quantifies similarities between local line directions if a state is preceded by an emitting state.

## 4.8 Writer-specific HMM training

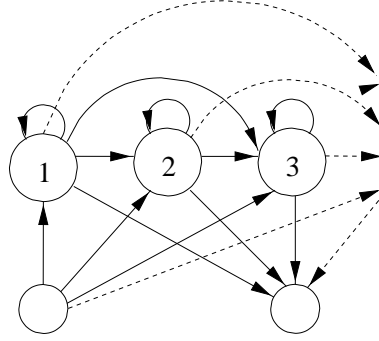
Since the associated skeleton sample and predecessor skeleton sample of each state are inherent in the means of the positional and directional PDF components, we do not want to train them.

Thus, for our PDFs, we train only  $\sigma_p$  and  $\sigma_v$  (see Section 4.7) to calculate  $\sigma'_p$  and  $\sigma'_v$  for each individual. We refer to this training scheme as *writer-specific training*. These parameters can be seen as indications of the general directional and positional variations of a specific individual's handwriting. The parameters  $\sigma'_p$  and  $\sigma'_v$  can therefore be interpreted as biometric measurements of an individual and may also be useful for other applications, e.g., in an off-line signature verification system.

Since we have a PDF for each state, many transition links and more than one state associated with each skeleton sample, there are not enough training data to estimate all the PDF and transition weight parameters. Thus, data sparsity is a serious problem when using our current HMM structure. Hence, we utilise the pre-recorded dynamic exemplars as follows: Firstly, an HMM for each dynamic exemplar of a specific individual is constructed. These HMMs and our HMMs for static scripts have certain characteristics in common. However, since the dynamic exemplar samples are not randomly ordered, a first-order HMM with a left-to-right topology is sufficient for our purpose. Note that all states in an HMM with a *left-to-right topology* are numbered in an increasing order so that  $a_{ij} = 0$  for  $j < i$ . In the next step, we match the HMM of a dynamic exemplar to the remaining dynamic exemplars of the same individual, in order to estimate the values for  $\sigma'_p$  and  $\sigma'_v$ . Thus, if there are 14 pre-recorded dynamic exemplars for a specific individual, 14 left-to-right first-order HMMs are constructed. For each of these HMMs, the remaining 13 dynamic exemplars are matched to it. The information inherent in the final 182 state sequences are then joined to estimate  $\sigma'_p$  and  $\sigma'_v$  for the individual. Some further issues regarding training are discussed in Section 7.2.3.

For training, we treat the dynamic exemplars as single-path trajectories, i.e., zero-pressure samples are removed. A first-order HMM is then derived for each dynamic exemplar  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T]$ . Similar to the HMMs for static scripts, an emitting state is associated with each dynamic exemplar sample  $\mathbf{x}_t$ , i.e.,  $\mathbf{q} = \{q_1, q_2, \dots, q_T\}$ . However, the topology can be greatly simplified, as the dynamic exemplar samples are in sequence. The non-emitting initial state is connected to all the emitting states with equally weighted transition links. In correspondence with the HMM's left-to-right topology, the state associated with  $\mathbf{x}_t$  can reach any other state in the forward direction, i.e., states corresponding to  $[\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \dots, \mathbf{x}_T]$ . Self-loops are included at all the emitting states. All transition links are equally weighted by assigning a value of 1.0 to them, except for self-loops, which are weighted with a value of  $1 \times 10^{-6}$ . All the transition probabilities from a state are then normalised to sum to 1.0. The topology for the dynamic exemplar HMM is illustrated in Figure 4.8. States that are indicated by the larger circles are emitting states, where the numbers represent the indexes of the states as well as their associated dynamic exemplar sample indexes. The dashed lines render transition links to and from states that are not shown. The smaller blank circles represent the non-emitting initial and terminating states. All states are connected to these non-emitting states. Note that the HMM depicted in

Figure 4.8 is much simpler than the HMM in Figure 4.2.



**Figure 4.8:** Deriving the first-order HMM from a dynamic exemplar. Non-emitting states (small circles) and emitting states (big circles) are shown. All states are connected with transition links (arrows), where dashed lines indicate links to states other than the ones that are shown. The numbers represent state indexes as well as their associated dynamic exemplar sample indexes.

Before constructing a PDF for each state, the dynamic exemplars are normalised as follows (note that this normalisation is only valid for the training scheme described in this section): The first two components of each dynamic feature vector  $\mathbf{x}_t$  form a sub-vector  $\mathbf{x}_t^{1,2}$  describing the dynamic pen position. The third and fourth direction components (normalised velocity) are  $\mathbf{x}_t^{3,4} = (\mathbf{x}_t^{1,2} - \mathbf{x}_{t-1}^{1,2}) / \|\mathbf{x}_t^{1,2} - \mathbf{x}_{t-1}^{1,2}\|$  with  $\mathbf{x}_1^{3,4} = (0, 0)$ . The fifth component  $x_t^5$  is a curvature component which measures the angle between two successive lines, where

$$x_t^5 = \frac{(\mathbf{x}_t^{1,2} - \mathbf{x}_{t-1}^{1,2}) \cdot (\mathbf{x}_{t+1}^{1,2} - \mathbf{x}_t^{1,2})}{\|\mathbf{x}_t^{1,2} - \mathbf{x}_{t-1}^{1,2}\| \cdot \|\mathbf{x}_{t+1}^{1,2} - \mathbf{x}_t^{1,2}\|} \quad (4.6)$$

$$= \frac{\mathbf{x}_t^{3,4} \cdot \mathbf{x}_{t+1}^{3,4}}{\|\mathbf{x}_t^{3,4}\| \cdot \|\mathbf{x}_{t+1}^{3,4}\|}, \quad (4.7)$$

$$(4.8)$$

where  $t = [2, \dots, T-1]$  and  $x_1^5 = x_T^5 = 0$ .

Each PDF associated with an emitting state  $i$  within the HMM  $\lambda_j$  (derived from the dynamic exemplar  $\mathbf{X}_j$ ) consists of three components: the first two components are the same as the two PDF components associated with an emitting state in the HMM of a static script. Thus,  $\boldsymbol{\mu}_{i,j}^P = \mathbf{x}_{t,j}^{1,2}$  and  $\sigma_P = 17$  in  $\mathcal{N}(\boldsymbol{\mu}_{i,j}^P, \sigma_P)$ , where the subscript  $(i, j)$  generalises all the above notation by indicating that state  $i$  is associated with  $\mathbf{X}_j$ . The sample  $\mathbf{x}_{t,j}$  at instance  $t$  of  $\mathbf{X}_j$  is also identified by the subscript  $(t, j)$ . Following the same procedure for the directional component  $\mathcal{N}(\boldsymbol{\mu}_{i,j}^V, \sigma_V)$ ,  $\sigma_{V1} = 2$ ,  $\sigma_V = 0.2$  and  $\boldsymbol{\mu}_{i,j}^V = \mathbf{x}_{t,j}^{3,4}$ . The third independent component of the PDF associated with state  $i$  is also a spherical Gaussian  $\mathcal{N}(\mu_{i,j}^A, \sigma_A)$  with  $\mu_{i,j}^A = x_{t,j}^5$  for  $t = [1, \dots, T_j]$ ,  $\sigma_A = 0.2$  for  $t = [2, \dots, T_j - 1]$  and  $\sigma_A = 2$  for  $t \in \{1, T_j\}$ , where  $T_j$  is the number of samples in  $\mathbf{X}_j$ . Let

$f_{i,j}^P(\mathbf{x}_{t,k}^{1,2})$  be the positional PDF component  $\mathcal{N}(\boldsymbol{\mu}_{i,j}^P, \sigma_P)$  evaluated at  $\mathbf{x}_{t,k}^{1,2}$ . Likewise, let  $f_{i,j}^V(\mathbf{x}_{t,k}^{3,4})$  be the directional PDF component  $\mathcal{N}(\boldsymbol{\mu}_{i,j}^V, \sigma_V)$  evaluated at  $\mathbf{x}_{t,k}^{3,4}$  and let  $f_{i,j}^A(x_{t,k}^5)$  be the curvature PDF component  $\mathcal{N}(\mu_{i,j}^A, \sigma_A)$  evaluated at  $x_{t,k}^5$ . Since these PDF components are assumed to be independent, the joint observation PDF of state  $i$  evaluated at feature vector  $\mathbf{x}_{t,k}$  is given by  $f_{i,j}(\mathbf{x}_{t,k}) = f_{i,j}^P(\mathbf{x}_{t,k}^{1,2})f_{i,j}^V(\mathbf{x}_{t,k}^{3,4})f_{i,j}^A(x_{t,k}^5)$ , for  $t = [1, \dots, T_k]$ ,  $i \in \{1, \dots, T_j\}$ ,  $j, k \in \{1, \dots, K\}$ , where  $T_k$  is the number of samples in  $\mathbf{X}_k$ ,  $T_j$  is the number of samples in  $\mathbf{X}_j$  (from which the HMM is constructed) and  $K$  is the number of dynamic exemplars for a specific individual. The observation PDFs and the transition weights of the HMM  $\lambda_j$ , derived from the dynamic exemplar  $\mathbf{X}_j$ , are now completely defined.

For each of the  $K$  dynamic exemplars of an individual, an HMM  $\lambda_j$ , as described above, is derived. All the other  $K - 1$  dynamic exemplars of the same individual are matched to  $\lambda_j$  using the Viterbi algorithm [16]. The result is an optimal state sequence  $\mathbf{s}_j = [s_1^j, s_2^j, \dots, s_{T_k}^j]$ . The parameters  $\sigma'_V$  and  $\sigma'_P$  are re-estimated by the method of *maximum likelihood* (ML) [68]. However, we do not update the PDF means and only one training iteration is invoked. The total number of observations after computing  $\lambda_j$  for each dynamic exemplar  $\mathbf{X}_j$  and matching the other  $K - 1$  dynamic exemplars to  $\lambda_j$  is computed by:

$$N_T = \sum_{j=1}^K \sum_{k=1, k \neq j}^K T_k, \quad (4.9)$$

where  $T_k$  is the number of samples from  $\mathbf{X}_k$  which is matched to  $\lambda_j$ . The re-estimated position variance  $\sigma'_P$  is now calculated by

$$\sigma'_P = \frac{1}{N_T - 1} \sum_{j=1}^K \sum_{k=1, k \neq j}^K \sum_{t=1}^{T_k} (\mathbf{x}_{t,k}^{1,2} - \boldsymbol{\mu}_{s_t^j, j}^P)^2, \quad (4.10)$$

where  $\mathbf{x}_{t,k}^{1,2}$  is the pen position at instance  $t$  in  $\mathbf{X}_k$  and  $\boldsymbol{\mu}_{s_t^j, j}^P$  is the mean of the position PDF component at state  $s_t^j$  in  $\lambda_j$ . Likewise, the re-estimated directional variance  $\sigma'_V$  is calculated by

$$\sigma'_V = \frac{1}{N_T - 1} \sum_{j=1}^K \sum_{k=1, k \neq j}^K \sum_{t=1}^{T_k} (\mathbf{x}_{t,k}^{3,4} - \boldsymbol{\mu}_{s_t^j, j}^V)^2. \quad (4.11)$$

Although the above training scheme is computationally expensive, it is still feasible as it needs to be employed only once (after the recording of the dynamic exemplars.) The effect of this training scheme on our system is described in Section 6.3.5, and related issues are discussed in Section 7.2.3. In the sections that follow  $\sigma'_V$  and  $\sigma'_P$  replace  $\sigma_V$  and  $\sigma_P$  from Section 4.7. However, the rest of the parameters remain unchanged.

## 4.9 Summary

This chapter has described the issues that need to be addressed when designing an HMM for a single-path static script, so that the pen trajectory of the script can be estimated from the HMM. The design of our HMM is summarised as follows:

1. A first-order HMM is derived from the skeleton of a static script, where each skeleton sample is associated with an emitting state. The topology of the first-order HMM allows a pen trajectory to start and end at any skeleton sample and allows the pen to turn around at any skeleton sample. Skip-links and self-loops create the necessary flexibility, so that corresponding dynamic exemplar curves and static skeleton curves can be compared even though the curves have different numbers of samples. The pen is constrained to move continuously within a three-pixel range (to an immediate or skip-link neighbour.) The problem with our first-order HMM is, however, that it lacks the necessary context to model long term dependencies accurately.
2. Further context is provided by expanding the first-order HMM to its second-order equivalent. This second-order HMM has exactly the same characteristics as the first-order HMM, except that the range of pen motions is constrained when the pen traverses line segments. A second directional PDF component is also included at each state and a more advanced duration modelling is accomplished. To include even more context, simple crossings, where only two straight lines intersect, are detached. Our skeletonisation scheme holds the benefit that regional information is available when critical decisions has to be made at possible simple intersections.
3. A training scheme that partially compensates for geometric variations, while compensating for data sparsity, has been employed. This training scheme specifically estimates only two parameters from the pre-recorded dynamic exemplars for a specific individual. These parameters may be useful as biometric measurements of a specific individual.

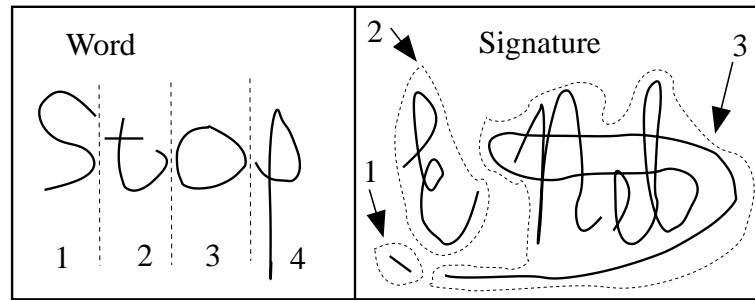
## Chapter 5

### The HMM for a multi-path static script

Up to this point, only single-path static scripts have been considered. In practice, static scripts often consist of more than one single-path trajectory, where the pen is lifted and moved to different positions between the single-path trajectories. This causes several problems that need to be addressed. Three major problems are:

- **Problem 1:** The accurate identification of the image parts/segments so that each image segment corresponds to a single-path trajectory. Thus, an accurate image segmentation had to be employed.
- **Problem 2:** The estimation of the sequence in which the different image segments were produced. Thus, the order of the image segments must be computed.
- **Problem 3:** The estimation of the pen trajectory of each image segment. As each image segment corresponds to a single-path trajectory (from Problem 1), all the techniques described in Chapter 4 are applicable to each segment.

In simple cases, e.g., a sequence of disconnected letters, one can associate each disconnected letter with a single-path trajectory. Unfortunately this approach fails if the single-path trajectories are recrossed—more often the case than not. Thus, there is mostly not a one-to-one correspondence between the number of *disconnected* image segments and the number of single-path trajectories that constitute a static script. Figure 5.1 illustrates these concepts, where the off-line representatives of the word “stop” and a signature are depicted. If each disconnected part corresponds to a single-path trajectory, the indicated segmentation results, where all the disconnected parts are encompassed by numbered dashed lines. Note that even in the simple case of the word “stop”, an error occurs using this segmentation, as the “t” consists of two single-path trajectories and not one.



**Figure 5.1:** A typical segmentation to identify disconnected parts in static scripts that correspond to single-path trajectories.

If it is assumed that we have segmented the static scripts, as indicated in Figure 5.1, the second problem is to determine in which order the disconnected parts occurred. In this example, it specifically becomes clear why most existing methods assume that cursive handwriting proceeds in a top-bottom-left-right-fashion. Accordingly, the sequence of the disconnected segments constituting the word “stop” is [1,2,3,4]. One can then estimate the single-path pen trajectory of each disconnected part using the methods described in Chapter 4. This strategy is, however, not necessarily sufficient for signatures, as multiple crossings occur, and some trajectories may proceed from right to left, e.g., segment 1 and the last part of segment 3 of the signature shown in Figure 5.1.

In practice, it is therefore not possible to rely on a prior “hard” segmentation of static scripts before estimating their pen trajectories. Again, the necessary information can be extracted from the pre-recorded dynamic exemplars, where non-zero-to-zero pen-pressure transitions indicate pen-up events. Note that pen-up events occur at the terminating positions of single-path trajectories, and therefore indicate the transition points between different single-path trajectories. However, even the dynamic exemplars do not provide unambiguous information. Due to the typical pen-sequence variations of handwritten signatures, the number of single-path trajectories that constitute the dynamic exemplars may vary for the same individual.

In this chapter it is shown how the above problems are addressed using hierarchical hidden Markov models. Hence, the HMMs for single-path static scripts are generalised to HMMs for multi-path static scripts. Our notation is generalised accordingly in Section 5.1. Section 5.2 shows how to exploit the dynamic exemplar pen pressure information to identify pen-up events. Accordingly, the HMM for a static script is manipulated to identify the single-path trajectories that constitute the script. Section 5.3 deals with the special case where unexpected disconnections occur in a static script. Section 5.4 describes how to match a dynamic exemplar to the HMM of a static script. The pen trajectory of the script is then derived from an optimal state sequence. Some concluding remarks are made in Section 5.5.

## 5.1 Hierarchical HMMs

A hierarchical hidden Markov model (HHMM) is a structured multi-level stochastic process [25]. Fine et al. [25] show that HHMMs extend HMMs, as an HHMM is also an HMM and a state in an HHMM can be an HHMM itself. Thus, the states of the HHMM can emit sequences rather than single symbols. This application deals only with two-level HHMMs. The higher level defines the choices of pen motions between different single-path trajectories that constitute a multi-path static script. The lower level defines the choices of pen motions between the skeleton samples that constitute a specific single-path trajectory.

The notation of Chapter 4 is generalised as follows: An HMM  $\lambda_h$  is constructed for each disconnected image  $\mathbf{P}_h = \{\mathbf{p}_1^h, \mathbf{p}_2^h, \dots, \mathbf{p}_{M_h}^h\}$  called a *sub-image* of a static script, where  $M_h$  is the number of skeleton samples that constitute  $\mathbf{P}_h$ , and  $N$  is the number of sub-images so that  $h = \{1, \dots, N\}$ . Thus, e.g., “s” has one sub-image so that  $N = 1$ , and “i” has two sub-images so that  $N = 2$ . The topology and PDFs for each  $\lambda_h$  are derived as described in Chapter 4, so that  $\lambda_h$  has  $N_h$  emitting states  $\mathbf{q}_h = \{q_1^h, q_2^h, \dots, q_{N_h}^h\}$  and two non-emitting states  $q_0^h$  and  $q_{N_h+1}^h$ . Each emitting state  $q_i^h$  is associated with a PDF  $f_{i,h}(\mathbf{x})$  consisting of two independent components that are both spherical Gaussians. The transition probability matrix of  $\lambda_h$  is  $\mathbf{A}_h = \{a_{01}^h, a_{02}^h, \dots, a_{N_h N_h+1}^h\}$  and is developed as described in the previous chapter. All the specific parameters for  $\lambda_h$  are summarised in Sections 4.7 and 4.8. As a direct consequence of the techniques developed in Chapter 4, the skeleton sample associated with  $q_i^h$  and the skeleton sample that is shared by the states preceding  $q_i^h$  are assumed to be known in the sections to follow; see Section 4.7 for further detail.

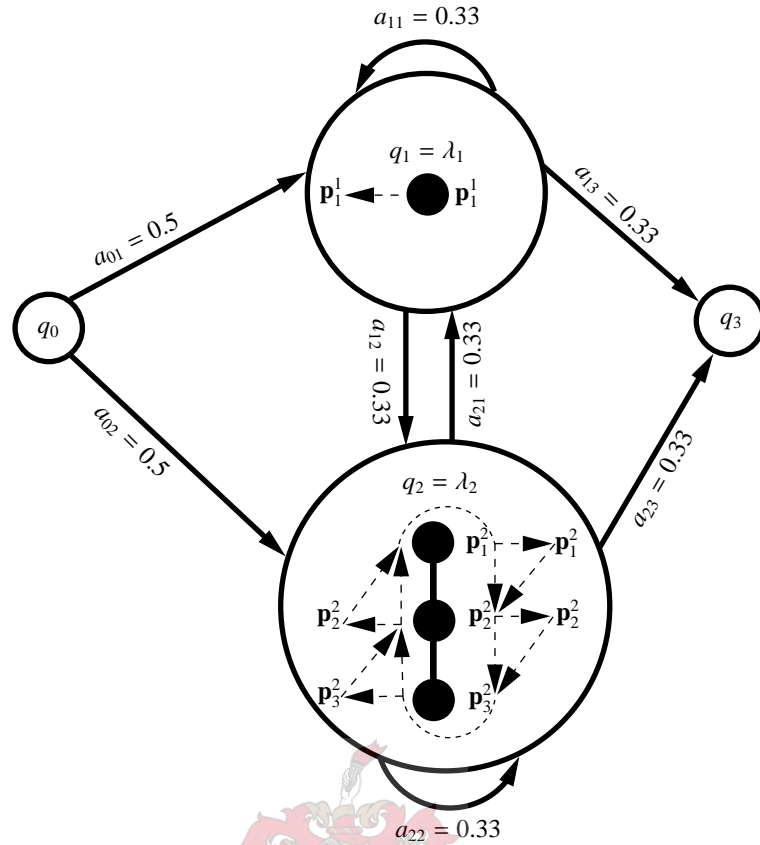
Recall that a dynamic exemplar  $\mathbf{X}$  is represented as  $T$   $d$ -dimensional feature vectors, so that  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T]$ . Section 4.6 has been shown that our current PDFs can be matched to 4D features vectors consisting of two positional and two directional components (i.e.,  $d = 4$ .) In this chapter it is shown how to include a fifth component, namely pressure. The first two components of each dynamic feature vector  $\mathbf{x}_t$  form a sub-vector  $\mathbf{x}_t^{1,2}$  describing the dynamic pen position. It has been shown in Section 3.3 that each dynamic exemplar is normalised so that the dynamic pen pressure is 0 or 1. We now let the fifth component of each dynamic feature vector  $\mathbf{x}_t$  form the scalar  $x_t^5$  describing the dynamic pen pressure. The dynamic exemplar is also normalised so that  $\mathbf{x}_t^{1,2} = (0, 0)$  if  $x_t^5 = 0$ . The third and fourth components are directional components (normalised velocity) with  $\mathbf{x}_t^{3,4} = (\mathbf{x}_t^{1,2} - \mathbf{x}_{t-1}^{1,2}) / \|\mathbf{x}_t^{1,2} - \mathbf{x}_{t-1}^{1,2}\|$ , with  $\mathbf{x}_t^{3,4} = (0, 0)$  if  $x_t^5 = 0$  or if  $t = 1$ . According to the current PDF structure only  $\mathbf{x}_t^{1,2}$  and  $\mathbf{x}_t^{3,4}$  are matched to our HMM. The next section includes a PDF component that can deal with  $x_t^5$ . The current PDF components for  $q_i^h$  are summarised as follows. Let  $f_{i,h}^{\text{P}}(\mathbf{x}_t^{1,2})$  be the positional PDF component  $\mathcal{N}(\mathbf{u}_{i,h}^{\text{P}}, \sigma_{\text{P}}')$  evaluated at  $\mathbf{x}_t^{1,2}$ . Likewise, let  $f_{i,h}^{\text{V}}(\mathbf{x}_t^{3,4})$  be the directional PDF component  $\mathcal{N}(\mathbf{u}_{i,h}^{\text{V}}, \sigma_{\text{V}}')$  evaluated at  $\mathbf{x}_t^{3,4}$ .



Since these PDF components are independent, the joint observation PDF of  $q_i^h$  evaluated at feature vector  $\mathbf{x}_t$  is given by  $f_{i,h}(\mathbf{x}_t) = f_{i,h}^P(\mathbf{x}_t^{1,2})f_{i,h}^V(\mathbf{x}_t^{3,4})$ , where  $i \in \{1, \dots, N_h\}$ ,  $h \in \{1, \dots, N\}$ .

The HMMs  $[\lambda_1, \dots, \lambda_N]$  are combined to form the higher-level emitting states  $\mathbf{q} = [q_1, \dots, q_N]$  of the HHMM  $\lambda$  of a multi-path static script. The higher-level states of  $\lambda$  are connected according to a fully-connected ergodic topology [8] with transition weights defined by  $\mathbf{A}$ . In accordance with a *fully-connected ergodic* topology, all states are connected to each other with non-zero transition probabilities. In this case,  $a_{ij}$  is set equal for each transition leaving  $q_i$ . The lower-level states remain as before. The state sequence that must be extracted from  $\lambda$  is  $\mathbf{s} = [s_1, s_2, \dots, s_T]$ . Recall that a skeleton sample is associated with each lower-level emitting state. Hence,  $\mathbf{s}$  can be directly translated to the pen trajectory of the static script, as shown in Section 5.4.

An example of the HHMM  $\lambda$  for the character “i”, consisting of four skeleton samples and two sub-images, i.e.,  $N = 2$ , is shown in Figure 5.2. Each higher-level emitting state (big circle) corresponds to a sub-image for which an HMM  $\lambda_h$  is derived. The skeleton samples (filled dots) that constitute the different sub-images and from which the lower-level emitting states are derived, are also shown. Pen motions between the randomly-ordered skeleton samples are rendered as dashed lines. (For the sake of simplicity it is not shown that a pen trajectory can start at any skeleton sample within each sub-image, and the links between  $\mathbf{p}_1^2$  and  $\mathbf{p}_3^2$  are not shown.) It is indicated that  $q_1$  is constructed from  $\mathbf{P}_1 = \mathbf{p}_1^1$ , whereas  $q_2$  is constructed from  $\mathbf{P}_2 = \{\mathbf{p}_1^2, \mathbf{p}_2^2, \mathbf{p}_3^2\}$ . The interconnections of the higher-level states  $\{q_0, q_1, q_2, q_3\}$  in  $\lambda$  dictate the choices of pen motions (solid arrows) between the sub-images that can be estimated. All the transitions leaving a state are equally weighted. Note that the topology allows the pen to arbitrarily jump between skeleton samples, thereby removing the context that has so carefully been included using second-order HMMs, as described in Chapter 4. According to Chapter 4, e.g.,  $\mathbf{p}_2^2$  is a segment point, so that the pen is not allowed to turn around or change direction abruptly from  $\mathbf{p}_2^2$ . However, according to the current topology, nothing prevents the extraction of the pen trajectory (amongst many others)  $[\mathbf{p}_3^2, \mathbf{p}_2^2, \mathbf{p}_3^2]$ , as the pen is allowed to exit and re-enter  $q_2$  at any instance. Thus, no pressure information is contained in the current HHMM structure: all the skeleton samples are just indirectly connected to each other. It is more sensible to allow a transition to any skeleton sample only directly after a pen-up event. Hence, the next section shows how to identify pen-up events by exploiting dynamic exemplar pen pressure information.

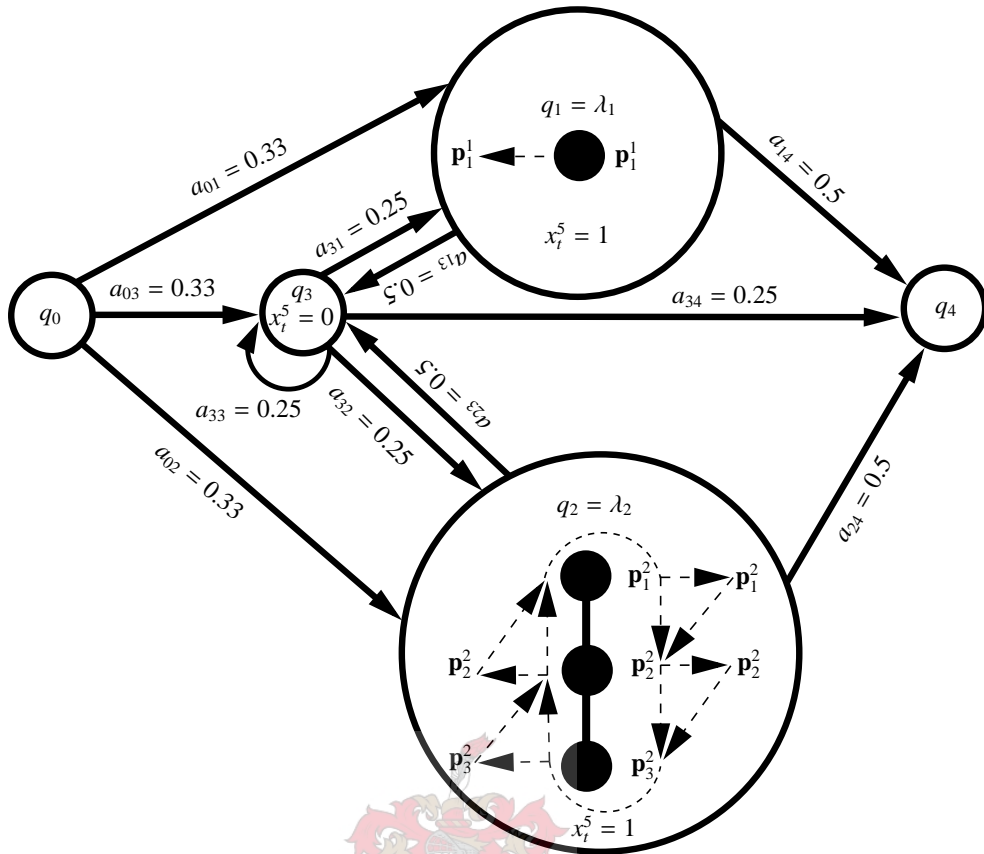


**Figure 5.2:** The HHMM  $\lambda$  for the static character “i”. The higher-level states of  $\lambda$  are represented as unfilled circles, where each emitting state  $q_h$  corresponds to an HMM  $\lambda_h$  for each sub-image. The weighted ( $a_{ij}$ ) choices of pen motions (solid arrows) between the different sub-images are shown. The simplified choices of pen motions (dashed arrows) between the skeleton samples (filled dots) within each sub-image are also shown.

## 5.2 Identifying pen-up and pen-down events

The first step to identify pen-up and pen-down events is to add a *zero-pressure* emitting state  $q_{N+1}$  to the higher level of the HHMM  $\lambda$ , so that  $\lambda$  has  $N + 1$  emitting states, a non-emitting initial state  $q_0$  and a non-emitting terminating state  $q_{N+2}$ . All transition links connecting higher-level emitting states to other higher-level emitting states of  $\lambda$  are removed. Instead, all higher-level emitting states and  $q_0$  are connected to  $q_{N+1}$ . The zero-pressure state  $q_{N+1}$  is connected to itself, to  $q_{N+2}$  and to all the other higher-level emitting states. All transition weights contained in  $A$  are set equal and normalised to sum to 1.0. Note, however, that all the lower-level transition weights contained in  $A_h$  for  $h \in \{1, \dots, N\}$  remain unchanged (as developed in Chapter 4). The new topology for  $\lambda$  is depicted by the solid arrows in Figure 5.3, as derived from Figure 5.2. There are now three higher-level emitting states in the figure, where  $q_3$  is the zero-pressure state.

Similar to the PDFs developed in Sections 4.6 to 4.8, two PDF components are associated



**Figure 5.3:** Manipulating the HHMM  $\lambda$  for the static character “i” to identify pen-up and pen-down events. An HMM  $\lambda_h$  corresponds to an emitting state  $q_h$  (big circle) for each sub-image. The higher-level emitting state  $q_3$  dictates pen motions (solid arrows) between the different sub-images only if  $x_t^5 = 0$ . The simplified choices of pen motions (dashed arrows) between the skeleton samples (filled dots) within each sub-image are also shown.

with  $q_{N+1}$ . The first position component  $f_{N+1}^P(\mathbf{x}_t^{1,2})$  is a spherical Gaussian PDF (described by (4.1)) and is written as  $\mathcal{N}(\boldsymbol{\mu}_{N+1}^P, \sigma_P')$ . The second directional PDF component  $f_{N+1}^V(\mathbf{x}_t^{3,4})$  is also a spherical Gaussian PDF and is written as  $\mathcal{N}(\boldsymbol{\mu}_{N+1}^V, \sigma_V')$ . Recall from the previous section that  $\mathbf{x}_t^{1,2} = (0, 0)$  and  $\mathbf{x}_t^{3,4} = (0, 0)$  if  $x_t^5 = 0$ . Hence, we let  $\boldsymbol{\mu}_{N+1}^P = \boldsymbol{\mu}_{N+1}^V = (0, 0)$ ,  $\sigma_P' = \sigma_V' = 0.4$  in  $\mathcal{N}(\boldsymbol{\mu}_{N+1}^P, \sigma_P')$  and  $\mathcal{N}(\boldsymbol{\mu}_{N+1}^V, \sigma_V')$ .

To force the state sequence to enter  $q_{N+1}$  under zero-pressure conditions, i.e.,  $s_t = q_{N+1}$  if  $x_t^5 = 0$ , a third statistically independent PDF component  $f(x)$  with a uniform distribution is added to the PDF of  $q_{N+1}$  and to all the lower-level emitting states, where

$$f(x) = \begin{cases} 1/(b-a), & \text{for } a \leq x \leq b \\ 0, & \text{elsewhere,} \end{cases} \quad (5.1a)$$

$$(5.1b)$$

for real constants  $-\infty < a < b < \infty$  (see [61].) For the sake of brevity, we refer to  $f_{i,h}^F(x_t^5)$ , the third PDF component (described by (5.1)) of  $q_i^h$ , as  $\mathcal{U}_{i,h}(a, b)$ , and let  $a = 0.5$  and  $b = 1.5$  (the reason for this specific choice will become clear in a moment.) Since all the PDF components

are independent, the joint observation PDF at the lower-level emitting state  $q_i^h$  evaluated at feature vector  $\mathbf{x}_t$  is now given by

$$f_{i,h}(\mathbf{x}_t) = f_{i,h}^P(\mathbf{x}_t^{1,2})f_{i,h}^V(\mathbf{x}_t^{3,4})f_{i,h}^F(x_t^5), \quad (5.2)$$

where  $i \in \{1, \dots, N_h\}$ ,  $h \in \{1, \dots, N\}$ ,  $t = [1, \dots, T]$ . Likewise, we refer to  $f_{N+1}^F(x_t^5)$ , the third PDF component of the higher-level emitting state  $q_{N+1}$ , as  $\mathcal{U}_{N+1}(a, b)$ , and let  $a = -0.5$  and  $b = 0.5$ , so that the joint observation PDF evaluated at feature vector  $\mathbf{x}_t$  is given by

$$f_{N+1}(\mathbf{x}_t) = f_{N+1}^P(\mathbf{x}_t^{1,2})f_{N+1}^V(\mathbf{x}_t^{3,4})f_{N+1}^F(x_t^5), \quad (5.3)$$

where  $t = [1, \dots, T]$ . The choices of  $a$  and  $b$  in  $\mathcal{U}_{i,h}(a, b)$  and  $\mathcal{U}_{N+1}(a, b)$  let  $f_{i,h}(\mathbf{x}_t) = 0$  if  $x_t^5 = 0$  from (5.2), and  $f_{N+1}(\mathbf{x}_t) = 0$  if  $x_t^5 = 1$  from (5.3). The third PDF components therefore serve as *binary gates* that force the state sequence to reveal  $q_{N+1}$  if the dynamic exemplar pen pressure is zero.

The above concepts are illustrated in Figure 5.3. As long as  $x_t^5 = 1$ ,  $s_t = q_i^1$  for  $i \in \{0, \dots, N_1 + 1\}$ , or  $s_t = q_i^2$  for  $i \in \{0, \dots, N_2 + 1\}$ . If a pen-up event is identified at instance  $t + 1$ , i.e.,  $x_{t+1}^5 = 0$ ,  $s_{t+1} = q_3$ . If a pen-down event is subsequently identified at instance  $t + 2$ , i.e.,  $x_{t+2}^5 = 1$ ,  $s_{t+2} = q_i^1$  or  $s_{t+2} = q_i^2$ . Thus, when the dynamic exemplar pen pressure is non-zero, the heuristic framework developed in Chapter 4 is applicable, so that the choices of pen motions within the different sub-images are restricted as illustrated by the dashed arrows. (For the sake of simplicity it is not shown that a pen-trajectory can start at any skeleton sample within each sub-image, and the links between  $\mathbf{p}_1^2$  and  $\mathbf{p}_3^2$  are not shown.)

This section has shown that the addition of a single emitting state and a third component to all PDFs enables us to identify and accommodate the single-path trajectories that constitute a multi-path static script. The next section shows how to deal with special cases where spurious disconnections (broken lines) occur in a static script.

### 5.3 Compensating for broken lines (spurious disconnections)

As mentioned in Section 1.5, *spurious disconnections* may result in the static script when the ink is not evenly distributed over the pen-tip. This practical problem arises especially when an individual signs rapidly, so that unevenly distributed ink results in line fragments with very light grey levels which typically vanish after binarisation.

Our HMM topology enables us to deal with broken lines as follows: If two endpoints are judged possibly to be part of the same broken line, then we add appropriate additional states for each of the two endpoints. These states are manipulated to enable the pen to reach the one endpoint

from the other, where the ease of such a transition is dictated by the “brokenness” of the line. We proceed with a detailed discussion. If the lower-level emitting state  $q_j^s$  is associated with an endpoint, it can be preceded by the lower-level non-emitting initial state, immediate lower-level neighbours or lower-level skip-link neighbours (see Chapter 4). For the sake of simplification, we allow only the immediate neighbours of endpoint states in our approach to accommodate broken lines. Thus, if  $q_i^s$  is the immediate neighbour of endpoint state  $q_j^s$  and another endpoint state  $q_k^h$  is encountered, it is tested whether  $q_j^s$  and  $q_k^h$  are associated with the two disconnected endpoints of a broken line, where  $i, j \in \{1, \dots, N_g\}$ ,  $g, h \in \{1, \dots, N\}$ ,  $k, \ell \in \{1, \dots, N_h\}$ .  $N$ ,  $N_g$  and  $N_h$  are the numbers of states in  $\lambda$ ,  $\lambda_g$  and  $\lambda_h$ , respectively (see Section 5.1 for further notation specifications.) It is assumed that these two disconnected endpoints must at least be within nearby vicinity of each other and that the disconnected lines to which they are connected have similar directions. The HMM topology for broken lines is designed in accordance with this assumptions.

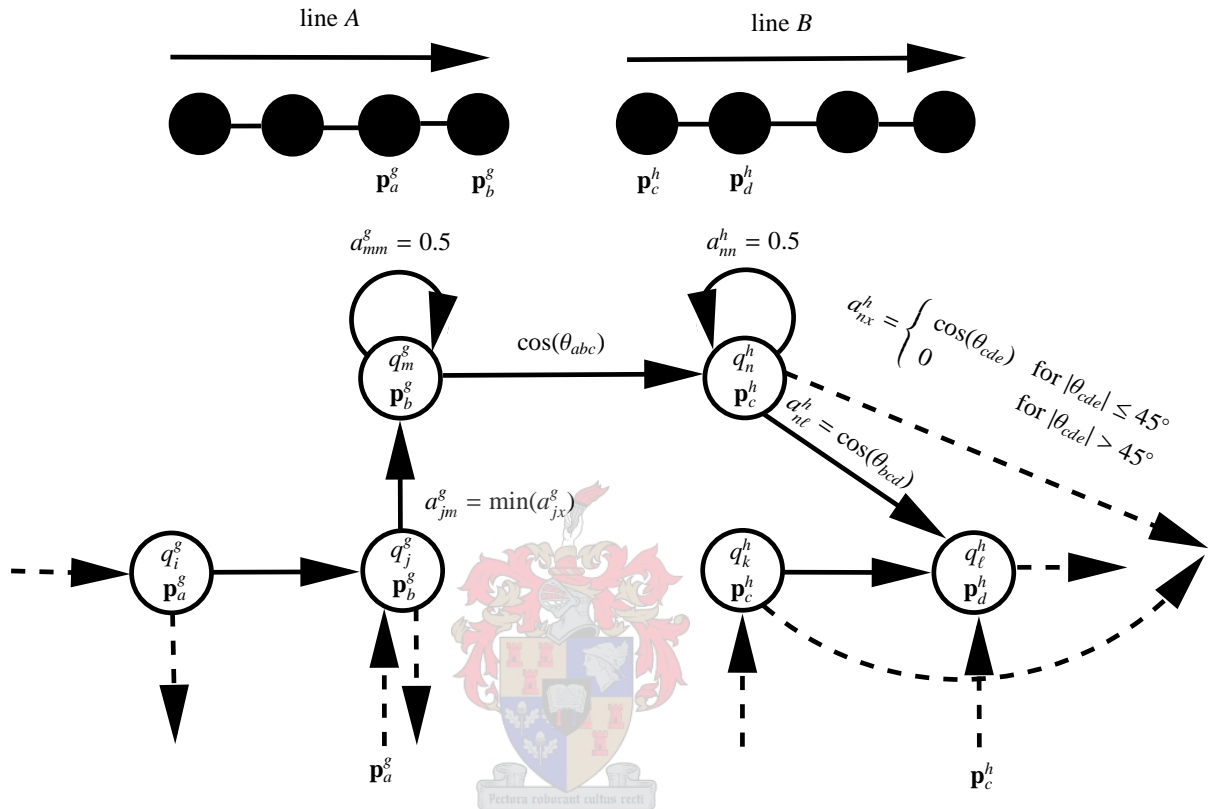
First, let  $\mathbf{p}_a^s$ ,  $\mathbf{p}_b^s$ ,  $\mathbf{p}_c^h$  and  $\mathbf{p}_d^h$  be the skeleton samples associated with states  $q_i^s$ ,  $q_j^s$ ,  $q_k^h$  and  $q_\ell^h$ , respectively. Hence, the angle  $\theta_{abc}$  is computed using (4.3). If  $\theta_{abc} \leq 45^\circ$  and  $\|\mathbf{p}_b^s - \mathbf{p}_c^h\| \leq 70$  (pixels),  $\mathbf{p}_b^s$  and  $\mathbf{p}_c^h$  are considered to be part of a broken line and they must therefore be reconnected. Note that the connection of  $\mathbf{p}_b^s$  and  $\mathbf{p}_c^h$  effectively unites  $\lambda_g$  and  $\lambda_h$  if  $g \neq h$ , so that there is one less higher-level emitting state in  $\lambda$ . Figure 5.4 shows an example of two endpoints  $\mathbf{p}_b^s$  (indicated by a solid circle) and  $\mathbf{p}_c^h$  (indicated by an x-marker) in the skeleton of a static script, which are identified as parts of a broken line. The dashed circle spans the region  $\|\mathbf{p}_b^s - \mathbf{p}_c^h\| \leq 70$  (pixels.)



**Figure 5.4:** The locations of two disconnected endpoints  $\mathbf{p}_b^s$  (solid circle) and  $\mathbf{p}_c^h$  (x-marker) in a broken line. Candidate disconnected endpoints are within the vicinity of the dashed circle from  $\mathbf{p}_b^s$ .

To connect  $\mathbf{p}_b^s$  and  $\mathbf{p}_c^h$ , two extra states  $q_m^s$  and  $q_n^h$  are added to  $\lambda_g$  and  $\lambda_h$ , respectively. The first extra state  $q_m^s$  is associated with the same skeleton sample  $\mathbf{p}_b^s$  as  $q_j^s$ . The state  $q_m^s$  is connected to  $q_n^h$  with a transition weight of  $\cos(\theta_{abc})$ , to itself with  $a_{mm}^s = 0.5$ , and to  $q_j^s$  with  $a_{jm}^s = \min(a_{jx}^s)$ , where  $x$  can be any transition link leaving  $q_j^s$  (as developed in Chapter 4.) All transition weights from  $q_j^s$  and  $q_m^s$  are again normalised to sum to 1.0. This topology is illustrated in Figure 5.5, where the two disconnected endpoints from the set of skeleton samples (filled dots)

that constitute lines *A* and *B* must be connected. All the states are rendered as circles, where the state labels (top) and their associated skeleton samples (bottom) are indicated. Transitions to states outside the figure are rendered as dashed lines. Note that more than one state is typically associated with a skeleton sample, e.g.,  $q_j^g$  is preceded by all the states that share skeleton sample  $\mathbf{p}_a^g$ .



**Figure 5.5:** *Compensating for broken lines. An unexpected disconnection occurs between the skeleton samples (filled dots) that constitute lines *A* and *B*, which has to be corrected. The appropriate HMM states (top labels in circles) and their associated skeleton samples (bottom labels in circles) are shown. Transition weights ( $a_{ij}$ ) are shown and transitions to/from states outside the figure are indicated by dashed arrows.*

State  $q_n^h$  is associated with the same skeleton sample  $\mathbf{p}_c^h$  as  $q_k^h$  and is connected to itself with  $a_{nn}^h = 0.5$ , to the non-emitting terminating state with  $a_{n(N_h+1)} = 0.5$  and to all the other destinations of  $q_k^h$  that are associated with skeleton samples. To compute the latter transition weights, the angle  $\theta_{cde}$  for each destination state  $q_x^h$  from  $q_n^h$  is computed, where  $\mathbf{p}_e^h$  is the skeleton sample associated with  $q_x^h$ . Subsequently,  $a_{nx}^h = \cos(\theta_{cde})$  for  $|\theta_{cde}| \leq 45^\circ$  and  $a_{nx}^h = 0$  for  $|\theta_{cde}| > 45^\circ$ , as illustrated in Figure 5.5.

Typically, when a broken line occurs, the part of the line that is absent in the static script corresponds to a line in the dynamic exemplar which is written with non-zero pen pressure. Thus, in such cases the state sequence must be allowed to linger in  $q_m^g$  or  $q_n^h$ . However, some penalty has to be introduced to prevent the exploitation of these extra states in cases where

broken lines do not occur. Accordingly, the PDFs for  $q_m^g$  and  $q_n^h$  are designed as follows. Firstly, the PDFs are designed so that only nearby dynamic exemplar samples with non-zero pressure are matched to  $\mathbf{p}_b^g$  and  $\mathbf{p}_c^h$ . Hence, to ensure the matching of nearby samples, the positional PDF component of  $q_m^g$  is  $\mathcal{N}(\mathbf{u}_{m,g}^P, \sigma'_p) = \mathcal{N}(\mathbf{u}_{j,g}^P, \sigma'_p)$ , i.e., the positional constraint invoked by  $\sigma'_p$  is the same as for all the other PDFs in  $\lambda_g$  (see Section 4.8.) The pressure PDF component  $\mathcal{U}_{m,g}(a, b) = \mathcal{U}_{j,g}(a, b)$ , i.e.,  $a = 0.5$  and  $b = 1.5$  to ensure the matching of dynamic exemplar samples written with non-zero pressure. Likewise,  $\mathcal{N}(\mathbf{u}_{n,h}^P, \sigma'_p) = \mathcal{N}(\mathbf{u}_{k,h}^P, \sigma'_p)$  and  $\mathcal{U}_{n,h}(a, b) = \mathcal{U}_{k,h}(a, b)$ . Note that  $\mathbf{x}_i^{3,4}$  is normalised so that  $\min(\mathbf{x}_i^{3,4}) = (0, 0)$  and  $\max(\mathbf{x}_i^{3,4}) = (1, 1)$ . Hence,  $\mathbf{u}_{m,g}^V = \mathbf{u}_{n,h}^V = (0, 0)$  in  $\mathcal{N}(\mathbf{u}_{m,g}^V, \sigma'_v)$  and  $\mathcal{N}(\mathbf{u}_{n,h}^V, \sigma'_v)$ . The directional deviation  $\sigma'_v$  is chosen so large that  $f_{n,h}^V(\mathbf{x}_i^{3,4})$  and  $f_{m,g}^V(\mathbf{x}_i^{3,4})$  are always approximately equal to  $5 \times 10^{-10}$ , making it rather expensive to linger in states  $q_m^g$  and  $q_n^h$  in comparison with other emitting states, if it is not absolutely necessary.

It should be noted that the same process to identify broken lines is followed for *all* the applicable endpoints so that there is a bi-directional connection between broken lines.

## 5.4 The hidden state sequence (estimated pen trajectory)

We have now developed the full HHMM for the static image of a handwritten script. This HHMM consists of states associated with the position coordinates of the static image skeleton, and transition probabilities that dictate consecutive transitions between states. Each state is associated with a single PDF that consists of three statistically independent components, describing the positional, directional and pen pressure variations. The transition probabilities govern the possible choices of pen motions, based on three basic assumptions:

1. The pen is not allowed to turn around suddenly within line segments of a sub-image whenever the pen pressure is non-zero. This assumption assumes that the pen must maintain its direction of traversal, i.e., it is based on the continuity criterion of motor-controlled pen motions, discussed in Section 2.2.
2. The pen is allowed to turn around at endpoints and crosspoints within a sub-image whenever the pen pressure is non-zero. It should be noted that this assumption does not invalidate the continuity criterion of motor-controlled pen motions. It simply identifies cases where the continuity criterion is not necessarily applicable to estimate the pen trajectories of the static scripts by assuming that ambiguities are most likely to occur at endpoints and crosspoints. Thus, due to the representation of the handwritten scripts as pre-processed 2D images, it is not necessarily possible to relate continuous pen motions to the available representative curves.

3. The pen can reach any skeleton sample at a pen-down event, i.e., after the individual has lifted the pen (after a pen-up event.) Similar to the previous assumption, this assumption does not invalidate the continuity criterion of motor-controlled pen motions. It simply accounts for the information loss in the 2D images—although an individual’s hand is still constrained to move continuously in the air while the pen is lifted, it can resume writing *anywhere* on the document at a pen-down event. Thus, this assumption also identifies cases where the continuity criterion is not necessarily applicable to estimate the pen trajectories of the static scripts.

It is clear from the above assumptions that provision is made in our HHMMs for situations where the continuity criterion of motor-controlled pen motions can not necessarily be used to extract dynamic information from 2D images. In such cases, we rely on the additional information provided by our pre-recorded dynamic exemplars. Recall that rule-based methods rely exclusively on a prior generalised model (which is prone to fail in certain situations.) In our application, one can think of the dynamic exemplars as models describing the underlying principles of handwritten motions. These models are, however, writer specific. In ambiguous parts, where one can not rely on a single prior model, no heuristics are employed. The writer-specific models, established by the dynamic exemplars, are simply followed. Thus, the additional information available from the dynamic exemplars (prior models) are applied, where this additional information is specific to each individual. The models employed by rule-based methods are therefore *fixed*, whereas our models (dynamic exemplars) are *flexible*, as our models are able to change in accordance with the changing handwriting scripts.

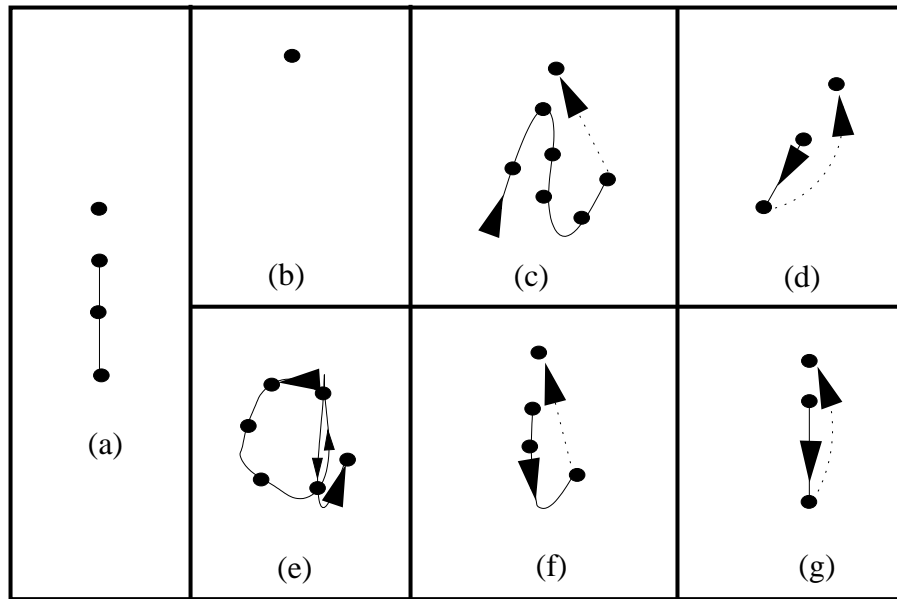
Most importantly, when a dynamic exemplar is matched to the HHMM, one determines the most likely state sequence. Since the higher-level states are associated with the sub-images and the lower-level states with the position coordinates of the skeleton, this sequence yields the maximum likelihood pen trajectory as determined by the model. The dynamic exemplar  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T]$  is matched to the HHMM  $\lambda$  of the static image using the Viterbi algorithm [16, 68, 25]. This results in an optimum state sequence  $\mathbf{s} = [s_1, \dots, s_T]$  as well as a likelihood.

The PDF associated with the zero-pressure state  $q_{N+1}$  in the higher level of  $\lambda$  emits a single observation, whereas the rest of the PDFs emit sequences of observations due to our hierarchical structure. However, every HHMM can also be represented as a single-level HMM (see [25].) For the sake of simplicity we represent  $\lambda$  as its single-level HMM equivalent  $\lambda'$  with  $N$  states. The globally optimised likelihood of  $\mathbf{s}$ , based on  $\lambda'$  and the dynamic data  $\mathbf{X}$ , is then given by

$$\delta = a_{s_0 s_1} \prod_{t=1}^T a_{s_t s_{t+1}} f_{s_t}(\mathbf{x}_t), \quad (5.4)$$

where  $s_0 = 0$  is the non-emitting initial state of  $\lambda'$ ,  $s_{T+1} = N + 1$  is the non-emitting terminating state of  $\lambda'$  and  $f_{s_t}(\mathbf{x}_t)$  is the PDF associated with emitting state  $s_t$  of  $\lambda'$  evaluated at feature vector





**Figure 5.6:** Estimating the pen trajectory of a static “i”. (a) The static skeleton of the character “i”. (b)-(g) Matching different dynamic exemplars to the HMM  $\lambda'$  of (a), where arrows indicate the sequences of the dynamic exemplar pen positions.

$\mathbf{x}_i$ . We can therefore obtain a maximum likelihood state sequence for each available dynamic exemplar of a static image. This provides a point-wise correspondence between the static image and each dynamic exemplar.

The likelihood  $\delta$  is a useful measure of similarity between a static image and a dynamic exemplar. It tends to decrease if a segment exists in the dynamic exemplar and not in the static image, or in cases of inconsistencies in size or orientation. However, it can happen that a dynamic exemplar matches only a portion of the static image very well. A dynamic character “1”, e.g., can produce a high likelihood on a static “7”. To illustrate the behaviour of  $\delta$ , the different dynamic exemplars shown in Figures 5.6(b)-(g) are matched to the HMM  $\lambda'$  of Figure 5.6(a), where  $\lambda'$  is the single-level equivalent of Figure 5.3. Hence,  $\delta$  tends to decrease in cases of inconsistent pen movements, e.g., Figure 5.6(c); extreme shape differences, e.g., Figure 5.6(e); different orientations, e.g., Figure 5.6(d); and trajectories occurring in the dynamic exemplar and not in the static script, e.g., Figures 5.6(c), (e) and (f).

As mentioned above,  $\delta$  will not necessarily decrease if the dynamic exemplar does not contain all the curves that constitute the static script, e.g., Figure 5.6(b) could have a similar  $\delta$  to Figure 5.6(g). To prevent this, we weight the likelihood from (5.4) in the following manner: Firstly, the total path length  $T_L$  is computed as the sum of distances between all the connected skeleton samples of the static image. Secondly, the path length  $R_L$  of the recovered pen trajectory is computed, so that  $R_L \leq T_L$ . Note that web-like structures may contain excessive lines to model complicated intersections. We therefore do not include the path lengths of consecutive

samples that constitute web-like structures. We now weight each of the maximum likelihood state sequences (one for each dynamic exemplar) as follows:

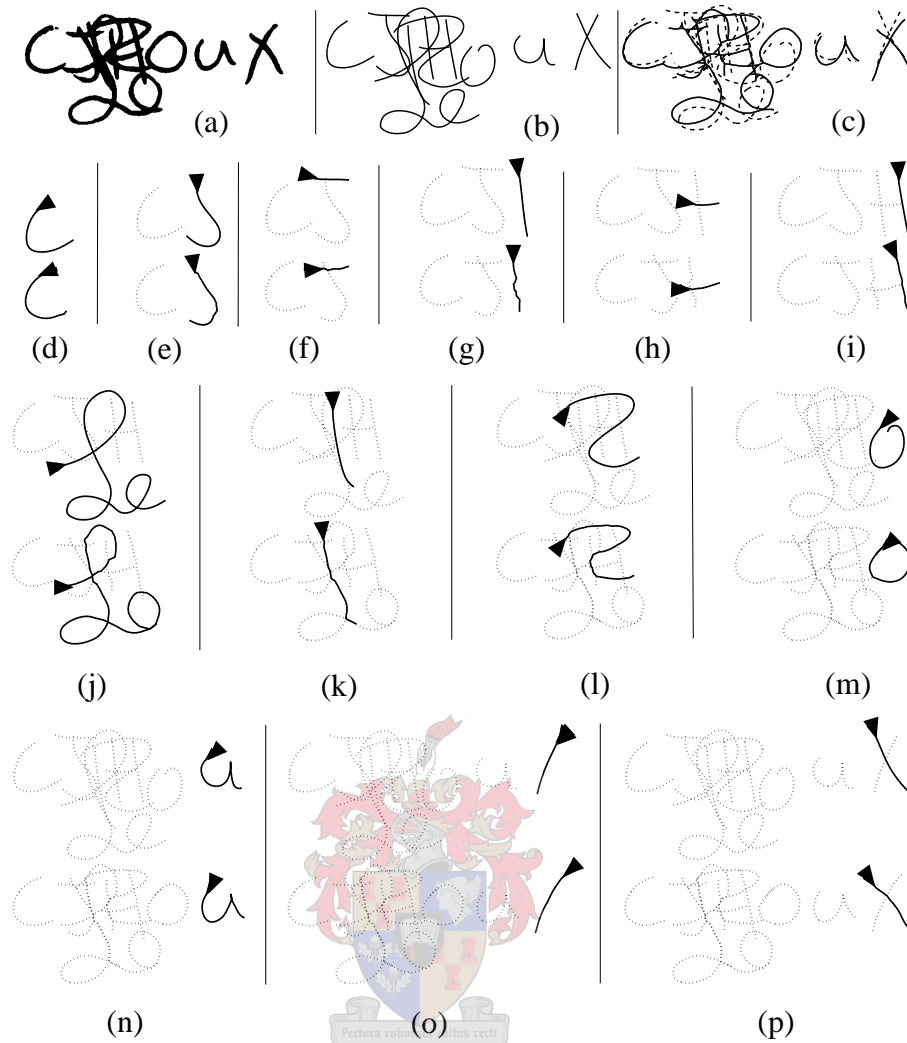
$$\log(\delta_w) = \text{sign}(\log(\delta)) \frac{R_L}{T_L} |\log(\delta)| \quad (5.5)$$

Finally, the dynamic exemplar's state sequence that produces the maximum weighted likelihood  $\delta_w$  is chosen as the estimated pen trajectory. If  $\delta_w$  is computed as described above, Figure 5.6(g) will have the highest and Figure 5.6(e) the lowest  $\delta_w$ . The state sequence resulting from Figure 5.6(g) will therefore yield the estimated pen trajectory of Figure 5.6(a).

Note that  $\delta_w$  can be useful to identify forgeries in a signature verification application. This is also indicated by some preliminary experiments that are presented in Section 7.2.4.

A typical static signature with three sub-images from our database is shown in Figure 5.7(a) (close inspection of the signature shows that there are only three disconnected images.) Of the fourteen pre-recorded dynamic exemplars, the one that has the highest  $\delta_w$  is shown in Figure 5.7(b). Figure 5.7(c) depicts the aligned dynamic exemplar (dashed line) from Figure 5.7(b) and the skeleton (solid line) of Figure 5.7(a) after preprocessing. The solid lines in Figures 5.7(d)-(p) illustrate how the single-path trajectories of the dynamic exemplar (top) and the static curves of the skeleton (bottom) from Figure 5.7(c) are matched. Dashed lines render previous single-path trajectories, and the direction of corresponding starting positions is indicated by arrows. The sequenced single-path trajectories of Figure 5.7(a) are therefore revealed by establishing a point-wise correspondence with the dynamic exemplar. Note that the dynamic exemplar is especially helpful to estimate the single-path trajectories in complicated regions, e.g., the bottom trajectories in Figures 5.7(e)-(l), which overlap greatly in the leftmost sub-image of the static script. It should also be noted that our system computes all thirteen single-path trajectories of Figure 5.7(a) although the static signature consists of only three sub-images.

One final alteration is made to the estimated pen trajectories before their accuracies are calculated. Chapter 4 has shown that some skeleton samples may be skipped or consecutively repeated. This is due to the inclusion of skip-link and duration states in our model, to compensate for static scripts and dynamic exemplars with different numbers of samples. To remove this compensation (which is necessary for our evaluation protocol, as described in Section 6.1), we reinstate the skipped samples and remove samples that are consecutively repeated.



**Figure 5.7:** Estimating the pen trajectory of a multi-path signature. (a) A multi-path static signature and (b) the dynamic exemplar that corresponds best to it. (c) The dynamic exemplar (dashed lines) from (b) superimposed on the static skeleton (solid lines) of (a) after preprocessing. (d)-(p) Estimating the sequence of single-path trajectories (bottom) that constitute the skeleton from (c) by establishing a pointwise correspondence with the dynamic exemplar trajectories (top) from (c). The directions of corresponding starting positions are indicated by arrows, and previous single-path trajectories are rendered as dashed lines.

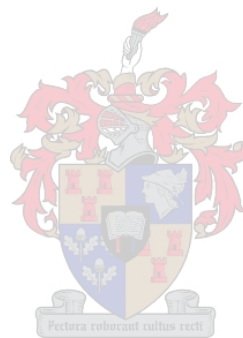
## 5.5 Summary

This chapter has extended the HMMs for single-path static scripts to handle multi-path static scripts. The most important accomplishments of this chapter are summarised as follows:

- It has been shown how to define pen-up and pen-down events, i.e., the single-path trajectories that constitute a static script, without removing the context within the second-order

HMMs that are derived from the different sub-images of the script. Specifically, bi-level hierarchical HHMMs with additional zero-pressure states force discontinuities in pen trajectories where pen-up events occur.

- Broken lines that may occur in static scripts have been treated as a special case of multi-path scripts. Compensation for such spurious disconnections has been included in our HMM topology.
- It has been shown how to estimate a pen trajectory from the HMM of a static script using the Viterbi algorithm. Each pen trajectory has a likelihood after matching a dynamic exemplar to the HMM. This likelihood can be used as a confidence measure for the accuracy of an estimated pen trajectory.



# Chapter 6

## Experiments

In order to evaluate the accuracy of an estimated pen trajectory, a ground-truth trajectory is required. Since we were unable to find any standardised database containing both the on-line and off-line versions of signatures, we developed our own, named US-SIGBASE. Section 6.2 describes US-SIGBASE in more detail. Section 6.1 presents evaluation protocols: Section 6.1.1 treats existing protocols, whereas Section 6.1.2 describes our evaluation protocol. Experiments are described in Section 6.3, where the configurations of the experiments and experimental results are presented. Typical errors made by our system are described in Section 6.3.6. These errors can be scrutinised by viewing the animation examples on the attached CD; see Appendix A for further details. Our results are compared with results from existing techniques in Section 6.4, and some conclusions are drawn in Section 6.5.

### 6.1 Evaluation protocol

US-SIGBASE consists of 814 multi-path signatures for 51 individuals. Each signature was recorded on a piece of paper placed on a digitising tablet. Thus, the dynamic counterpart of each signature was obtained. However, due to noise introduced during every stage of the recording process, i.e., while recording a dynamic signature and while scanning, binarising and skeletonising its static counterpart, the image skeleton generally differs from its dynamic counterpart. Thus, one cannot assume a one-to-one correspondence between a skeletonised static image and its dynamic counterpart. The ground-truth pen trajectory is obtained by matching the dynamic counterpart of a static script to a slightly modified HHMM of the script (as described in Chapter 5). In general, the position coordinates of a static skeleton and a dynamic counterpart are much better aligned than the position coordinates of the static skeleton and any of its dynamic exemplars. Accordingly, the mentioned HHMM modification tightens the standard deviation

$\sigma'_p$  to 7 (measured in pixels) for all the lower-level emitting states in the HHMM. All that remains is to compare the two trajectories—the ground-truth trajectory, as computed above, and the estimated trajectory obtained from the dynamic exemplar. Both trajectories are extracted from the same static skeleton so that it is possible to compare them. However, since the two trajectories are obtained from different dynamic sequences, the trajectories do not necessarily have the same number of samples. A point-wise comparison is therefore not possible. Section 6.1.1 describes existing methods that establish quantitative comparisons between estimated and ground-truth trajectories, whereas Section 6.1.2 presents our evaluation protocol.

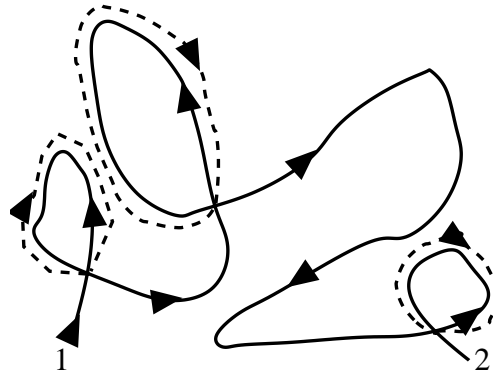
### 6.1.1 Existing evaluation protocols

To our knowledge, there are only two noteworthy quantitative evaluation protocols described in existing literature. The first protocol, by Lau et al. [52], is based on ranking analysis, whereas the second method, by Jäger [38], is based on Dynamic Programming (DP).

The evaluation protocol by Lau et al. [52] is based on paired ranking comparison. In brief, this amounts to presenting the ground-truth trajectory as a sequence of numbers (ranked items)  $\mathfrak{w}_{\text{ground}} = [1, \dots, T]$ . It is then assumed that the estimated pen trajectory consists of the same samples as the ground-truth trajectory, but that the sample sequence might be different. The estimated pen trajectory  $\mathfrak{w}_{\text{est}}$  is therefore also presented as a sequence of  $T$  numbers, where each number occurs exactly once, i.e.,  $w_{\text{est}}(t) \in \{1, \dots, T\}$ , where  $t = [1, \dots, T]$  so that  $w_{\text{est}}(t) \notin \{w_{\text{est}}(1), w_{\text{est}}(2), \dots, w_{\text{est}}(t-1), w_{\text{est}}(t+1), \dots, w_{\text{est}}(T)\}$ , where  $w_{\text{est}}(t)$  is sample  $t$  of  $\mathfrak{w}_{\text{est}}$ . The basis of the method by Lau et al. [52] is Kendall's metric, where Kendall's metric is the minimum pair-wise adjacent transpositions to transform  $\mathfrak{w}_{\text{est}}$  into  $\mathfrak{w}_{\text{ground}}$ . Thus, if  $\mathfrak{w}_{\text{ground}} = [1, 2, 3, 4]$  and  $\mathfrak{w}_{\text{est}} = [1, 3, 4, 2]$ , a Kendall distance of 2 is required to transform  $\mathfrak{w}_{\text{est}}$  into  $\mathfrak{w}_{\text{ground}}$ . The metric established by Lau et al. [52] is a refinement of Kendall's metric, also taking into account general directions and discontinuities between consecutive items in  $\mathfrak{w}_{\text{est}}$ . However, the evaluation protocol and results of Lau et al. [52] are not applicable to our estimation algorithm for the following reasons:

1. No provision is made for estimated and ground-truth trajectories with different path lengths.
2. The protocol requires that segmented static scripts are only represented by their line endpoints. Thus, a static script is presented as a sequence of numbers, where each number represents an endpoint, as shown in Figure 6.1, where the ground-truth trajectory is rendered as a solid line. Thus,  $\mathfrak{w}_{\text{ground}} = [1, 2]$  and the only sequences that can be extracted from the shown script are  $\mathfrak{w}_{\text{est}} = [1, 2]$  or  $\mathfrak{w}_{\text{est}} = [2, 1]$ . If, e.g.,  $\mathfrak{w}_{\text{est}} = [1, 2]$ , their evaluation protocol would indicate that  $\mathfrak{w}_{\text{est}}$  is 100% accurate. A problem arises when a system (such as ours) computes the sequence of *all* the skeleton samples that constitute a static

script: if the estimated sequence of skeleton samples is incorrect between the endpoints, e.g., if the directions of traversal at the loops are reversed (dashed lines) it would have no influence on the result of their evaluation protocol, as only the sequences of endpoints are considered.

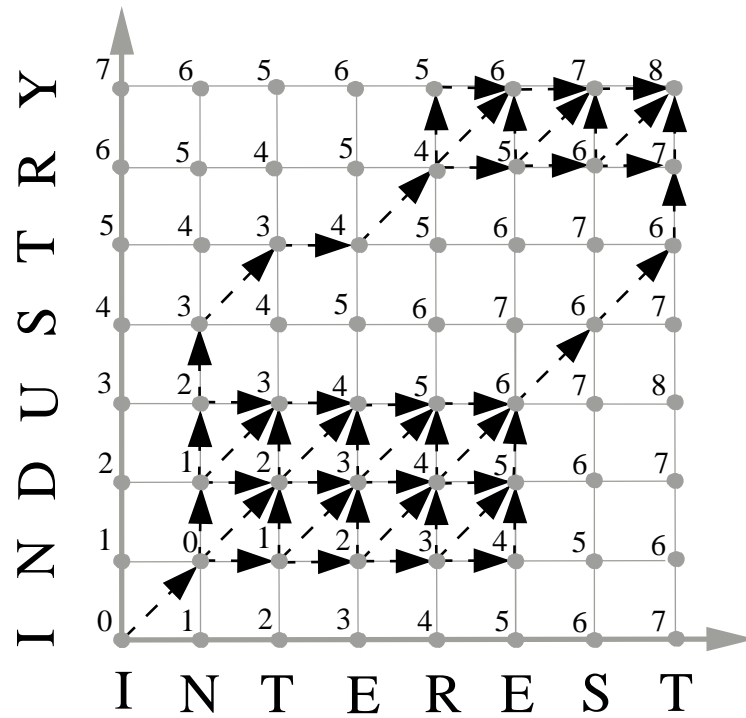


**Figure 6.1:** Segmentation of a static skeleton at endpoints. Solid lines represent the ground-truth trajectory and the dashed lines indicate that the directions of the loops can be erroneously reversed in the estimated pen trajectory.

A more applicable evaluation protocol is followed by Jäger [38], where the smallest number of elementary operations, called the *Levenshtein distance*, required to transform  $\mathfrak{O}_{\text{est}}$  into  $\mathfrak{O}_{\text{ground}}$  is calculated. In this case,  $\mathfrak{O}_{\text{est}}$  and  $\mathfrak{O}_{\text{ground}}$  are sequences of 2D position coordinates and are both extracted from the same skeleton of a static script. To compute the Levenshtein distance, DP is employed. A short presentation of the evaluation protocol established by Jäger [38] follows, using a generic example from [38], where the ground-truth and estimated sequences are both strings, with  $\mathfrak{O}_{\text{ground}} = \text{INTEREST}$  and  $\mathfrak{O}_{\text{est}} = \text{INDUSTRY}$ .

Firstly, a grid is constructed, where each node represents a possible corresponding point between  $\mathfrak{O}_{\text{ground}}$  and  $\mathfrak{O}_{\text{est}}$ , as indicated by the grey dots in Figure 6.2. In general, if  $\mathfrak{O}_{\text{ground}}$  consists of  $n$  samples and  $\mathfrak{O}_{\text{est}}$  consists of  $m$  samples, the grid has  $m \times n$  nodes. The matrix  $D$  is constructed to contain the values for a locally defined cost function, so that  $D(j, i)$  reflects the similarity between  $\mathfrak{O}_{\text{est}}(j)$  (sample  $j$  of  $\mathfrak{O}_{\text{est}}$ ) and  $\mathfrak{O}_{\text{ground}}(i)$  (sample  $i$  of  $\mathfrak{O}_{\text{ground}}$ ), where  $i \in \{1, \dots, n\}$ , and  $j \in \{1, \dots, m\}$ . Hence, for  $\mathfrak{O}_{\text{ground}} = \text{INTEREST}$  and  $\mathfrak{O}_{\text{est}} = \text{INDUSTRY}$ ,  $i = j = 8$ , so that there are 64 nodes in total.

By starting with the initial distance  $D(1, 1)$ , a final accumulated cost at each node  $(j, i)$  is computed in a left-to-right, bottom-to-top fashion. Each node can be preceded by at most three nodes, namely the left, bottom and bottom-diagonal-left nodes. The matrix  $C$  is constructed to contain the final costs at all the nodes. Within the grid's constraints, the cost  $C(j, i)$  is assigned



**Figure 6.2:** Using DP to calculate the optimal Levenshtein distance between two sequences. A grid is constructed with weighted nodes for each character in the strings *INTEREST* and *INDUSTRY*, as indicated by the grey dots. The final cost at each node is shown and the possible paths which all result in a minimum Levenshtein distance of 8 at node (8,8) are rendered as dashed arrows.

to the node  $(j, i)$  in a recursive way, as follows:

$$C(j, i) = \min \left\{ \begin{array}{l} C(j-1, i) + D(j, i), \\ C(j-1, i-1) + D(j, i), \\ C(j, i-1) + D(j, i), \end{array} \right\}, \quad (6.1)$$

where  $i \in \{1, \dots, n\}$  and  $j \in \{1, \dots, m\}$ . The DP evaluation technique enables one to distinguish between different error types that contribute to the total error rate. A sample that occurs in  $\mathfrak{G}_{\text{est}}$  and not in  $\mathfrak{G}_{\text{ground}}$  is called an *insertion*, whereas a sample that occurs in  $\mathfrak{G}_{\text{ground}}$  and not in  $\mathfrak{G}_{\text{est}}$  is called a *deletion*. If a sample from  $\mathfrak{G}_{\text{ground}}$  is mapped to a mismatched sample from  $\mathfrak{G}_{\text{est}}$ , the erroneous sample in  $\mathfrak{G}_{\text{est}}$  is called a *substitution*. When DP is employed, deletions are identified by vertical transitions and insertions are identified by horizontal transitions. Substitution are identified by diagonal transitions mapping mismatched samples onto each other.

The node that precedes  $(j, i)$  (with minimum local cost) is logged, so that the optimal sequence of nodes (leading to a global optimum) can be back-traced when the final node  $(m, n)$  is reached.



Jäger [38] defines  $D(j, i)$  at node  $(j, i)$  from (6.1) as follows:

$$D(j, i) = \begin{cases} 1, & \text{for predecessor nodes } (j-1, i) \text{ and } (j, i-1) & (6.2a) \\ 2, & \text{for predecessor node } (j-1, i-1) \text{ if } \|\mathfrak{p}_{\text{ground}}(i) - \mathfrak{p}_{\text{est}}(j)\| > 0 & (6.2b) \\ 0, & \text{otherwise.} & (6.2c) \end{cases}$$

The final cost  $C(j, i)$  at each node  $(i, j)$  in Figure 6.2 is shown. Note, however, that there are numerous paths that all lead to the minimum Levenshtein distance of 8, as indicated by the dashed arrows. This is problematic if one wishes to pinpoint error sources and erroneous regions. For our application, substituting  $D(j, i) = \|\mathfrak{p}_{\text{ground}}(i) - \mathfrak{p}_{\text{est}}(j)\|$  into (6.1), i.e., minimising the Euclidean distance between the estimated and ground-truth trajectory, would already result in fewer optimal paths. However, big distances between successive samples in  $\mathfrak{p}_{\text{est}}$  and  $\mathfrak{p}_{\text{ground}}$ , where pen-up and pen-down events occur, might have a negative impact. Another problem is that the evaluation protocol followed by Jäger [38] is not parameterisation invariant.

Based on our studies, a satisfactory evaluation protocol should have the following characteristics:

1. The evaluation protocol must establish a mapping between  $\mathfrak{p}_{\text{ground}}$  and  $\mathfrak{p}_{\text{est}}$  that is as unique as possible, so that errors can be identified easily and with confidence. If the pointwise mapping between  $\mathfrak{p}_{\text{ground}}$  and  $\mathfrak{p}_{\text{est}}$  is unique, one can use the point-wise correspondence between the two sequences to identify the errors instead of the final cost of the mapping. This enables a direct identification of erroneous curves in the estimated pen trajectories.
2. The evaluation protocol must be invariant to parameterisation, while also being comprehensive of cases where successive samples are far apart between pen-up and pen-down events, or in cases where successive samples are part of spurious broken lines.
3. The evaluation protocol must be easy to implement, so that it can be used as a standardised technique to evaluate the efficacy of an estimated pen trajectory.

Section 6.1.2 describes our evaluation protocol, which meets the above requirements.

### 6.1.2 Using a left-to-right HMM to establish an evaluation protocol

Our evaluation protocol, introduced in this section, relies on an HMM that is constructed from  $\mathfrak{p}_{\text{ground}}$ . This HMM is matched to  $\mathfrak{p}_{\text{est}}$ , resulting in an optimal state sequence. Similar to the concepts developed in the previous chapters, the state sequence provides a pointwise correspondence between  $\mathfrak{p}_{\text{est}}$  and  $\mathfrak{p}_{\text{ground}}$ . In this case, two states are associated with each sample in

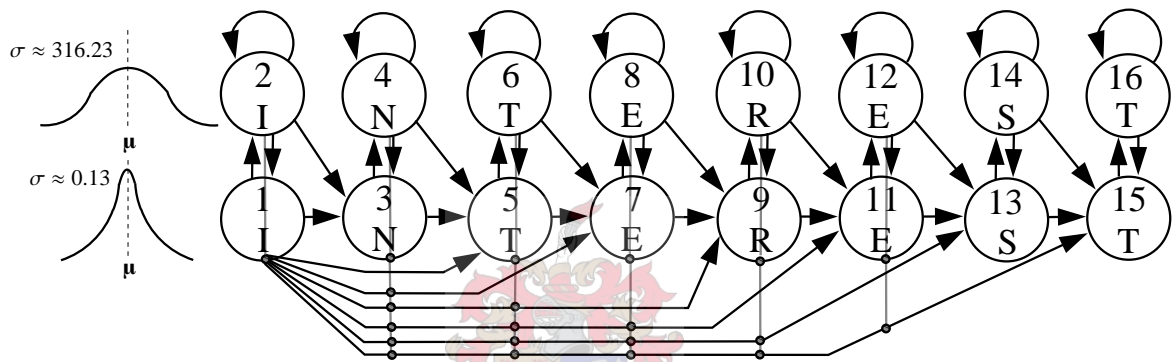
$\mathfrak{O}_{\text{ground}}$ , an “error” state which must be entered when an erroneous sample occurs in  $\mathfrak{O}_{\text{est}}$ , and a “correct” state which must be entered when a correct sample occurs in  $\mathfrak{O}_{\text{est}}$ . Compared with our HMMs for static scripts, our HMM for  $\mathfrak{O}_{\text{est}}$  is much less complex, as  $\mathfrak{O}_{\text{est}}$  and  $\mathfrak{O}_{\text{ground}}$  are extracted from the same image skeleton (i.e., corresponding samples are exactly the same) and both sequences are time signals. The two states associated with each skeleton sample in  $\mathfrak{O}_{\text{ground}}$  are constructed so that their PDFs can be exploited to determine whether errors occur: In cases where corresponding samples in  $\mathfrak{O}_{\text{est}}$  and  $\mathfrak{O}_{\text{ground}}$  are the same, the “correct” states have significantly higher observation likelihoods than the corresponding “error” states, so that the state sequences are forced to reveal the “correct” states. Likewise, in cases where  $\mathfrak{O}_{\text{est}}$  and  $\mathfrak{O}_{\text{ground}}$  differ, the “error” states have significantly higher observation likelihoods than the “correct” states, so that the state sequences are forced to reveal the “error” states.

Our evaluation protocol uses a first-order HMM  $\lambda_{\text{L2R}}$  with a left-to-right topology. For this application,  $\mathfrak{O}_{\text{ground}}$  and  $\mathfrak{O}_{\text{est}}$  are sequences of 2D position coordinates, where  $\mathfrak{O}_{\text{ground}}(k)$  and  $\mathfrak{O}_{\text{est}}(j)$  are the 2D position coordinates at time instances  $k$  and  $j$ , for  $k \in \{1, \dots, n\}$  and  $j \in \{1, \dots, m\}$ . As mentioned above, two states are associated with each sample in  $\mathfrak{O}_{\text{ground}}$ . Each emitting state is associated with a spherical Gaussian PDF, as described by (4.1). We now let  $i \in \{0, \dots, n-1\}$  so that the first state associated with the sample  $\mathfrak{O}_{\text{ground}}(i+1)$  is labelled with an odd number  $2i+1$ . The PDF associated with state  $2i+1$  is initialised with  $\sigma_{2i+1} = \sqrt{0.016}$  (measured in pixels) and  $\boldsymbol{\mu}_{2i+1} = \mathfrak{O}_{\text{ground}}(i+1)$ . The second state associated with a sample  $\mathfrak{O}_{\text{ground}}(i+1)$  is labelled with an even number  $2i+2$ . The associated PDF of state  $2i+2$  is initialised with  $\sigma_{2i+2} = \sqrt{100000}$  (measured in pixels) and  $\boldsymbol{\mu}_{2i+2} = \mathfrak{O}_{\text{ground}}(i+1)$ . All states are connected to the non-emitting initial and terminating states. An odd-numbered state  $2i+1$  is also connected to state  $2i+2$  (its even “partner” associated with  $\mathfrak{O}_{\text{ground}}(i+1)$ ) and to any other odd-numbered state  $x$  if  $x > (2i+1)$ . An even-numbered state  $2i+2$  is connected to itself and to any odd-numbered state  $x$  if  $x > 2i$ . All transition links leaving a state are equally weighted.

Similar to Section 5.4,  $\mathfrak{O}_{\text{est}}$  is matched to  $\lambda_{\text{L2R}}$  using the Viterbi algorithm. The result is a hidden state sequence  $\mathbf{s} = [s_1, s_2, \dots, s_m]$ . Note that the PDFs of states  $2i+1$  and  $2i+2$  have the same mean  $\mathfrak{O}_{\text{ground}}(i+1)$ , so they overlap approximately where the distance  $\|\mathfrak{O}_{\text{ground}}(i+1) - \mathfrak{O}_{\text{est}}(j)\| = 0.25$ , for  $i \in \{0, \dots, n-1\}$  and  $j \in \{1, \dots, m\}$ . The topology and the PDFs of  $\lambda_{\text{L2R}}$  are now manipulated so that if  $\|\mathfrak{O}_{\text{ground}}(i+1) - \mathfrak{O}_{\text{est}}(j)\| \leq 0.25$ ,  $s_j$  is odd, whereas if  $\|\mathfrak{O}_{\text{ground}}(i+1) - \mathfrak{O}_{\text{est}}(j)\| \geq 0.25$ ,  $s_j$  is even. Hence, if  $s_j$  is even,  $\mathfrak{O}_{\text{est}}(j)$  is either an insertion or a substitution. If  $(s_{j+1} - s_j) > 2$ , where  $s_{j+1}$  and  $s_j$  are both odd numbers, a deletion is identified.

Our resampling scheme (described in Section 3.3) ensures that the distance between any two successive samples in skeleton of a static script is greater than or equal to approximately 1.0. Hence, if each character in the strings INDUSTRY and INTEREST is associated with a 2D coordinate, so that the distance between any two characters is always greater than 0.25 (where

the two PDFs associated with the same sample overlap) we can apply our evaluation protocol. Accordingly, the HMM  $\lambda_{L2R}$  of the word INTEREST is shown in Figure 6.3. Each letter is associated with an even- and odd-numbered state, where the PDFs of even-numbered states are less constrained than the PDFs of odd-numbered states, as indicated on the left-side of the figure. Although the state sequence can start and terminate at any state, transition links connected to the non-emitting initial and terminating states are omitted for the sake of simplicity. Arrows indicate the destinations of the states. For the sake of simplicity, some of the states are also connected to transition links using filled dots and grey lines. The word INDUSTRY is matched to  $\lambda_{L2R}$ , either the state sequence  $\mathbf{s} = [0, 1, 3, 4, 4, 13, 15, 16, 16, 17]$ , or  $\mathbf{s} = [0, 1, 3, 4, 4, 4, 5, 9, 10, 17]$  will be revealed, depending on the 2D values that are associated with each character.



**Figure 6.3:** The left-to-right HMM from our evaluation protocol for the string INTEREST.

PDFs associated with odd-numbered states have tighter standard deviations than PDFs associated with even-numbered states. Transition links are indicated by arrows. Grey lines and filled dots also connect some of the states to transition links. For the sake of simplicity, links that are connected to the non-emitting initial and terminating states are omitted.

An error measure that is invariant to parameterisation is computed as follows: If sample  $\mathfrak{w}_{\text{est}}(j)$  is identified as substitution or insertion, the error is quantified as the erroneous path length  $\|\mathfrak{w}_{\text{est}}(j) - \mathfrak{w}_{\text{est}}(j-1)\|$ . If  $\mathfrak{w}_{\text{est}}(j)$  is a deletion, the error is quantified as  $\|\mathfrak{w}_{\text{ground}}(j) - \mathfrak{w}_{\text{ground}}(j-1)\|$ , i.e., the path length of the ground-truth curve which is absent in the estimated trajectory is calculated. The path lengths between endpoints that bridge gaps in broken lines and the path length between zero-pressure discontinuities are not added to erroneous path lengths. The sum of all errors of a static script is then expressed as a percentage of the script's total ground-truth path length. Note that, due to the addition of insertion errors, the error percentage can exceed 100% so that the accuracy of an estimated pen trajectory can be negative. The final error rate is computed by averaging individual error rates for the static images over the number of static images. Some more examples of the evaluation protocol are presented in Section 6.3.6.

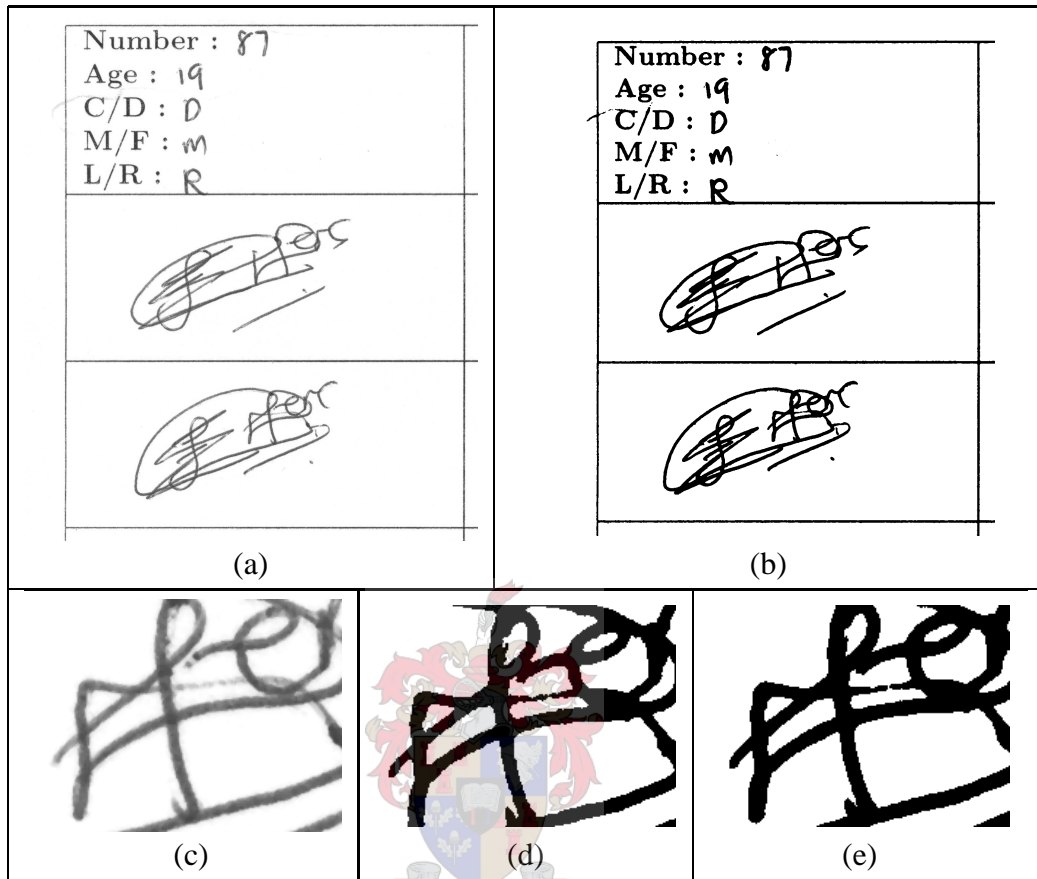
## 6.2 Database for experimental evaluation

As briefly mentioned in Section 6.1, our signature database consists of a total of 814 multi-path signatures for 51 individuals. All signatures were recorded on paper placed on a Wacom UD-0608-R digitising tablet. The paper signatures were scanned as grey-scale images at 600 dpi. The scanner was set to highlight as much detail as possible in light areas, to prevent spurious disconnections if possible. Unfortunately this also introduces more background noise. Most of the noise was easily removed using a median and low-pass filter. A median filter was used, as it is a highly effective and simple method to reduce salt-and-pepper noise in grey-scale images [73]. A median filter sorts the intensities in the  $k \times k$  neighbourhood of a pixel  $\mathbf{p}_i$  and chooses the middle (median) intensity  $i_s$ . The effect is that if the intensity of  $\mathbf{p}_i$  is higher/lower than  $i_s$ , the intensity of  $\mathbf{p}_i$  is replaced with  $i_s$ . Next, a low-pass filter is applied, because, in general, noise manifests as sharp intensity transitions [28]. Low-pass filters with uniform distributions replace the intensity value at each pixel with the average intensity value in a specified neighbourhood. We opted for a Gaussian-shaped low-pass filter, so that central pixels contribute more to the final intensity of a pixel. In general, low-pass filters tend to blur the image slightly due to the smoothing of edges. Hence, image features that constitute small regions in the image (typically noise) attenuate, while significant regions are slightly expanded.

After applying the low-pass filter, the document was binarised, where a global threshold was chosen using the entropy method described in [73]. The *entropy* of an image measures the average, global information content of an image in terms of average bits per pixel [28, 73]. For a grey-scale image with 256 intensity-levels, an entropy of 8 bits/pixel indicates an information-rich image, where the pixel intensities cover the full range equally. An entropy of 0 indicates the presence of a predominant pixel intensity and little variation in the intensity levels. To compute a threshold based on entropy, the image data is separated into two classes for each intensity level. The optimal threshold is then computed as the intensity level for which the sum of entropies for the two classes is the maximum. After binarisation, the static signatures have a line thickness varying between five and ten pixels in parts where the lines do not intersect (depending on the pen pressure.)

Individuals were constrained to write within a grid, e.g., Figure 6.4(a) depicts a typical grey-scale grid extracted from a page in US-SIGBASE. The document's binarised version, after filtering, is shown in Figure 6.4(b). Note that the signatures in Figure 6.4(b) appear thicker than in Figure 6.4(a) due to the entropy-based threshold (at a light grey level)—this has the advantage of accentuating light features, thus preventing broken lines, at the cost of thicker lines. To illustrate the effect of low-pass filtering and binarisation, Figure 6.4(c) zooms in on a part of the second signature in Figure 6.4(a) after median filtering. Figure 6.4(d) depicts Figure 6.4(c)

after binarisation. Figure 6.4(e) depicts Figure 6.4(c) after low-pass filtering followed by binarisation. Note that although Figure 6.4(e) is slightly thicker in parts compared to Figure 6.4(d), it is noticeably smoother.



**Figure 6.4:** An example of a typical scanned document in US-SIGBASE. (a) Two static signatures on a grey-scale document and (b) the document's binarised representative. (c) A part of the second signature in (a) after a median filter has been applied. (d) The binarised image of (c). (e) The binarised image of (c) after low-pass filtering.

More data were originally recorded, but the following practicalities have invalidated some of the data:

1. Some individuals are unable to sign within a grid's constraints. In fact, some individuals tend to write not only over the grid lines, but also over neighbouring signatures, making it impossible to extract the individual signature images automatically without corrupting them.
2. The recording devices may corrupt the data. In fact, a research field is dedicated to this problem; see [77, 85, 76]. Our Hewlett-Packard (HP) scanjet 5470C also causes various slight distortions, and our digitising tablet's pressure resolution is insufficient to capture small pressure values. In some cases, e.g., there are lines in the static scripts that do not occur in the dynamic counterparts of the scripts.

3. The differences between the orientations of the tablet, the paper on the tablet and the paper on the scanner's surface have to be accounted for.
4. In some cases, the paper shifted on the tablet while data were recorded simultaneously on the digitising tablet and paper, thereby causing severe discrepancies between the static signatures and their dynamic counterparts. Hence, we iteratively apply various linear transforms to map the static signatures onto their dynamic counterparts. If a linear transform is unable to determine a satisfactory mapping for a static signature and its dynamic counterpart, the static signature is considered invalid.

## 6.3 Experiments

### 6.3.1 Overview of experiments

A static image is randomly selected for each individual. As stated above, the dynamic counterparts that are used to compute ground-truth trajectories are available for all static images. All the dynamic exemplars (excluding the dynamic counterparts, of course) are used for estimating the pen trajectories of the static scripts, as described in Section 5.4. The estimated pen trajectories are then compared with the ground-truth trajectories to obtain accuracy scores, as described in Section 6.1. The *average accuracy* for a specific experiment is then calculated by averaging the accuracy scores over all the static images in the database. The database and processing steps that have been applied to the dynamic exemplars and static skeletons are specified for each experiment.

A summary of the experiments described in Sections 6.3.2-6.3.5 is presented in Table 6.1. The top of the table indicates that the signatures in US-SIGBASE were used to conduct all the experiments, where US-SIGBASE consists of 51 static images and 712 dynamic exemplars. Subsequently, the configurations of the conducted experiments are shown, where each experiment is numbered in the first column. Due to different skeletonisation and resampling schemes, the number of skeleton samples and the path lengths of the ground-truth trajectories may differ for various configurations. Hence, the second column presents the sum of all the ground-truth path lengths (expressed in pixels) in the database averaged over the number of static scripts for each experiment. The third column presents the number of edges (number of successive samples) in the ground-truth trajectories, averaged over the number of static scripts. A short description of each experiment is provided at the bottom of the table. An overview of these experiments is presented as follows:

Signatures in database:		
Database	Number of images	Number of exemplars
US-SIGBASE	51	712

Experimental configurations:										
	Path length	Number of edges	Skeletonisation		Orientation		Resampling		Training	
			Standard	Pseudo	Radon	PCA	Euclidean	Critical points	Empirical	Writer-Specific
<b>1(a)</b>	2811	2210.4		×	×		×			×
<b>1(b)</b>	2970	2525	×		×		×			×
<b>2(a)</b>	2844.1	776.4		×	×			×		×
<b>2(b)</b>	2844.1	776.4		×		×		×		×
<b>3(a)</b>	2811	2210.4		×	×		×			×
<b>3(b)</b>	2844.1	776.4		×	×			×		×
<b>4(a)</b>	2844.1	776.4		×	×			×		×
<b>4(b)</b>	2844.1	776.4		×	×			×	×	

- 1 Experimental configuration for measuring the effect of two different skeletonisation schemes.
- 2 Experimental configuration for measuring the effect of two different orientation normalisation schemes.
- 3 Experimental configuration for measuring the effect of two different resampling schemes.
- 4 Experimental configuration for measuring the effect of two different training schemes.

**Table 6.1:** *A summary of all the experimental configurations.*

1. The effect of artifacts that frequently occur in the skeletons of static scripts are measured in Section 6.3.2. Specifically, results from our skeletonisation scheme, described in Section 3.1, are compared with results from standard skeletonisation schemes. The abbreviation *standard* in Table 6.1 indicates that the standard thinning algorithm in [29] was employed, whereas *pseudo* indicates that our skeletonisation scheme was employed.
2. The efficacy of our orientation normalisation scheme in the Radon domain, as described in Section 3.2, is evaluated in Section 6.3.3. Results are specifically measured against results from the popular PCA-based scheme. Table 6.1 specifies whether the *Radon* transform or *PCA* was used.
3. The effect of *Euclidean* versus *critical-point* resampling, as described in Section 3.3, is investigated in Section 6.3.4. The resampling scheme for each experiment is specified in Table 6.1.
4. The influence of the training schemes, described in Sections 4.7-4.8, on our HMM is measured in Section 6.3.5. Specifically, the effect of predefined HMM parameters and trained parameters is investigated. Table 6.1 specifies whether *empirical* or trained *writer-specific* standard deviations are used in our PDFs.

**Results.** In the sections to follow, the optimal state sequences from our left-to-right HMMs (described in Section 6.1.2) are used to map estimated trajectories onto their ground-truth trajectories. Quantitative results for each experiment are presented, where the accuracies of the estimated pen trajectories are expressed as percentages of the total ground-truth path lengths, averaged over the number of static images in the database. In Section 6.3.6, a qualitative investigation is described to identify typical errors in estimated pen trajectories.

**Statistical significance.** Where applicable, single-sided statistical significance indicators were calculated to determine whether the differences between the two experimental configurations are attributable to chance [71, 21, 12, 61]. In the sections to follow, the results for two different configurations are compared to each other for each experiment. Thus, for each configuration, the pen trajectories of the 51 static signatures in US-SIGBASE were estimated. For each experiment, the number of times (for the 51 trials) where the accuracy of Configuration A is greater than or equal to the accuracy of Configuration B, i.e., the number of times  $a_A \geq a_B$ , is calculated. If we assume that the two configurations are equally accurate  $p = P(a_A \geq a_B) = 0.5$ . We can model the number of times  $k$  where  $a_A \geq a_B$  with the random variable  $Y$ , where  $Y$  has a binomial distribution. Thus, the probability that  $Y$  is realised at least  $k$  times is given by

$$P(Y \geq k) = \sum_{i=k}^N \binom{N}{i} p^i (1-p)^{N-i} \quad (6.3)$$

$$= 0.5^{51} \sum_{i=k}^{51} \binom{51}{i}, \quad (6.4)$$



where  $N = 51$  is the number of trials (static images to unravel.) The mean of the binomial distribution of  $Y$  is  $\mu_Y = Np = 25.5$ , whereas the standard deviation  $\sigma_Y = \sqrt{Np(1-p)} = 3.57$  (see [71, 61] for further detail.) Equation 6.4 indicates that if  $k$  is relatively large, our assumption that the two configurations are equally accurate is wrong, i.e., where  $P(Y \geq k) \leq T$  if  $k > \mu_Y$ , there is a significant difference between the results for the two configurations. For this application, we let  $T = 5\%$ . For each of the experiments in the sections to follow, we compare the results for the two different configurations and calculate  $k$ ,  $P(Y \geq k)$  and  $|\mu_Y - k| / \sigma_Y$  to compute the statistical significance of the results. For all the experiments  $P(Y \geq k)$  is expressed as a percentage so that it can be easily compared to  $T$ .

### 6.3.2 Experimental results for different skeletonisation schemes

Two sets of experiments are described. The first set uses the sophisticated skeletonisation procedure described in Section 3.1, and yields an average accuracy of 88.3%. The second set uses a standard thinning algorithm [29] without any removal of artifacts, and yields an accuracy of 88.1%. The exact configurations of the experiments are specified at entries 1(a) and 1(b) in Table 6.1.

Results			Statistical significance		
	Skeletonisation	Accuracy	$k$	$P(Y < k)$	$ \mu_Y - k  / \sigma_Y$
(a)	Pseudo	88.3%	25	50%	0.14
(b)	Standard	88.1%			

**Table 6.2:** *Experimental results for (a) our pseudo-skeletonisation scheme and (b) for the standard thinning algorithm in [29].*

Note that although the overall average accuracy of Configuration A is higher than the overall average accuracy of Configuration B,  $k$  is smaller than  $\mu_Y$ . Hence, to obtain an indication of the significance of this result, we test how many times Configuration B outperforms Configuration A for the separate trials. Thus, in this case,  $P(Y < k)$  is computed. Accordingly, the results for this experiment are not regarded as significant.

We can conclude from this experiment that the simple thinning approach achieves a surprisingly high accuracy. In fact, by also observing the statistical significance tests, one cannot say with confidence that our skeletonisation performs better than the standard thinning algorithm. However, our skeletonisation algorithm definitely does not degrade the overall performance of our system, especially considering that it is essential for a good critical-point resampling and to identify simple crossing with confidence.

It is important to realise that the detachment of intersecting lines at simple crossings removes ambiguities, effectively including more context. It is therefore expected that thinner signatures or less complicated handwritten scripts which, in general, have more simple crossings, will result in more accurately estimated pen trajectories. This expectation is endorsed by the results from our system that deals only with single-path scripts (see [58]): Results were generated by converting on-line signatures from the Dolfing database [20, 82]) to thin off-line signatures. The results are shown in Table 6.3 (from [58]), where accuracy scores are expressed as percentages of the ground-truth path lengths that were estimated correctly. Although a different evaluation protocol was used (DP-based, as our evaluation protocol from Section 6.1.2 was not developed yet), it is still evident that, for the Dolfing database, estimated pen trajectories extracted from our pseudo skeletons are approximately 5.6% more accurate than the ones extracted from the standard skeletons. Similar to our results for this experiment, however, the performance increase of the pseudo skeletonisation is only 0.7% for the thick signatures from US-SIGBASE. The only significant difference between the signatures from the Dolfing database and US-SIGBASE were that the Dolfing signatures were significantly thinner than US-SIGBASE signatures. In our opinion, the performance increase of the Dolfing signatures using pseudo skeletons is mostly due to the inclusion of more context at simple crossings. Note that only single-path signatures are unravelled in the experiments of Table 6.3. The single-path signatures are different and fewer in number than the multi-path signatures in US-SIGBASE. Different evaluation protocols were also employed for the two experiments of Table 6.3 and Table 6.2. Thus, the results are slightly different, i.e., an accuracy of 91% is achieved for the single-path signatures in US-SIGBASE, whereas 88.3% is achieved for the multi-path signatures in US-SIGBASE.

	<b>Dolfing</b>	<b>US-SIGBASE</b>	<b>Combined</b>
<b>Number of static images</b>	15	35	50
<b>Number of dynamic exemplars</b>	210	450	660
<b>Accuracy for thinning in [29]</b>	87.2%	90.3%	89.3%
<b>Accuracy for our pseudo skeletons</b>	92.8%	91.0%	91.5%

**Table 6.3:** *Experimental results, showing the average accuracy of recovered pen trajectories for single-path static scripts expressed as the correct percentages of the ground-truth path lengths that were extracted.*

### 6.3.3 Experimental results for different orientation normalisation schemes

Two sets of experiments are described. The first set uses the orientation normalisation scheme in the Radon domain, as described in Section 3.2, and yields an average accuracy of 87.9%. The second set normalises the orientations of the static skeletons and dynamic exemplars using PCA,

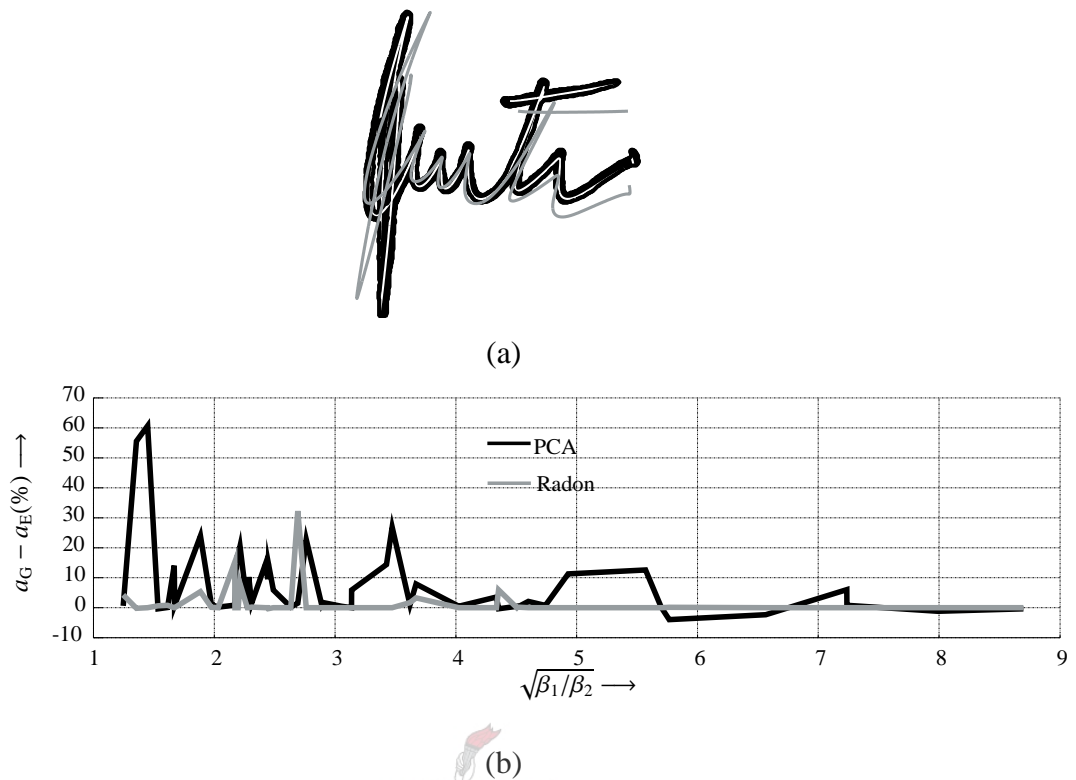
and yields an accuracy of 86.9%. Thus, the orientation scheme in the Radon domain performs slightly better than PCA-based normalisation: there is a 1% improvement in the accuracy with a confidence which is definitely better than chance. The exact configurations of the experiments are specified at entries 2(a) and 2(b) in Table 6.1.

Results			Statistical significance		
	Orientation normalisation	Accuracy	$k$	$P(Y \geq k)$	$ \mu_Y - k  / \sigma_Y$
(a)	Radon	87.9%	35	0.55%	2.7
(b)	PCA	86.9%			

**Table 6.4:** *Experimental results for (a) our Radon-based orientation normalisation scheme and (b) for PCA.*

To illustrate the potential and efficacy of the Radon-based orientation normalisation scheme, as well as our HMM’s robustness to geometric variations, a more direct comparison between the Radon-based and PCA-based schemes has been devised. It is mentioned in Section 6.2 that we iteratively applied various linear transforms to map static signatures onto their dynamic counterparts. The orientation of the dynamic counterpart after this alignment is used as the ground-truth orientation of the dynamic counterpart. We then extract the boundaries of the static script in order to approximate the script with a polygon (see Section 3.1.4.) Hence, the number of dynamic counterpart samples that fall *inside* the approximating polygon of the static script are calculated. This value is expressed as a percentage  $a_G$  of the total number of dynamic counterpart samples. For this experiment, the orientation of the static script remains unchanged, whereas the dynamic counterpart is rotated to align it with the static script using PCA-based and Radon-based rotations. After the dynamic counterpart is rotated using these schemes, the number of dynamic counterpart samples that fall inside the approximating polygon of the static script is recalculated. This value is expressed as a percentage  $a_E$  of the total number of dynamic counterpart samples for each of the orientation normalisation schemes. A typical example is shown in Figure 6.5(a), where a scanned image is depicted in black. The ground-truth alignment of the dynamic counterpart and static script results in an accuracy of  $a_G = 98.5\%$ , i.e., 98.5% of the dynamic counterpart samples (white) fall inside the approximating polygon (the area spanned by the image boundaries.) According to the PCA-based scheme, the dynamic counterpart must be rotated clockwise by  $8.9^\circ$ . According to our Radon-based scheme, no rotation is required. After PCA-based rotation  $a_E = 37.8\%$ . Thus, only 37.8% of the dynamic counterpart samples (grey lines) fall inside the approximated polygon of the static script.

Figure 6.5(b) shows the results for 51 static scripts from US-SIGBASE after PCA-based and Radon-based normalisation. Before rotation,  $a_G$  is calculated, where  $a_G$  is the number of dynamic counterpart samples inside the static script’s approximated polygon, expressed as a percentage of the total number of dynamic counterpart samples. The same procedure is followed



**Figure 6.5:** *Measuring the accuracy of Radon-based versus PCA-based orientation normalisation. (a) A scanned image (black) superimposed on its dynamic counterpart before (white) and after (grey) PCA-based rotation. (b) Results for Radon-based (grey) and PCA-based (black) rotation normalisation, where the difference between the number of dynamic counterpart samples inside the approximating polygons of the original images before and after rotation is shown as a function of the principle components  $\sqrt{\beta_1/\beta_2}$ .*

after Radon-based and PCA-based rotation to calculate  $a_E$ . The difference between  $a_G$  and  $a_E$  is shown as a function of the skeleton principle components ( $\sqrt{\beta_1/\beta_2}$ ) for PCA-based (black) and Radon-based (grey) orientation normalisation. Note that  $\sqrt{\beta_1/\beta_2}$  for a specific signature provides one with an indication of the ratio of the signature length in the direction of the principle axis over the signature length in the direction of the axis that is orthogonal to the principle axis. The highest PCA-based error results from the signature of Figure 6.5(a), where  $a_G - a_E = 60.7\%$  and  $\sqrt{\beta_1/\beta_2} = 1.41$ . In general, the graph shows that for PCA,  $a_G - a_E$  is especially high where  $\sqrt{\beta_1/\beta_2}$  is small. It is therefore deduced that PCA-based rotation becomes unreliable where principle components are similar. Averaged over the total number of static scripts (51) the average  $a_G - a_E$  for PCA-based rotation is 7.7%, whereas for the Radon-based rotation it is 1.5%. Thus, despite the relatively few signatures where  $\sqrt{\beta_1/\beta_2}$  is small (there are only 3 signatures where  $\sqrt{\beta_1/\beta_2} \leq 1.5$ ), the Radon-based orientation normalisation scheme already outperforms the PCA-based scheme according to this evaluation. It is also clear that, although the Radon-based scheme is more accurate than the PCA-based scheme, only a 1% performance increase in

our HMM’s performance is achieved for the Radon-based rotation. This experiment is therefore a good indication that our HMM is quite robust to global orientation variations. It is, however, beneficial to employ the Radon-based rotation, especially where  $\sqrt{\beta_1/\beta_2} \leq 1.5$ .

Our HMM’s robustness to rotational variations is explained as follows:

- Firstly, it is observed that our HMM is rather robust to local rotational variations. This robustness is especially evident from the good results obtained for the standard pixel-based thinning technique (discussed in Section 6.3.2), where the angular differences between connected samples are multiples of  $45^\circ$ . This local robustness is attributed to:
  - The standard deviation  $\sigma'_v$  in our directional PDF component  $\mathcal{N}(\mathbf{u}_{ij}^v, \sigma'_v)$  (see Section 4.8) compensates for local directional differences between a dynamic exemplar and a static skeleton.
  - The Viterbi algorithm computes a *globally* optimal path, thereby providing a robustness to local differences.
- Secondly, our HMM is rather robust to global rotational variations. This robustness is especially evident from the experiments described in this section. Equation 4.5 indicates that the directional feature  $\mathbf{u}_{ij}^v$  in  $\mathcal{N}(\mathbf{u}_{ij}^v, \sigma'_v)$  is *relative* and normalised. Thus, in most cases, the same global optimal path for a wide scope of global rotational shifts between the same dynamic exemplar and static script results. However, it is important to note that  $\delta$  from (5.4) decreases for the same state sequence if the relative global rotational shift is increased. Since the selection of the optimal pen trajectory for a static script is based on  $\delta$ , rotational variations may lead to a sub-optimal selection.

### 6.3.4 Experimental results for different resampling schemes

Two sets of experiments are described. The first set resamples the static skeletons and dynamic exemplars so that the distance between any two successive samples is roughly the same, i.e., a Euclidean resampling is used. Specifically, the distance between two successive pixels is approximately 1 pixel, as described in Section 3.3. For the first set, an average of 88.3% of the estimated trajectory path lengths are correct. The second set uses the critical-point resampling described in Section 3.3, and achieves an average accuracy of 87.9% (expressed as a percentage of the ground-truth path lengths.) The exact configurations of the experiments are specified in entries 3(a) and 3(b) in Table 6.1.

Firstly, it is evident from Table 6.5 that the accuracies for critical-point and Euclidean resampling are approximately the same. In fact, according to the statistical significance test the differences are chiefly attributable to chance. Section 3.3 has shown that our critical-point resam-

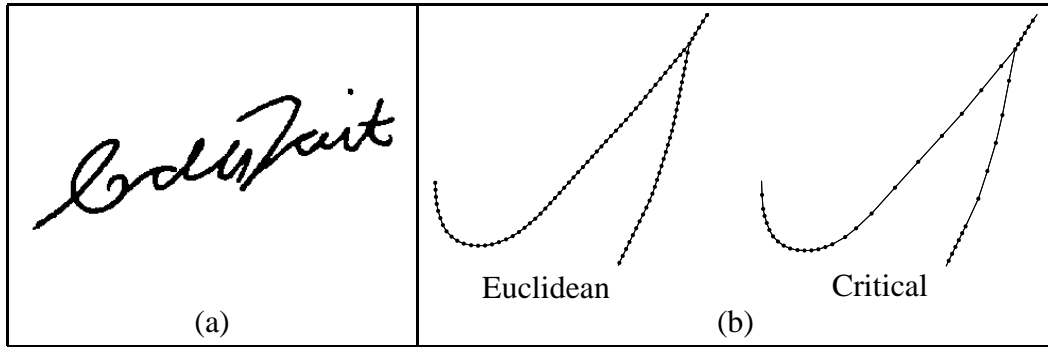
Results			Statistical significance		
	Resampling	Accuracy	$k$	$P(Y \geq k)$	$ \mu_Y - k  / \sigma_Y$
(a)	Euclidean	88.3%	26	50%	0.14
(b)	Critical point	87.9%			

**Table 6.5:** *Experimental results for (a) Euclidean resampling and (b) critical-point resampling.*

pling reduces samples on a parametric curve by minimising the information loss inherent in the point-reduction scheme. Thus, it is expected that the results for the two configurations would be similar if all the high-curvature points (containing important information) of the samples on the Euclidean resampled curve were selected by our critical-point resampled curve. However, it should also be noted that the critical-point resampling includes more context for our HMMs. This inclusion of context also contributes to the good results of the critical-point resampling scheme and can be explained as follows. As mentioned earlier, skip-link and duration states allow for a factor two difference between the number of samples in corresponding segments of static skeletons and dynamic exemplars. A straight line and a curved line with the same path length have approximately the same number of samples when using Euclidean resampling. Although the directional PDF components suppress the matching of curved and straight lines, it is still possible. When using our critical-point resampling, the number of samples on a straight line and a curved line with the same path length might differ with a factor of more than two, thereby further restricting the matching of these curved and straight lines.

Secondly, a significant increase in the speed of our system is observed when using critical-point instead of Euclidean resampling. This increase in speed is simply due to the decrease in the number of samples when using critical-point resampling. The reduction of computational requirements is demonstrated using a typical signature in US-SIGBASE, shown in Figure 6.6. During the recording process of the static signature in Figure 6.6(a) and its dynamic exemplars, the signatory was constrained to sign within a bounding box of approximately 50 mm × 20 mm (somewhat larger than what would normally be allowed on documents such as bank cheques.)

All experiments were conducted on a 1.6 GHz AMD XP1900+. The signature in Figure 6.6(a) has approximately 1378 samples in its skeleton using the Euclidean resampling scheme. Its final HMM has 12 294 states and 73 502 transition links. Practically, it takes approximately 31 seconds to estimate the pen trajectory of Figure 6.6(a) based on a different Euclidean resampled dynamic exemplar with 1652 samples. The critical-point skeleton of Figure 6.6(b) results from selecting high curvature points from the Euclidean resampled skeleton and reducing the number of samples on straight lines, as described in Section 3.3. The final critical-point resampled skeleton of Figure 6.6(a) has 572 samples, 5040 HMM states and 29 966 transition links.



**Figure 6.6:** A typical example from US-SIGBASE to explain experimental results for different resampling schemes. (a) A static script with (b) a fragment of its Euclidean and critical-point resampled skeleton.

Practically, it takes approximately 7 seconds to estimate its pen trajectory based on a different dynamic exemplar with 676 critical points. The computational requirements of the system will continue to decrease if the number of samples is reduced. However, it is anticipated that a cut-off would surface where the trade-off between the accuracy and the speed of the system becomes less rewarding. It should also be noted that we are using generic software with no optimisation for this particular application. It is expected that the execution time can also be substantially reduced with code optimised for this application. Other optimisation suggestions are made in the final chapter.

### 6.3.5 Experimental results for different training schemes

Two sets of experiments are described. The first set uses the HMM parameters defined in Section 4.8. Thus, for each individual, the unique PDF parameters  $\sigma'_p$  and  $\sigma'_v$  are calculated using the training scheme described in Section 4.8. Recall that these parameters are derived from  $\sigma_p$  and  $\sigma_v$  by utilising pre-recorded dynamic exemplar information, and that they allow for size and orientation variations in static scripts. The first set achieves an average accuracy rate of 87.9%. The second set uses the empirical HMM parameters defined in Section 4.7. Hence, no training is applied to the position and directional standard deviations of the PDFs, so that  $\sigma_p = 17$  and  $\sigma_v = 0.2$  for all the individuals in US-SIGBASE. The second set achieves an average accuracy rate of 87%. The exact configurations of the experiments are specified in entries 4(a) and 4(b) in Table 6.1.

No significant difference between the two training schemes is observed. It is therefore concluded that our empirical standard deviations are well chosen. To roughly estimate how well the empirical standard deviations correspond with the writer-specific standard deviations, the average  $\sigma'_p$  and  $\sigma'_v$  have been computed over all the individuals. The average  $\sigma'_p$  is 13.3 and the

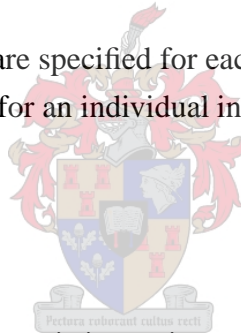
Results			Statistical significance		
	Training	Accuracy	$k$	$P(Y \geq k)$	$ \mu_Y - k  / \sigma_Y$
(a)	Writer-specific	87.9%	28	28.8%	0.7
(b)	Empirical	87%			

**Table 6.6:** Experimental results for (a) our writer-specific training scheme and (b) for empirically determined PDF parameters.

average  $\sigma'_V$  is 0.15, which are relatively similar to the empirical values  $\sigma_P = 17$  and  $\sigma_V = 0.2$ . It is also concluded that our system is not highly sensitive to allographic variations, i.e., the system that uses the same position and directional standard deviations for different individuals performs surprisingly well. However, writer-specific training is beneficial when considering the following:

- There is a slight increase in the performance of the system when writer-specific training is employed.
- The parameters  $\sigma'_P$  and  $\sigma'_V$  are specified for each individual, and may therefore be useful as biometric measurements for an individual in other applications.

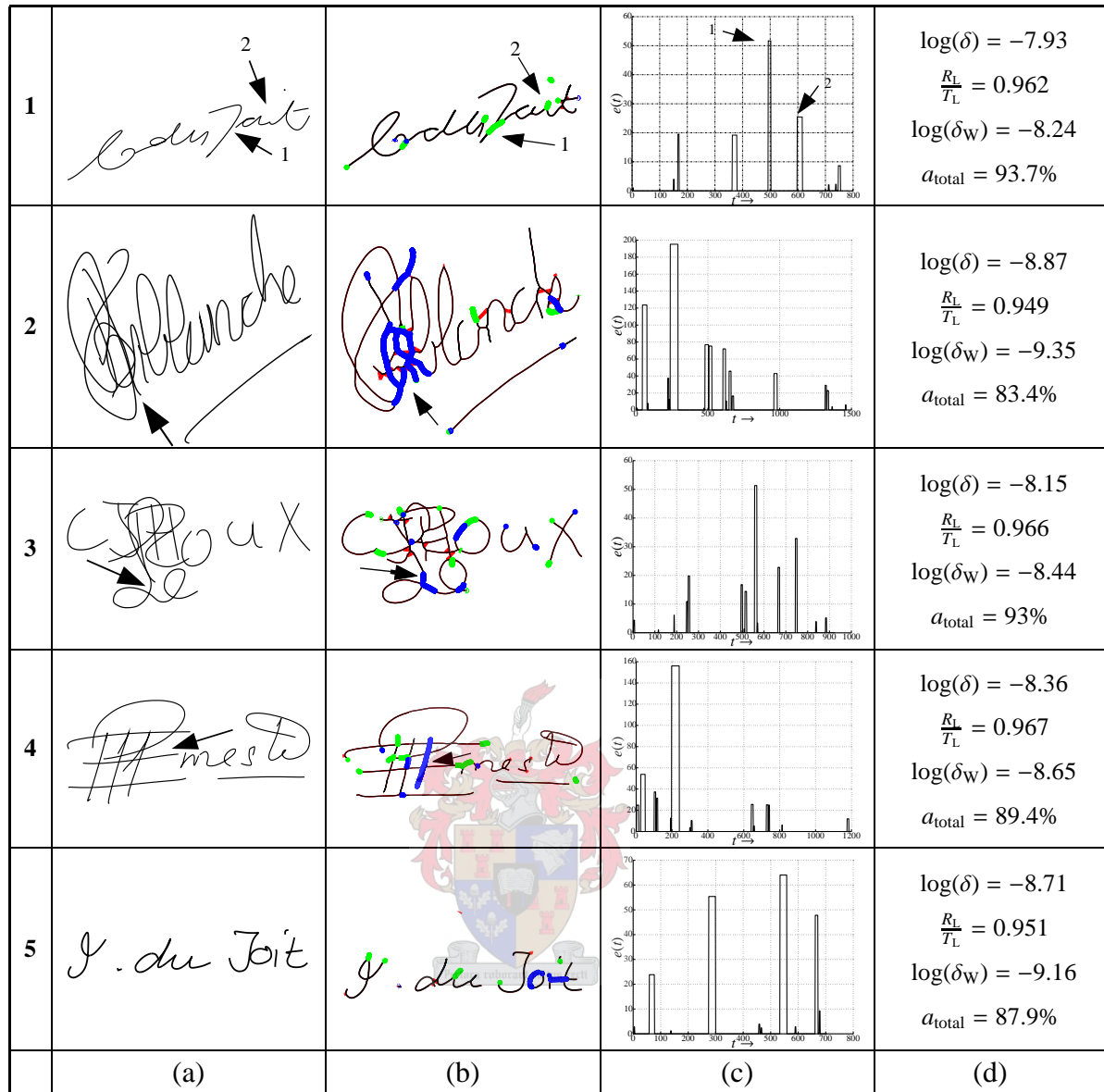
### 6.3.6 Typical errors



As expected, the main cause of errors is inconsistencies between a static image and a dynamic exemplar. Specifically, the system is prone to errors in regions where a line segment is present in either the dynamic exemplar or static image, but absent in the other. Figure 6.7 shows examples of typical errors that are encountered when estimating the pen trajectories of static scripts. Examples of typical dynamic exemplars and skeletonised static images from US-SIGBASE are shown in Figures 6.7(a) and (b), respectively. Black lines in Figure 6.7(b) identifies curves that match the ground-truth trajectories. Red lines indicate which skeleton samples were not extracted during the computation of the estimated and ground-truth trajectories. As expected, the red lines are mostly part of web-like structures. Erroneous skeleton samples, i.e., estimated samples that do not match the ground-truths, are indicated using green and blue lines. Specifically, deletion errors are green, whereas substitution and insertion errors are blue.

Figure 6.7(d) presents the accuracies  $a_{\text{total}}$  of the estimated pen trajectories of Figure 6.7(b), as calculated from  $e(t)$  in Figure 6.7(c). The error function  $e(t)$  is computed by identifying all the errors, as described in Section 6.1.2, and calculating the sum of all the errors that are part of the same non-zero pulse. Thus, each continuous blue or green curve in Figure 6.7(b) corresponds to a non-zero pulse in Figure 6.7(c), where the height of the pulse is the total error of the





**Figure 6.7:** Typical errors that are encountered in estimated pen trajectories. (a) Dynamic exemplars. (b) Skeleton samples that are estimated correctly (black lines), skeleton samples that were not extracted during the calculation of the ground-truth and the estimated trajectories (red lines), errors due to deletions (green lines), and errors due to substitutions and insertions (blue lines.) (c) Graphs of the cost functions that are used to quantify errors from (b). (d) Quantitative results when the dynamic exemplars of (a) are matched to the HMMs derived from (b).

continuous erroneous curve. Figure 6.7(d) presents the log likelihoods  $\log(\delta)$  after matching the dynamic exemplars of Figure 6.7(a) to the HMMs derived from Figure 6.7(b). The values for  $\frac{R_L}{T_L}$  and  $\log(\delta_W)$  from (5.5) are also shown.

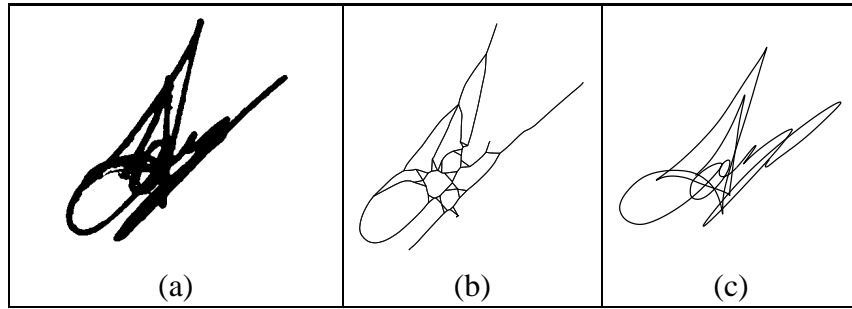
Samples in a static image that are absent in the corresponding dynamic exemplar cannot be ex-

tracted from the static image and therefore cause deletion errors. The reason is that the dynamic exemplar does not provide the necessary information. The curves at arrows 1 and 2 of skeleton 1 in Figure 6.7(b) do not occur in the dynamic exemplar, as indicated by the corresponding arrows in Figure 6.7(a). The two corresponding deletion errors in  $e(t)$  are indicated in Figure 6.7(c). Note that the first error has the longest path length and therefore results in the highest pulse in  $e(t)$ , as shown in Figure 6.7(c). The second error's pulse is wider, as the erroneous curve has a high curvature point and therefore consists of more skeleton samples than the first error.

Other errors are caused by dynamic exemplar substitutions and insertions, as indicated by the blue lines in Figure 6.7(b). An example of a substitution is indicated by the arrow of dynamic exemplar 2 in Figure 6.7(a). The resulting error is indicated by the blue lines at the arrow of Figure 6.7(b). Examples of insertions are at the arrows of dynamic exemplars 3 and 4. Corresponding errors are indicated by the blue lines and arrows of Figure 6.7(b). The error at the arrow of signature 3 is better comprehended when notice is taken of the differences between the static skeleton and dynamic exemplar, as depicted by the animation in Figure 5.7(k). Animation examples are also provided on the attached CD; see Appendix A for further detail.

Size differences between corresponding segments can also cause errors, e.g., the character “o” of signature 5. Because the “o” in the dynamic exemplar is bigger than the “o” in the static skeleton, a part of the “o” in the static skeleton is traversed twice, resulting in the error shown in blue. Note, however, that our HMM does accommodate size differences in most cases, e.g., the leftmost loop of signature 1.

It is observed that the accuracy of our system is not primarily linked to the complexity of the static scripts. Our system's efficacy depends more on the consistency between an individual's signatures. More dynamic exemplars are required for inconsistent individuals than for consistent individuals to produce accurate results. In some cases, however, the images may have an excessive line thickness relative to the size of the signature. Hence, information loss due to multiple crossings in small areas becomes severe, making it difficult or impossible to unravel the image. Figures 6.8(a)-(c) show an example of such a signature. The original image, its skeleton, and the dynamic exemplar corresponding best to it are shown in Figures 6.8(a)-(c), respectively. Not only is the shape of the image in Figure 6.8(a) corrupted in the middle region, but the dynamic exemplars have inconsistent pen movements in corresponding regions. Despite the obvious difficulties, a total path length of approximately 64% is recovered correctly.



**Figure 6.8:** (a) A complicated static image to unravel, with (b) its skeleton and (c) the dynamic exemplar corresponding best to it.

## 6.4 Comparison with existing approaches

Compared to the related work listed in Table 2.1, we do not impose any significant restrictions on the range of scripts that can be dealt with. Our HMM topology enables an estimated pen-trajectory to start and end at any skeleton sample of the static script and enables us to model turning points. No restrictions are imposed at intersections (except in cases of very simple crossings; see Section 4.5). Furthermore, the PDF components that take the pen pressure of the dynamic exemplars into consideration enable us to accommodate scripts that consist of multi-path trajectories. The only other method that can establish a local correspondence between a static script and a dynamic exemplar is the method of Guo et al. [31]. However, their method differs significantly from ours, and they have no quantitative results. Experimental results indicate that our approach is not highly sensitive to skeletonisation artifacts, whereas many existing techniques are. Compared to existing techniques that use dynamic exemplar information [31, 51], basic preprocessing makes our technique scale, translation and rotationally invariant. Experimental results show that our approach is not highly sensitive to allographic variations. Pen sequence variations can be accommodated depending on the consistency of a specific individual’s handwriting and the number of available dynamic exemplars.

The only technique that provides a quantitative evaluation protocol and comparative results is that of Jäger [38]. Firstly, it should be noted that the travelling salesman approach (employed by Jäger [38]) seems to produce the most accurate results of all the graph-theoretical approaches. However, the travelling salesman problem cannot be solved efficiently; see Chapter 2. The Viterbi algorithm, on the other hand, is efficient. The computational cost of the Viterbi algorithm is  $O(Tm)$ , where  $m$  is the number of non-zero transition probabilities at each time step [7]. Thus, e.g., in the worst case, when all the transition probabilities are non-zero at each time step for our HMM derived from the critical-point skeleton of Figure 6.6,  $m = 29\,966$  and  $T = 676$ , so that  $mT = 14\,728\,688$ . Solving the travelling salesman problem (Jäger’s approach [38]) for the same skeleton with 572 edges, causes a combinatorial explosion. In fact, when solving the

travelling salesman problem  $O(n!)$  computations evaluate all the paths so that  $n! = 39\,916\,800$  for a skeleton with only 11 edges. Hence, it is concluded that our approach is computationally more effective than methods that solve the travelling salesman problem.

Table 6.7 shows the experimental configurations for our approach and Jäger’s approach [38] in rows labelled (a) and (b), respectively.

<b>Database characteristics:</b>					
	<b>Database</b>	<b>Average path length (pixels)</b>	<b>Average number of edges</b>	<b>Number of images</b>	<b>Number of exemplars</b>
(a)	US-SIGBASE	2844.1	776.4	51	712
(b)	Jäger [38]	–	17.7	6934	6934
<b>Data configuration:</b>					
	<b>Skeleton</b>	<b>Orientation</b>	<b>Resampling</b>	<b>Training</b>	<b>Evaluation</b>
(a)	Pseudo (webs)	Radon	High curvature	Writer	DP
(b)	Non-standard	–	Cross/Endpoint	–	DP
<b>Results:</b>					
	<b>Total accuracy:</b> (Levenshtein distance expressed as a % of the number of edges)				
(a)	83.1%				
(b)	78%				

**Table 6.7:** A comparison with an existing approach. The data configurations that were used to establish a comparison with Jäger [38] are shown.

A few essential differences between the two configurations displayed in Table 6.7 should be taken into consideration before the results can be discussed:

1. It should be noted that Jäger [38] generated results by converting on-line words into off-line words. Although we conducted validation tests, there are still discrepancies between the on-line and off-line signatures in our database that result in additional errors in some cases. In general, off-line scripts that are generated from on-line scripts result in skeletons with less artifacts and background noise than skeletons derived from scanned-in scripts.
2. Jäger [38] is primarily concerned with words, whereas we focus on signatures. In general, characters and words do not have as many complicated intersections and lines that are traced more than twice as signatures do.
3. Jäger [38] employs a skeletonisation different from ours. However, they also use a method to remove artifacts from the skeletons.
4. Jäger [38] segments the skeletons of static scripts at crosspoints and endpoints. In our database, pen-up and pen-down events can occur at any skeleton sample, and there are

many web-like structures in our skeletons, making it rather difficult to employ the same segmentation.

5. Another important point to note is that Jäger’s approach [38] cannot indicate where pen-up and pen-down events occur—it is assumed that every word is written without the pen being lifted. The author notes that arbitrary pen-up and pen-down events pose serious problems. Thus, it is expected that the approach followed by Jäger [38] will struggle to unravel the signatures in our database, where the positions of pen-up and pen-down events cannot be easily predicted.
6. The evaluation protocol of Jäger [38] has been employed, as described in Section 6.1. The total Levenshtein distance is expressed as a percentage of the total number of edges in the ground truths and averaged over the number of static images in the database. This error measure is, however, not invariant to parameterisation, and may therefore cause discrepancies between the results that are compared.

From the above remarks, it is clear that there are so many significant differences between the two experimental configurations in Table 6.7 that an accurate comparison is not really possible. In fact, our system is penalised due to our resampling and the complicated scripts that constitute US-SIGBASE. However, an extremely rough estimate indicates that our approach outperforms Jäger [38] by approximately 5%. Note that this is the absolute difference between our results and the results of Jäger [38]. However, in view of a relative difference, our algorithm reduces the error rate with approximately 23% compared to Jäger. It is anticipated that our system would perform significantly better for characters and words, depending on the application.

## 6.5 Summary

In summary, this chapter has accomplished the following:

1. To compute the efficacy of an estimated pen trajectory, the ground-truth and estimated pen trajectories must be compared with each other. Existing evaluation protocols that establish such a quantitative comparison have been investigated. Specifically, the two existing protocols by Lau et al. [52] and Jäger [38] have been examined to establish the general beneficial characteristics of an evaluation protocol. Accordingly, our evaluation protocol has the following desired characteristics:
  - Ground-truth and estimated pen trajectories having different numbers of samples can be handled.
  - Our evaluation protocol is invariant to parameterisation.

- Our evaluation protocol establishes a rather unique mapping between the estimated and ground-truth trajectories, so that errors can be identified easily and with confidence.
  - Our evaluation protocol is easy to implement, so that it can be used as a standardised evaluation protocol for trajectory estimation algorithms.
2. A database that contains both the on-line and off-line versions of signatures was created. As far as we are aware, no such standardised signature database exists. It is important to determine if a system's performance degrades in a practical environment. Thus, the most influential processing steps and practical implications when recording such a database have been briefly pointed out in this chapter.
  3. Various experiments have been described in this chapter. The following important results, which contribute to the establishment of our system's robustness, emerged from the experiments:
    - Our system is not highly sensitive to skeletonisation artifacts.
    - Experimental results indicate that the Radon-based orientation normalisation scheme performs better than the PCA-based scheme. A sensitive measurement scheme indicates that the Radon-based rotation is more accurate than the PCA-based rotation. However, the performance of our system for the PCA-based rotation remains very good. This indicates that our system is rather robust to rotational variations.
    - Experimental results indicate that the empirical PDF parameters were suitably chosen and that our system is not highly sensitive to allographic variations.
  4. The experiments show that critical-point resampling is preferable over Euclidean resampling: without a significant decrease in the performance of the system, a significant increase in the speed is acquired. This critical-point resampling is, however, dependent on high-quality skeletons.
  5. Compared to existing techniques, our system imposes few restrictions on the range of scripts that can be handled. An extremely rough quantitative comparison indicates that our system outperforms the system of Jäger [38]. Our estimation algorithm is also much more efficient.

# Chapter 7

## Conclusions and future work

### 7.1 Conclusions

In this dissertation a probabilistic model was developed for extracting the pen trajectory of a static script. Although our emphasis was on handwritten signatures, the technique should also be valuable for more general scripts such as handwritten words, characters and line drawings. The HMM encapsulates a two-level representation of a static image. Both levels take context into account by modelling time dependencies. On the higher level, the sub-images and single-paths that constitute a static script are modelled, whereas on the lower level the skeleton samples that constitute the sub-images are modelled. Our principle achievements in this design are the following:

- On the lower level, only single-path trajectories can be extracted. Thus, if applied to a script consisting of multiple sub-images, a single trajectory with continuous non-zero pressure will be extracted from one of the sub-images. Time dependencies are modelled by exploiting the virtues of higher-order HMMs. An effective implementation of higher-order HMMs is made practically realisable through the ORED algorithm [22, 21]. Knowledge of past context, available from second-order HMMs, is especially useful for modelling the following important facets:
  - Knowledge of past context enables the enforcement of continuous pen motions (without abrupt erratic turns) on straight lines (see Figure 4.5.)
  - Simple intersections are detected within the HMM and turning points are also modelled by exploiting past context (see Figure 4.7.)
  - Knowledge of past events enables the inclusion of a directional feature in our PDFs. This directional feature is similar to the velocity feature used in various on-line handwriting applications, and is a crucial factor for good results.

- On the higher-level, the virtues of hierarchical HMMs are exploited. Time dependencies between the different sub-images that constitute a static script *and* the different paths between them are modelled. This provides an elegant solution to the following problems:
  - The gateway to model pen pressure in handwritten static scripts is opened when using HHMMs. The HHMMs enable the estimation of locations where the individual lifted the pen, thus allowing us to deal with multi-path static scripts. This generalises our approach so that no restrictions are placed on the number of sub-images that may constitute a static script, or the number of single-path trajectories that generated the script.
  - By manipulating the topology of our HHMM we can deal effectively with situations where unexpected disconnections occur in static scripts.
- Our higher-level and lower-level HMMs are designed to incorporate prior knowledge from pre-recorded dynamic exemplars. The Viterbi algorithm matches a dynamic exemplar to the final HMM and determines the most likely state sequence, which can be translated into the most likely pen trajectory. The Viterbi algorithm is globally optimised, making it highly suitable for resolving local ambiguities in a static script.

From a more general point of view, this dissertation has shown how the topology and PDFs of HMMs can be engineered to meet application-specific requirements. In short, our model can deal with a wide variety of static scripts robustly and effectively. Throughout this dissertation, we have remarked on and referenced back to the outlined contributions in Section 1.5. Our foremost results are now summarised as follows:

- Our experiments compared the ground-truth with the estimated pen trajectories of static scripts, where a database with on-line and off-line representatives of signatures was used to conduct experiments. The experimental results indicate that an average of approximately 88% of the estimated trajectory path lengths from this database correspond to the ground-truth path lengths (averaged over all the images.)
- Many systems that estimate the pen trajectories of static scripts and other off-line handwriting recognition techniques suffer from a high sensitivity to artifacts. Experiments indicate that our system is rather robust with respect to the type of skeletonisation used. Moreover, we find that complex static images, even ones that are hard to unravel with the eye, do not pose serious problems. Of course, some thick-lined images can be corrupted to such an extent during skeletonisation that information loss becomes severe, making it very difficult to unravel the image.
- The main source of errors is inconsistencies between a dynamic exemplar and a static image, which are inevitable for handwritten documents. Although our system is not immune to dissimilarities and ambiguities, it takes global context into account, making it



more robust than algorithms that rely heavily on local correspondences. It is important to note, however, that in those cases where errors do occur, one still has access to local correspondences. This can be useful in a signature verification application, as it allows the comparison of only those parts of the signature that were accurately recovered.

- Experiments indicate that our HMM is rather robust to rotational variations. However, it is still beneficial to employ our Radon-based scheme in cases where the principle components of the script are similar.
- The effect of different resampling schemes was measured. Experimental results indicate that a dense sampling rate at high curvature points and the reduction of samples on straight lines tend to complement our approach: a substantial increase in speed without a substantial decrease in accuracy is effected.
- We also experimented with writer-specific training schemes, where the position and directional standard deviations were estimated from the dynamic exemplars of each individual. Our experiments indicate that our basic writer-specific training scheme is not significantly better than our system where the same empirical deviations are used for all the individuals. This indicates that our system is insensitive to allographic variations and that the empirically obtained parameter values are adequate. Although prior conventional scaling and translation schemes are invoked, embedded PDF variances also contribute to our system's ability to cope with signatures of different sizes.
- The availability of dynamic exemplars enables us to incorporate pressure information to deal with multi-path scripts. In fact, to our knowledge, no existing technique includes dynamic pressure information. A practical problem arises when spurious disconnections (broken lines) are encountered. By virtue of our HMM topology, we can deal with most spurious disconnections, turning points and signatures with different numbers of samples.
- Compared to existing techniques, our approach does not impose severe restrictions on the scripts that can be handled—it depends chiefly on the availability of dynamic exemplars, easily obtained in practise. A pen trajectory can, e.g., start at any position in a static script, depending on where the dynamic exemplars of the individual that generated the script start. There are also no limitations on the number of times the pen can revisit a line in the static script. It has been shown that our HMM-based approach is computationally more viable than methods that solve the travelling salesman problem. Furthermore, there is sufficient scope to reduce the computational time of our system even more, as discussed in the next section. Very few techniques provide quantitative results, and no standard database exists with which to make accurate comparisons with existing techniques. However, using the same evaluation protocol (which is not invariant to parameterisation), and under very different circumstances, a rough estimate indicates that our system outperforms the system of Jäger [38] by approximately 5%.
- Different evaluation protocols were used to calculate the efficacy of estimated pen trajec-

tories. Compared to existing techniques, our evaluation protocol can deal with ground-truth and estimated pen trajectories with different numbers of samples, and is invariant to parameterisation. Additionally, it also calculates a more unique local correspondence between an estimated and ground-truth trajectory, which makes the identification and classification of errors possible.

## 7.2 Future work

### 7.2.1 Reducing the computational complexity

It is felt that the speed of the system can still be increased substantially. Section 6.3.4 has shown that systems based on the critical-point resampling of Section 3.3 are faster than and achieve approximately the same accuracy rate as systems based on Euclidean resampling. A further improvement in the computational time could be effected by a reduction in the number of samples in the static images and dynamic exemplars. However, some experiments must then be conducted to calculate the degree of accuracy sacrificed for an increase in speed. General optimisation of code for this application would also effect a reduction of execution time (see Section 6.3.4.)

The Viterbi algorithm is used to estimate the pen trajectory of a static image. The computational cost of the Viterbi algorithm is  $O(Tm)$ , where  $m$  is the number of non-zero transition probabilities at each time step [7] and  $T$  is the number of samples in the dynamic exemplar that are matched to the HMM. There are other approximate but faster algorithms that could be used instead of the Viterbi algorithm (see [7].)

The number of transition links in our HMM can be reduced by establishing region boundaries that enclose the typical starting and terminating positions of a static script. For example, the region that spans the typical starting positions of a specific individual's dynamic exemplars can be computed. The appropriate transition weights can then be set to zero to prohibit a pen trajectory from starting at skeleton samples outside the computed region. Note that this is also a realisation of training. This training scheme would not, however, necessarily improve the accuracy of results, but would decrease the computational time.

## 7.2.2 Exploiting pressure information

To improve the accuracy of our system, pen pressure information can be exploited to a greater extent. As mentioned in Chapter 1, pressure information is frequently embedded in the grey levels of a static image (also see [84, 78].)

Initial stages of our skeletonisation scheme, described in Section 3.1, extracts the boundaries of static scripts from their binarised images. Digitisation noise on the boundaries has a significant effect on the smoothness and accuracy of final skeletons. Hence, much attention is paid to smoothing the image boundaries. In our opinion, a better approach would be to extract the boundaries of the static scripts directly from their grey-scale images before the calculation of the skeletons. Section 3.1.4 has shown that smoother boundaries reduce peripheral artifacts. Thus, a further improvement could be obtained if the boundaries are smoothed on a sub-pixel basis while they are extracted. Subsequent smoothing procedures can then be removed, e.g., the empirical formula of (3.1). Under ideal circumstances, no peripheral artifacts would appear in the skeletons of the static scripts, thereby also reducing the risk of removing actual image features.

With our current HMM configuration (described in Section 4.8) the only normalised pressure values that are matched to our pressure PDF components are 0 and 1. Thus, “hard decisions” are made by the pressure PDF components to determine if the pen trajectory must be extracted from a specific sub-image in the static script, or if the zero-pressure state must be entered (when a pen-up event occurs.) A more accurate approach would be to let the pen pressure PDF components of the zero-pressure states and other emitting states overlap slightly. This would be especially beneficial in cases where the dynamic pen pressure is so small that some lines occur in a static script which are absent in the script’s dynamic counterpart. However, the resolution of the pen pressure quantisation levels of our digitising tablet is too low for such investigations. It can record only 256 pressure levels, whereas more advanced tablets can record at least 1024 pressure levels. Thus, to measure the effect of overlapping PDF components practically, we would have to record a new database with a more up-to-date tablet.

As shown in Section 2.2, Spagnolo et al. [78] developed a system that performs 3D acquisition of documents. This device is particularly useful for computing the relationship between the depth and grey-levels of static handwritten images, as well as the pen pressure that generated the images, especially in cases where the line densities at intersections are high. It is anticipated that this additional information can be included in our PDF components to complement our approach.

### 7.2.3 Extending the training schemes

Our current efforts, not described in this thesis, investigate training schemes, where specific standard deviations are trained for positional and directional PDF components of emitting states. In this dissertation, such training schemes are referred to as *localised* training schemes. Again, localised training has to be adapted for this application. The aim of localised training is to include more flexibility for geometric variations within our HMM, as corresponding parts in the dynamic exemplars and static scripts frequently have inconsistent sizes and orientations. The following pertinent observations have been made from preliminary experiments:

1. Data sparsity and the association of skeleton samples with emitting states prohibit training of transition weights and the PDF means  $\mu^P$  and  $\mu^V$ . As a first attempt we intend to model geometric variations on specific curves within the static script more accurately. It is therefore beneficial to train only  $\sigma'_p$  and  $\sigma'_v$  from Section 4.8 for each PDF.
2. Usually the HMM parameters are estimated by the method of *maximum likelihood* (ML). However, due to data sparsity *maximum a posteriori* (MAP) estimation is more appropriate, so that  $\sigma'_p$  and  $\sigma'_v$  can be used as priors for each PDF; see [27, 74] for more detail on MAP estimation.
3. Due to data sparsity and our specific HMM structure, it is beneficial to group observations from neighbouring emitting states (excluding the zero-pressure state) together. Since these emitting states are associated with skeleton samples, this would result in grouping together connected skeleton samples within a specified proximity of each other.
4. Due to typical pen sequence variations, it is possible that dynamic exemplar samples that do not, in reality, correspond to each other are joined to update a specific PDF's parameters. Thus, it seems beneficial to employ a strategy that can identify mismatched dynamic exemplar samples before parameter estimates.

### 7.2.4 Applications: Signature verification and character recognition




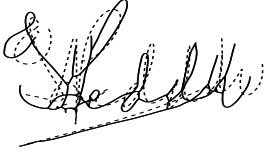

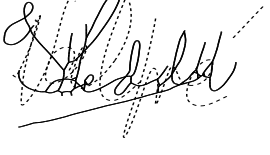
Our HMMs for static scripts are potentially useful for a variety of applications. The HMMs are rather robust to geometric and allographic variations in static scripts, indicating potential to generalise them for arbitrary static shapes. Our system also has the ability to separate patterns belonging to different classes. It may therefore be useful in applications where it is necessary to classify arbitrary shapes. Two possible applications where our system may be integrated is discussed in this section: An off-line verification system and an off-line character recognition system.

**Signature verification.** Off-line signature verification systems that extract dynamic information from static scripts are rare. As far as we are aware, only Guo et al. [31] establish local correspondences between dynamic exemplars and static signatures for verification purposes. Furthermore, by estimating the pen trajectories of static signatures, one has access to many on-line signature verification techniques. In general, these on-line techniques differ noticeably from off-line verification techniques. Thus, it is anticipated our system can be combined with a wide range of other different signature verification techniques to produce even better results.

Recall from Section 1.3.1 that a good pattern recognition system increases the separability between test and training patterns belonging to different classes. Preliminary results indicate that the likelihood  $\delta_w$  can already be used as a confidence measure of the separability between patterns from different classes, and can therefore be useful to identify forgeries, especially casual forgeries, in an off-line signature verification system. Figure 7.1 depicts a typical scenario where the pen trajectory of a static script is estimated using a correct dynamic exemplar (created by the same individual) and an incorrect dynamic exemplar (created by a different individual.) Figure 7.1(b) shows the skeleton of the static signature in Figure 7.1(a). Figure 7.1(c) shows the results when the dynamic exemplar of Figure 7.1(c), by the same individual, is compared with the HMM derived from Figure 7.1(b). Thus, the situation is simulated where a genuine static signature must be verified in an off-line signature verification system, where the static signature is the test pattern and the dynamic exemplar the training pattern. Note that all accuracy scores are expressed as a percentage of the ground-truth path length of the script in (a). Figure 7.1(d) shows the results when the dynamic exemplar of Figure 7.1(d), by a different individual in the database, is compared with the HMM derived from Figure 7.1(b). Thus, the situation is simulated where a casually forged static signature must be verified in an off-line signature verification system. Note that the signatures have been normalised before comparison, as shown in Figures 7.1(e)-(f), where the skeleton of Figure 7.1(b) is rendered as a solid line and the signatures of Figures 7.1(c)-(d) are rendered as dashed lines.

Figures 7.1(c)-(d) illustrate our system's ability to separate test and training patterns from different classes: Even though the accuracy of the estimated pen trajectory derived from Figure 7.1(c) is not 100%, there is already a noticeable separation between the log likelihood  $\log(\delta)$  of Figures 7.1(c) and (d). Note from (5.5) how  $\log(\delta_w)$  increases this separability since only 47.4% of Figure 7.1(b) is recovered when Figure 7.1(d) is used to estimate the pen trajectory of Figure 7.1(b). Note that the accuracy of Figure 7.1(d) is negative, as determined by our evaluation protocol. Specifically, the 68.3% deletion errors are added to the 32.4% substitution and insertion errors resulting in a negative accuracy score.

**Character/word Recognition.** Although our pen trajectory estimation algorithm can also be applied to a character/word recognition system, a large vocabulary of characters and cursive

 <p>(a)</p>			 <p>(e)</p>
 <p>(b)</p>	$\log(\delta) = -8.75$ $\frac{R_L}{T_L} = 0.9511$ $\log(\delta_w) = -9.2$ Accuracy: 83.8% (c)	$\log(\delta) = -18.56$ $\frac{R_L}{T_L} = 0.474$ $\log(\delta_w) = -39.18$ Accuracy: -0.7% (d)	 <p>(f)</p>

**Figure 7.1:** Illustration of our system’s ability to separate test and training patterns from different classes. (a) A static script with (b) its skeleton. (c) A dynamic exemplar by the same individual who generated (b). (d) A dynamic exemplar by a different individual than the one who generated (b). (e)-(f) The skeleton (solid line) from (b) superimposed on the dynamic exemplars (dashed lines) of (c) and (d) after normalisation.

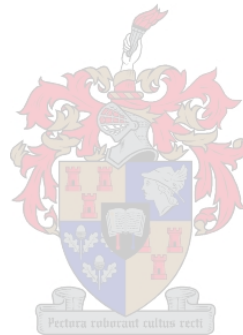
words may result in exhaustive searches for the most suitable pen trajectories (the likelihoods of each character/word may be used as a confidence measure to recognise them.) It would therefore be more beneficial to recognise characters/words from a restricted library (see [31, 64]), or by simplifying our model. We have, e.g., done some promising preliminary experiments, where our HMM was adapted to model the boundaries of shapes. In such cases, our HMM can be greatly simplified, as the boundaries of shapes are non-overlapping, closed, contiguous trajectories. For cursive words, prior segmentation of the static words into separate characters may also be beneficial.

### 7.2.5 Recording more data and conducting more experiments

**Recording more data.** Our system was tested on a carefully designed database. Although every effort was made to ensure that the results quoted in this thesis are true reflections of the capabilities of our system, more extensive tests on more data will improve the confidence in the performance of the system. Cursive characters and words can also be recorded to test our system’s performance in a character/word recognition environment. Application-specific simplifications can then be made to our HMM.

**Conducting more experiments.** As our experiments were conducted using primarily ball-point pens, it will be interesting to record static images using different pen types, e.g., pencils, crayons, permanent markers and even paint brushes. One can then quantify the robustness of

our system relative to the type of pen. In this application, Euclidean and critical-point resampling schemes were employed. For the Euclidean resampling scheme the length of a curve segment is always approximately one (measured in pixels), whereas for the critical-point resampling scheme the length of a curve segment is always approximately five or one (measured in pixels.) Thus, the length of a curve segment is dependent on its resampling. Future work can include the investigation of different resampling schemes resulting in more curve-length variations. One can then include more context by adding another curve-length component in our feature vectors. In general, one can also investigate the utilisation of other features, e.g., a curvature feature, to enhance the performance of our system.



# Bibliography

- [1] Abou-Moustafa, K. T., Cheriet, M., and Suen, C. Y., “On the Structure of Hidden Markov Models.” *Pattern Recognition Letters*, June 2004, Vol. 25, No. 8, pp. 923–931.
- [2] Abuhaiba, I. and Ahmed, P., “Restoration of Temporal Information in Off-Line Arabic Handwriting.” *Pattern Recognition*, July 1993, Vol. 26, No. 7, pp. 1009–1017.
- [3] Al-Ohali, Y., Cheriet, M., and Suen, C. Y., “Dynamic Observations and Dynamic State Termination for Off-Line Handwritten Word Recognition using HMM.” in *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, 2002.
- [4] Al-Ohali, Y., Cheriet, M., and Suen, C. Y., “Efficient Estimation of Pen Trajectory from Off-Line Handwritten Words.” in *Proceedings of the International Conference on Pattern Recognition*, pp. 323–326, 2002.
- [5] Allen, C. R. and Navarro, A., “The Recognition of Roman Handwritten Characters using Dynamic Information Recovery.” in *IPA97*, pp. 741–745, 1997.
- [6] Allen, R. E. (Ed.), *The Concise Oxford Dictionary of Current English*. Oxford University Press, 1990.
- [7] Bengio, Y., “Markovian Models for Sequential Data.” *Neural Computing Surveys*, 1999, Vol. 2, pp. 129–162.
- [8] Bengio, Y. and Frasconi, P., “Diffusion of Context and Credit Information in Markovian Models.” *Journal of Artificial Intelligence Research*, 1995, Vol. 3, pp. 249–270.
- [9] Boccignone, G., Chianese, A., Cordella, L. P., and Marcelli, A., “Recovering Dynamic Information from Static Handwriting.” *Pattern Recognition*, 1993, Vol. 26, No. 3, pp. 409–418.
- [10] Chaikin, G., “An Algorithm for High Speed Curve Generation.” *Computer Vision, Graphics and Image Processing*, 1974, Vol. 3, pp. 346–349.



- [11] Chang, H. and Yan, H., "Analysis of Stroke Structures of Handwritten Chinese Characters." *IEEE Transactions on Systems, Man, and Cybernetics B*, February 1999, Vol. 29, No. 1, pp. 47–61.
- [12] Coetzer, J., *Off-line Signature Verification*. PhD thesis, Stellenbosch University Press, 2005.
- [13] Dawoud, A. and Kamel, M., "New Approach for the Skeletonization of Handwritten Characters in Gray-Scale Images." in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 1233–1237, 2003.
- [14] De Berg, M., Van Kreveld, M., Overmars, M., and Schwarzkopf, O., *Computational Geometry Algorithms and Applications, 2nd Edition*. Springer, 1997.
- [15] De Boor, C., "Good Approximation by Splines with Variable knots." in *Conference on the Numerical Solution of Differential Equations*, pp. 12–20, 1973.
- [16] Deller, J. R., Hansen, J. H. L., and Proakis, J. G., *Discrete-Time Processing of Speech Signals*. IEEE Press, 2000.
- [17] Doermann, D. S., *Document Image Understanding: Integrating Recovery and Interpretation*. PhD thesis, Center for Automation Research, University of Maryland, 1993.
- [18] Doermann, D. S. and Rosenfeld, A., "Recovery of Temporal Information from Static Images of Handwriting." *International Journal of Computer Vision*, 1995, Vol. 15, pp. 143–164.
- [19] Doermann, D. and Rosenfeld, A., "The Interpretation and Reconstruction of Interfering Strokes." in *Frontiers in Handwriting Recognition*, pp. 41–50, 1993.
- [20] Dolfing, J. G. A., *Handwriting Recognition and Verification: A Hidden Markov Approach*. PhD thesis, Eindhoven, Netherlands, 1998.
- [21] Du Preez, J. A., *Efficient High-Order Hidden Markov Modelling*. PhD thesis, Stellenbosch University Press, 1997.
- [22] Du Preez, J. A., "Efficient Training of High-Order Hidden Markov Models using First-Order Representations." *Computer Speech and Language*, January 1998, Vol. 12, No. 1, pp. 23–39.
- [23] Du Preez, J. A. and Weber, D. M., "The Integration and Training of Arbitrary-Order HMMs." *Australian Journal on Intelligent Information Processing Systems (ICSLP98 Special Issue)*, 1998, Vol. 5, No. 4, pp. 261–268.

- [24] Fairhurst, M. C., “Document Identity, Authentication and Ownership: The Future of Biometric Verification.” in *Proceedings of the International Conference on Document Analysis and Recognition*, 2003.
- [25] Fine, S., Singer, Y., and Tishby, N., “The Hierarchical Hidden Markov Model: Analysis and Applications.” *Machine Learning*, 1998, Vol. 32, No. 1, pp. 41–62.
- [26] Fränti, P., Mednionogov, A., and Kälviäinen, H., “Hough Transform for Rotation Invariant Matching of Line-drawing Images.” in *Proceedings of the International Conference on Pattern Recognition*, pp. 389–392, 2000.
- [27] Gauvain, J.-L. and Lee, C.-H., “Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains.” *IEEE Transactions on Speech and Audio Processing*, 1994, Vol. 2, pp. 291–298.
- [28] Gonzalez, R. C. and Woods, R. E., *Digital Image Processing*. Addison-Wesley, 1992.
- [29] Gonzalez, R. C., Woods, R. E., and Eddins, S. L., *Digital Image Processing using Matlab*, Ch. 9. Pearson Prentice Hall, 2004.
- [30] Guo, J. K., *Forgery Detection by Local Correspondence*. PhD thesis, Center for Automation Research, University of Maryland, 2000.
- [31] Guo, J. K., Doerman, D., and Rosenfeld, A., “Forgery Detection by Local Correspondence.” *International Journal of Pattern Recognition and Artificial Intelligence*, 2001, Vol. 15, No. 4, pp. 579–641.
- [32] Heileman, G., *Data Structures, Algorithms, and Object-Orientated Programming*. MIT Press and McGraw-Hill, 1992.
- [33] Huang, T. and Yasuhara, M., “A Total Stroke SLALOM Method for Searching for the Optimal Drawing Order of Off-Line Handwriting.” in *Proceedings IEEE Systems, Man and Cybernetics*, vol. 3, pp. 2789 – 2794, Oct 1995.
- [34] Huang, W., Rong, G., and Bian, Z., “Strokes Recovering from Static Handwriting.” in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 861–864, IEEE Computer Society, 1995.
- [35] Jain, A. K., *Fundamentals of Digital Image Processing*. Prentice-Hall, 1989.
- [36] Jain, A. K., Duin, R. P. W., and Mao, J., “Statistical Pattern Recognition: A Review.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, January 2000, Vol. 22, No. 1, pp. 4–37.

- [37] Jäger, S., “A Psychomotor Method for Tracking Handwriting.” in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 528–531, IEEE Computer Society, 1997.
- [38] Jäger, S., *Recovering Dynamic Information from Static, Handwritten Word Images*. PhD thesis, University of Freiburg, 1998.
- [39] Kapoor, R., Bagai, D., and S., K. T., “A New Algorithm for Skew Detection and Correction.” *Pattern Recognition Letters*, August 2004, Vol. 25, pp. 1215–1229.
- [40] Kato, Y. and Yasuhara, M., “Recovery of Drawing Order from Single-Stroke Handwriting Images.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 2000, Vol. 22, No. 9, pp. 938–949.
- [41] Kato, Y. and Yasuhara, M., “Recovery of Drawing Order from Scanned Images of Multi-Stroke Handwriting.” in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 261–264, IEEE Computer Society, 1999.
- [42] Kegl, B. and Krzyzak, A., “Piecewise Linear Skeletonization using Principal Curves.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, January 2002, Vol. 24, No. 1, pp. 59–74.
- [43] Lallican, P. M., Viard-Gaudin, C., and Knerr, S., “From Off-Line to On-Line Handwriting Recognition.” in *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pp. 303–312, International Unipen Foundation, 2000.
- [44] Lallican, P. M. and Viard-Gaudin, C., “A Kalman Approach for Stroke Order Recovering from Off-Line Handwriting.” in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 519–523, IEEE Computer Society, 1997.
- [45] Lallican, P. M., Viard-Gaudin, C., Knerr, S., and Binter, P., “The IRESTE ON-OFF (IRONOFF) Handwritten Image Database.” in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 455–458, 1999.
- [46] Lam, L. and Suen, C. Y., “Automatic Comparison of Skeletons by Shape Matching Methods.” *International Journal of Pattern Recognition and Artificial Intelligence*, 1993, Vol. 7, No. 5, pp. 1271–1286.
- [47] Lam, L. and Suen, C. Y., “An Evaluation of Parallel Thinning Algorithms for Character-Recognition.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 1995, Vol. 17, No. 9, pp. 914–919.

- [48] Lam, L., Lee, S., and Suen, C. Y., "Thinning Methodologies-A Comprehensive Survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992, Vol. 14, No. 9, pp. 869–885.
- [49] Lane, J. M. and Riesenfeld, R. F., "A Theoretical Development for the Computer Generation of Piecewise Polynomial Surfaces." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1980, Vol. 2, pp. 34–46.
- [50] Lau, K. K., Yuen, P. C., and Tang, Y. Y., "Stroke Extraction and Stroke Sequence Estimation on Signatures." in *Proceedings of the International Conference on Pattern Recognition*, pp. 119–122, 2002.
- [51] Lau, K. K., Yuen, P. C., and Tang, Y. Y., "Recovery of Writing Sequence of Static Images of Handwriting using UWM." in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 1123–1128, 2003.
- [52] Lau, K. K., Yuen, P. C., and Tang, Y. Y., "Directed Connection Measurement for Evaluating Reconstructed Stroke Sequences in Handwriting Images." *Pattern Recognition*, 2005, Vol. 38, pp. 323–339.
- [53] Le Riche, P., "Handwritten Signature Verification: A Hidden Markov Model Approach." Master's thesis, Stellenbosch University, 2000.
- [54] Lee, S. and Pan, J. C., "Offline Tracing and Representation of Signatures." *IEEE Transactions on Systems, Man, and Cybernetics*, July 1992, Vol. 22, No. 4, pp. 755–771.
- [55] Martinez, T. and Schulten, K., "Topology Representing Networks." *Neural Networks*, 1994, Vol. 7, No. 3, pp. 507–522.
- [56] Munich, M. E. and Perona, P., "Visual Identification by Signature Tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, February 2003, Vol. 25, No. 2, pp. 200–217.
- [57] Nel, E., Du Preez, J. A., and Herbst, B. M., "Estimating the Pen Trajectories of Static Scripts using Hidden Markov Models." in *Proceedings of the International Conference on Document Analysis and Recognition*, 2005.
- [58] Nel, E., Du Preez, J. A., and Herbst, B. M., "Estimating the Pen Trajectories of Static Signatures using Hidden Markov Models." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005, Vol. 27, pp. 1733–1746.
- [59] Pan, J. C. and Lee, S., "Offline Tracing and Representation of Signatures." in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 679–680, 1991.

- [60] Park, H., Sin, B., Moon, J., and Lee, S., "A 2-D HMM Method for Offline Handwritten Character Recognition." *International Journal of Pattern Recognition and Artificial Intelligence*, 2001, Vol. 15, No. 1, pp. 91–105.
- [61] Peebles, P. Z., *Probability, Random Variables, and Random Signal Principles*. Third edition. McGraw-Hill, 1993.
- [62] Plamondon, R. and Maarse, F. J., "An Evaluation of Motor Models of Handwriting." *IEEE Transactions on Systems, Man, and Cybernetics B*, 1989, Vol. 19, No. 5, pp. 1060–1072.
- [63] Plamondon, R. and Privitera, C. M., "The Segmentation of Cursive Handwriting: An Approach Based on Off-Line Recovery of the Motor-Temporal Information." *IEEE Transactions on Image Processing*, January 1999, Vol. 8, No. 1, pp. 80–91.
- [64] Plamondon, R. and Srihari, S. N., "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, January 2000, Vol. 22, No. 1, pp. 63–84.
- [65] Prasad, L., "Morphological Analysis of Shapes." tech. rep., Los Alamos National Laboratory, Los Alamos, July 1997.
- [66] Privitera, C. M. and Plamondon, R., "A System for Scanning and Segmenting Cursively Handwritten Words into Basic Strokes." in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 1047–1050, 1995.
- [67] Qiao, Y. and Yasuhara, M., "Recovering Dynamic Information from Static Handwritten Images." in *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, pp. 118–123, IEEE Computer Society, 2004.
- [68] Rabiner, L. R. and Juang, B. H., "An Introduction to Hidden Markov Models." *IEEE ASSP Magazine*, January 1986, pp. 4–16.
- [69] Rocha, J., "Perceptually Stable Regions for Arbitrary Polygons." *IEEE Transactions on Systems, Man, and Cybernetics B*, February 2003, Vol. 33, No. 1, pp. 165–171.
- [70] Rosenthal, A. S., Hu, J., and Brown, M. K., "Size and Orientation Normalization of On-Line Handwriting using Hough Transform." in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pp. 3077–3080, IEEE Computer Society, 1997.
- [71] Scheaffer, R. L. and McClave, J. T., *Probability and Statistics for Engineers*. Wadsworth Publishing Company, 1995.

- [72] Sedgewick, R., *Algorithms*. Addison-Wesley Publishing Company, 1946.
- [73] Seul, M., O’Gorman, L., and Sammon, M. S., *Practical Algorithms for Image Analysis: Description, Examples, and Code*. Cambridge University Press, 2000.
- [74] Shinoda, K. and Lee, C.-H., “A Structural Bayes Approach to Speaker Adaptation.” *IEEE Transactions on Speech and Audio Processing*, March 2001, Vol. 9, pp. 276–287.
- [75] Sindle, C., “Handwritten Signature Verification using Hidden Markov Models.” Master’s thesis, Stellenbosch University, 2003.
- [76] Smith, E. H. B., “Scanner Parameter Estimation using Bilevel Scans of Star Charts.” in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 1164–1168, 2001.
- [77] Smith, E., “Characterization of Image Degradation caused by Scanning.” *Pattern Recognition Letters*, November 1998, Vol. 19, No. 13, pp. 1191–1197.
- [78] Spagnolo, G. S., Simonetti, C., and Cozzella, L., “Superposed Strokes Analysis by Conoscopic Holography as an Aid for a Handwriting Expert.” *Journal of Optics A: Pure and Applied Optics*, 2004, Vol. 6, pp. 869–874.
- [79] T. Stheinherz, N. I., “A Special Skeletonization Algorithm for Cursive Words.” in *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pp. 529–534, International Unipen Foundation, 2000.
- [80] Tang, Y. Y. and You, X., “Skeletonization of Ribbon-Like Shapes Based on a New Wavelet Function.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 2003, Vol. 25, No. 9, pp. 1118–1133.
- [81] Terrades, O. R. and Valveny, E., “Radon Transform for Lineal Symbol Representation.” in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 195–199, 2003.
- [82] Van Oosterhout, J. J. G. M., Dolfing, J. G. A., and Aarts, E. H. L., “On-Line Signature Verification with Hidden Markov Models.” in *Proceedings of the International Conference on Pattern Recognition*, pp. 1309–1312, 1998.
- [83] Verwer, B., van Vliet, L., and Verbeek, P., “Binary and Grey-Value Skeletons: Metrics and Algorithms.” *PRAI*, 1993, Vol. 7, No. 5, pp. 1287–1308.
- [84] Wirotius, M., Seropian, A., and Vincent, N., “Writer Identification from Gray Level Distribution.” in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 1168–1172, 2003.

- [85] Zhou, J. Y., Lopresti, D., Sarkar, P., and Nagy, G., *Spatial Sampling Effects on Scanned 2-D Patterns*. Singapore: World Scientific, 1997.
- [86] Zou, J. J. and Yan, H., “Skeletonization of Ribbon-Like Shapes Based on Regularity and Singularity Analysis.” *IEEE Transactions on Systems, Man, and Cybernetics B*, June 2001, Vol. 31, No. 3, pp. 401–407.
- [87] Zou, J. J. and Yan, H., “Vectorization of Cartoon Drawings.” in *Selected Papers from Pan-Sydney Workshop on Visual Information Processing* (Eades, P. and Jin, J. (Eds)), (Sydney, Australia), ACS, 2001.



# Appendix A

## The static scripts in US-SIGBASE: Animation examples.

**Slides on the attached CD.** The signatures that were randomly chosen to generate the results in Section 6 are shown in Figure A.1. The static scripts of the users are numbered and correspond to the static scripts that are presented in the Portable Document Format (PDF) file “slides.pdf” on the attached CD. The file “slides.pdf” also contains animation examples for each user. The slides can be viewed with Adobe Reader on a Windows or Linux platform. The slides are best viewed full screen (the shortcut key is usually CTRL-L). Quick references to the different slides can be found on the top of each slide. To quickly reach the slide for user 40, e.g, mouse-click on the section reference “User 31-45” on the top of any slide and then on the sub-section reference “User 39-41”, and scroll down one slide to slide 40.

**Slide description.** Each slide shows the static script for a user (also shown in Figure A.1) as well as the dynamic exemplar that obtained the highest  $\delta_w$  (of all the dynamic exemplars for the same individual). Refer to Section 5.4. The skeleton (solid lines) is also shown superimposed on the dynamic exemplar (dashed lines) to illustrate geometric variation between them, after prior preprocessing alignment (see Chapter 3). The Adobe Reader tools, e.g, the zoom utility, can be used to inspect the shown signatures closely. The accuracy for each script’s estimated pen trajectory is shown, and some Moving Picture Experts Group (MPEG) format animation examples are provided. Click once on the play buttons to view them.

It is important to note that Adobe Reader opens an appropriate program to view the MPEG animations. It was verified that the format of the animations is compatible with Windows Media Player and Winamp (on a Windows platforms) and Mplayer (on a Linux platform). It was reported that some versions of Windows Media Player distort the last frame of the animations slightly. If Adobe Reader does not display the animations, one should verify the following:



- Verify that a suitable MPEG viewing program is installed.
- Verify that the suitable MPEG viewing program is set as the default application associated with MPEG files for the operation system in question.

**Animation 1: View ground-truth pen trajectory.** The first button launches an animation that illustrates the result after matching the dynamic *counterpart* of the static script to the HMM of the script's skeleton, as described in Section 6.1. Hence, the ground-truth trajectory (bottom trajectory) of the static script is extracted from the script skeleton (bottom signature) by establishing a pointwise correspondence (red dots) with the dynamic counterpart (top signature and trajectory).

**Animation 2: View estimated pen trajectory.** The second button launches an animation that illustrates the result after matching the dynamic *exemplar* (also shown on the slide) of the static script to the HMM of the script's skeleton, as described in Chapters 4 and 5. The result is a local correspondence (red dots) between the dynamic exemplar and the skeleton. The bottom trajectory, extracted from the script skeleton, is the estimated pen trajectory of the static script, where the top trajectory is the dynamic exemplar.

**Animation 3: View evaluation trajectories.** The third button launches the result when the estimated pen trajectory (bottom trajectory from Animation 2) is matched to the script's ground-truth trajectory (bottom trajectory from Animation 1). Errors are represented the same as in Figure 6.7(c), i.e., the ground-truth and estimated trajectories are superimposed, where green lines indicate deletions and blue lines indicate substitutions and insertions. The error functions (top) are computed exactly the same as the error functions in Figure 6.7(d). Thus, an erroneous curve corresponds to a continuous pulse in the error function, where the pulse height is equal to the path length of the erroneous curve. The rendered red dots correspond to errors of zero and they are used to indicate where ground-truth and estimated trajectories (bottom) correspond. The rendered yellow dots are used to indicate inception points of erroneous curves. They only become visible when the red dots traverse erroneous curves. In such cases the error signal (top) is non-zero.



**Figure A.1:** The static scripts in US-SIGBASE that were unravelled to generate experimental results.

# Index

- Allographic variations, 4
- Application-specific skeleton, 43
- Approximating polygon, 34, 103
  
- Behavioural biometric measurement, 1
- Biometric measurement, 1
- Boundary edge, 32
  
- Chaikin corner cutting, 43
- Chinese postman problem, 21
- Classification, 5
- Complete graph, 20
- Complicated J-T, 39
- Critical point resampling, 49
- Crosspoint, 33, 64
- Cycle, 20
  
- Delaunay triangulation, 32
- Deletion, 92
- Duration state, 64
- Dynamic counterpart, 2
- Dynamic exemplar, 4
  
- Edge, 20
- Efficient solution, 22
- Emitting state, 53
- End-triangle (E-T), 36
- Endpoint, 32, 64
- End region, 32
- Entropy, 96
- Euclidean resampled curve, 49
- Eulerian cycle, 21
- Eulerian path, 21
- External triangles, 35
- External edge, 35
- Feature vector, 5, 54
- Fully-connected ergodic, 77
  
- General-purpose skeleton, 43
- Geometric variations, 4
- Graph, 20
- Graph-theoretical methods, 20
- Ground-truth trajectory, 9
- Guo et al. [31], 26
  
- Hamilton cycle, 21
- Hamilton path, 21
- HHMM, 76
- HMM, 6
- HMM State, 6
- Insertion, 92
- Internal edge, 35
- Internal triangles, 35
- Intersection artifacts, 31
- Intersection region, 33
- Isolated-triangle (I-T), 36
  
- Jäger,S [38], 24
- Junction-triangle (J-T), 36
  
- Lau et al. [51], 26
- Left-to-right topology, 70
- Levenshtein distance, 24, 91
- Line graph, 20
- Line segment, 60
- Localised training, 120
- Local correspondence methods, 26



- Long ribbon, 37  
 Maximum likelihood estimation, 72, 120  
 Merging, 33  
 Multi-path static script, 3, 76  
 Multi-path trajectory, 3  
 Neighbouring states, 54  
 Nodes/Vertices, 20  
 Non-emitting state, 53  
 Normal-triangle (N-T), 36  
 Off-line handwriting, 1  
 On-line handwriting, 1  
 Order of an HMM, 6, 53  
 ORED algorithm, 57  
 Orientation of handwriting, 46  
 Path, 20  
 PCA, 46  
 Peripheral artifacts, 31  
 Physiological biometric measurement, 1  
 Predecessor state, 58  
 Primary skeleton, 36  
 Probability Density Function (PDF), 6  
 Pseudo skeleton, 32  
 Radon transform, 47  
 Ribbon, 37  
 Rule-based methods, 16  
 Segment point, 60  
 Self-loop, 55  
 Self-loop state, 63  
 Sequence variations, 4  
 Short ribbon, 37  
 Single-path static script, 3  
 Single-path trajectory, 3  
 Skeleton, 8, 30  
 Skeletonisation algorithms, 30  
 Skip-link, 55  
 Skip-link state, 63  
 Spanning tree, 20  
 Spurious disconnections, 11, 80  
 Standard skeleton, 31  
 Sub-image, 76  
 Substitution, 92  
 Thinning algorithms, 30  
 Tied PDF, 58  
 Topology, 6  
 Training, 5  
 Transition links, 6  
 Travelling salesman problem, 21  
 Tree, 20  
 Uncomplicated J-T, 40  
 Unstable J-T, 40  
 Web-like structures, 32  
 Weighted graph, 20  
 Writer-specific training, 70  
 Zero pressure state, 78

