THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
L'Université Paul Sabatier

Présentée et soutenue par :

Gaëtan ANDRÉ

Le 11 décembre 2013

Titre :

Modélisation oscillatoire de l'écriture manuscrite.

Oscillatory modelling of handwriting.

Ecole doctorale et discipline ou spécialité :
ED MITT : Intelligence Artificielle

Unité de recherche :
IRIT - UMR 5505

Directeurs de Thèse :
Jean-Christophe BUISSON, Professeur (IRIT, INPT-ENSEEIHT, Toulouse)

Rapporteurs :
Patrick HÉNAFF, Professeur (LORIA, Mines de Nancy)
Gregor SCHÖNER, Professeur (IFN, Rhür Universität, Bochum)

Examinateurs :
Denis MOTTET, Professeur (M2H, Université Montpellier 1)
Jean-Charles QUINTON, Maître de Conférences (Institut Pascal, Université Blaise Pascal, Clermont-Ferrand)
Pier-Giorgio ZANONE, Professeur (PRISSMH, Université Toulouse 3)

Remerciements

Merci tout d'abord à mon directeur de thèse, Jean-Christophe Buisson, qui m'a soutenu en toutes circonstances. Merci aussi à Pier-Giorgio Zanone, qui a su m'intéresser à l'étude de la coordination et de la motricité. Ils sont à l'origine de nombreuses idées reprises dans cette thèse. Merci à eux d'avoir pris le temps de relire (plusieurs fois pour le premier cité) mon manuscrit.

Merci aux membres du LAPMA ayant participés à ce travail : Vivianne Kostrubiec, avec son expertise des statistiques, et Jean-Michel Albaret. Merci aussi à Frédéric Messine, pour sa collaboration sur la partie optimisation.

Merci aux rapporteurs, Patrick Hénaff et Gregor Schöner, d'avoir accepté et pris le temps d'évaluer ce manuscrit. Merci aux autres membres du jury, Denis Mottet et Jean-Charles Quinton, de s'intéresser à ce travail et de venir en débattre.

Merci à tous mes collègues de l'IRIT, en particulier aux thésards du bureau d'en face et aux secrétaires qui m'ont toujours facilité la vie dans mes démarches.

Un dernier merci enfin à ma relectrice spéciale ainsi qu'à toute ma famille.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Motivation

L'écriture peut être vue comme l'habileté ultime [André, Kostrubiec, Buisson, Albaret, et Zanone]. Cette capacité nécessite un long apprentissage et fait appel à de nombreuses facultés du corps et de l'intelligence. Non seulement le poignet et l'avant-bras sont impliqués, mais l'ensemble de la posture du scripteur a un effet sur la production de l'écriture. Au delà de son aspect essentiellement ancré dans le corps, cette tâche fait appel à nombre de facultés cognitives : la vision, la mémoire, la motricité, le langage, le raisonnement et bien sûr la communication.

L'étude de l'écriture et plus particulièrement de la trace écrite est un vaste domaine qui a mené à l'élaboration de nombreux modèles. Ceux-ci peuvent-être divisés en deux groupes. Le premier groupe est celui des modèles applicatifs, qui accompagnent généralement la résolution d'un problème concret, le plus étudié étant la reconnaissance de caractères [Plamondon, 2000; Vinciarelli, 2002]. Ces modèles s'appuient généralement sur des méthodes, souvent basées sur les chaînes de Markov [Plötz et Fink, 2009; Hu et Brown, 1996; Nel, du Preez, et Herbst, 2005; Artières, Marukatat, et Gallinari, 2007], les réseaux de neurones [Kherallah, Haddad, Alimi, et Mitiche, 2008] ou même la géométrie fractale [Vincent, Seropian, et Stamon, 2005], performantes pour ce qu'elles sont censées faire mais qui apportent peu à notre compréhension de la production de trace écrite chez l'humain. Le problème de la synthèse a aussi été étudié [Wang, Wu, Xu, Shum, et Ji, 2002; Lin et Wan, 2007], le but étant de produire un texte en imitant le style d'un scripteur, mais les modèles produits ont les mêmes défauts. En revanche, le deuxième groupe de modèles, a pour but d'expliquer et de comprendre la production de trace écrite chez l'humain.

Après avoir fait une revue des différents types de modèles génératifs de l'écriture et de la trace humaine (au chapitre 2), nous proposons notre modèle oscillatoire de l'écriture basé sur les travaux d'Hollerbach [1981]. L'avantage de notre modèle, appelé POMH, est qu'il est symétrique et continu. Nous verrons qu'il est capable de rendre compte de plusieurs propriétés importantes de l'écriture et de sa production (au chapitre 3).

Il a été montré que l'approche dynamique de la motricité appliquée tant à la coordination bimanuelle qu'au tracé d'ellipse, permet de rendre compte de patrons de coordination préférentiels [Haken, Kelso, et Bunz, 1985; Athènes, Sallagoïty, Zanone, et Albaret, 2004; Sallagoïty, Athènes, Zanone, et Albaret, 2004; Danna, Athènes, et Zanone, 2011]. Nous appuyant sur ces travaux et les propriétés qui y sont mises en avant, nous tentons au chapitre 4, d'étudier la production de la trace écrite sous cet angle, en nous basant sur la vision oscillatoire de l'écriture telle que décrite par POMH. Nous nous intéressons ensuite au lien entre vitesse de l'écriture et géométrie de la trace. Nous proposons une hypothèse

tentant de rendre compte de la déformation subie par une trace en cas d'augmentation drastique de la vitesse de production.

En matière de reconnaissance de caractères, les modèles génératifs sont rarement utiles Plamondon [2000]. Pourtant, chez les humains il semble que la perception de l'écriture et sa production soient deux facettes d'un même processus [Longcamp, Hlushchuk, et Hari, 2011]. Au chapitre 5 nous tentons d'explorer cette voie.

1.2 Plan de lecture

Ce qui suit est un aperçu rapide du contenu de chacun des chapitres et annexes de cette thèse.

Chapitre 2, Modèles génératifs de l'écriture et du mouvement Ce chapitre présente plusieurs modèles génératifs de l'écriture. Les deux modèles les plus notables sont le modèle d'Edelman-Flash en section 2.1.3.2 et l'un des premiers modèles oscillatoires de l'écriture, le modèle d'Hollerbach (en section 2.2.1) qui est à la base de notre modèle POMH présenté dans le chapitre suivant.

Chapitre 3, POMH Nous présentons un modèle oscillatoire de l'écriture nommé POMH exhibant des propriétés intéressantes telles que la symétrie et la parcimonie. Les performances du modèle sont comparées sur des traces simples au modèle d'Edelman-Flash, l'un des modèles standards de la génération de l'écriture. Un algorithme appelé FHA, permettant l'extraction des paramètres de notre modèle depuis des traces enregistrées est ensuite présenté. Enfin, notre modèle est testé sur différentes traces dans diverses conditions afin de tester sa robustesse.

Chapitre 4, Dynamique de l'écriture Dans le domaine de la motricité, l'approche dynamique est un outil puissant, souvent utilisé au cours des dernières décennies. Cette approche a été appliquée avec succès sur le phénomène de la coordination bimanuelle, ou encore pour prédire le sens de tracé d'un segment en fonction de son orientation. En ce qui concerne la production de trace écrite, des travaux préliminaires ont été effectués sur des ellipses de différentes excentricités. Nous tentons ici d'appliquer et de vérifier ceux-ci sur des traces écrites en général afin de mieux comprendre cette production. Nous tentons en particulier, de relier la vitesse à la forme de la trace écrite, et prédire quelles seraient les déformations subies par une trace si elle était écrite plus rapidement.

Chapitre 5, Reconnaissance off-line de l'écriture manuscrite, un modèle interactiviste Après une présentation du cadre interactiviste, nous proposons une piste ayant menée à l'élaboration de deux algorithmes de reconnaissance off-line de l'écriture. Le premier utilise le modèle POMH afin de retrouver les paramètres qui produiraient une trace analogue. Le second est basé sur le concept d'image "temporalisée" qui sera détaillé alors. Ces algorithmes fonctionnent sur le principe de "simulation par l'action", qui permet d'analyser des traces bruitées et/ou difficiles à segmenter pour les algorithmes usuels.

Chapitre 6, Conclusion Ce chapitre résume les apports de ce travail de thèse et ouvre la voie à de futures recherches au travers de différentes pistes.

Annexe A, Outils logiciels de saisie et d'analyse de l'écriture Pour mener à bien les expériences présentées dans cette thèse, une suite logicielle de saisie et d'analyse de l'écriture à été développée. Composée de quatre programmes indépendants, elle permet entre-autre de gérer de manière automatique les séances de saisie de données de l'expérimentateur. Elle permet aussi au chercheur d'appliquer de manière intuitive des transformations aux signaux composants une trace écrite afin de comprendre les différentes interactions pouvant exister entre ceux-ci.

Annexe B, Oscillateurs, couplage et synchronisation Des groupes d'oscillateurs couplés exhibent des propriétés intéressantes. Nous présentons dans cette annexe un travail de simulation visant à étudier la synchronisation spontanée sur des populations d'oscillateurs couplés.

Annexe C, Compilateur MATLAB vers Java La suite d'outils réalisée dans le cadre de cette thèse pour l'étude de l'écriture est écrite en Java tandis que les algorithmes d'analyses de trace ont été initialement développés en MATLAB. Afin de simplifier le travail de traduction de MATLAB vers Java nous avons mis en place un traducteur automatique.

Annexe D, Simulateur d'environnement virtuel par crochetage d'API Cette annexe décrit une technique dite de "crochetage d'API", qui s'applique à la plupart des jeux vidéo sur PC, et qui permet de récupérer les images et d'agir en temps réel dans l'univers virtuel du jeu. Cela permet à des chercheurs de bénéficier à moindre coût d'un simulateur généralement très réaliste. C'est cette technique qui a été utilisée dans le simulateur de conduite basé sur des oscillations qui est abordé au chapitre 5.

Chapitre 2

Modèles de génération de l'écriture et du mouvement

Il peut être considéré qu'il y a deux grandes catégories de modèles de l'écriture. La première est constituée des modèles que l'on peut qualifier d'applicatifs. Ils cherchent à résoudre un problème concret (par exemple : reconnaître un scripteur, reconnaître une phrase ou encore imiter l'écriture de quelqu'un en préservant son style). Les modèles de cette catégorie sont souvent ad-hoc et ont pour but de traiter efficacement les problèmes qu'ils cherchent à résoudre.

La deuxième catégorie, que nous allons étudier dans ce chapitre sont les modèles dits génératifs. Ils ont pour but de comprendre la production de l'écriture. Nous inspirant de Gangadhar, Joseph, et Chakravarthy [2007] et Schomaker [1991], nous décomposons le processus de production de l'écriture en plusieurs niveaux (voir la figure 2.1).

Le scripteur commence par l'intention d'écrire un message : c'est le *niveau sémantique*. Ensuite, il transforme ce message en une suite de mots organisés sous une forme syntaxiquement correcte : c'est le *niveau lexical et syntaxique*. Puis le scripteur choisit les graphèmes utilisés pour former les mots (*niveau graphémique*), et pour chaque graphème sélectionne un allographe (variante possible d'un graphème, par exemple, un 'a' où un 'A') associé en fonction des contraintes syntaxiques et sémantiques (tenant compte des goûts du scripteur et soumises à un certain aléa). Enfin le niveau moteur traduit les allographes

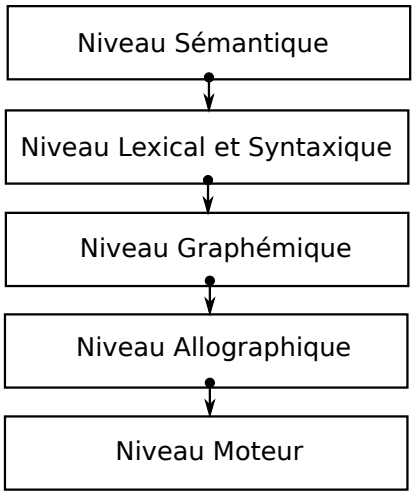


FIGURE 2.1 – Les cinq niveaux intervenant dans la formation de l'écriture.

en commandes de mouvement, puis en mouvement (en une bonne partie corps). Il est à noter qu'un graphème ne correspond pas forcément à une lettre, mais il peut être constitué de plusieurs lettres qui se suivent ou d'une partie de lettre.

Cette vision de la production de l'écriture, décomposée en étapes, est une vision simplifiée. En effet, il est fort probable que si ces niveaux correspondent à une certaine réalité vérifiable, il y a des interactions réciproques entre toutes ces étapes. A titre d'exemple, la coordination motrice entraîne des représentations allographiques différentes d'un individu à l'autre. Ces allographes ont probablement une influence sur le découpage graphémique des mots. Plus généralement chaque entité fonctionnelle de l'écriture est influencée par et a une influence sur toute entité fonctionnelle cognitive de l'individu. Ce type de découpage en entités fonctionnelles est en réalité mal adapté à la complexité des processus cognitifs.

Ce grossier découpage permet pourtant de nous simplifier la compréhension du processus d'écriture en nous permettant de nous focaliser sur l'une où l'autre des entités fonctionnelles. Cependant, lorsqu'on s'intéresse à l'étude d'un niveau fonctionnel en particulier, il ne faut pas négliger les influences extérieures sur le fonctionnement de celui-ci.

La suite de ce chapitre et le chapitre suivant s'intéressent plus particulièrement au *niveau moteur*, c'est à dire du passage des allographes à la trajectoire finale de l'écriture en passant par une représentation intermédiaire qui coderait cette trajectoire selon certains critères, en utilisant une suite de paramètres. Ce niveau a des liens étroits avec le problème plus général de la génération du mouvement et nous allons passer en revue ici quelques modèles de la production de traces écrites.

Ce chapitre sera divisé en deux sections. La première s'intéressera a un ensemble de modèles de l'écriture dans le but de donner une vision globale des modèles existants. La seconde partie, s'intéressera plus particulièrement aux modèles oscillatoires de l'écriture puisqu'ils sont à l'origine du modèle POMH qui sera présenté au chapitre suivant.

2.1 Modèles de production de l'écriture

2.1.1 Modélisation probabiliste

Dans son travail de thèse, Gilet [2009] a mis en place un modèle bayésien de l'écriture (appelé BAP) vue comme une boucle perception-action. Ce modèle tente de rendre compte tant de la reconnaissance que de la génération de caractères, pour différents scripteurs. La variabilité de la génération de la trace tant intra-scripteur qu'inter-scripteur y est étudiée. De fait, ce modèle permet aussi la reconnaissance du scripteur. La figure 2.2 montre l'architecture du modèle BAP avec comme éléments essentiels une représentation interne sensorielle et une représentation interne motrice.

Le réseau bayésien associé au modèle est basé sur plusieurs variables aléatoires. Tout d'abord, L correspond aux lettres et a valeur dans l'alphabet. W correspond aux scripteurs. Enfin, $C_L^{0:N}$ correspond à la séquence des points de contrôle représentant une lettre : les lettres sont représentées par une suite de points de contrôle situés en début et fin de trace, aux points de rebroussement et aux points de tangente verticale et horizontale. On retient les positions et les vitesses de ces lettres aux points de contrôle, $C_L^{0:N} = C_{L_x}^{0:N} C_{L_y}^{0:N} C_{L_{\dot{x}}}^{0:N} C_{L_{\dot{y}}}^{0:N}$. Chaque $C_{L_{x/y}}^{0:N}$ a valeur dans $[\![0, 40]\!]$ et chaque $C_{L_{\dot{x}/\dot{y}}}^{0:N}$ a valeur dans $[\![-4, 4]\!]$, après discrétisation des domaines de travail.

La figure 2.3 montre le modèle BAP complet. Dans ce dernier, la colonne motrice (au centre sur l'image) est dupliquée. Le duplicata (à droite sur l'image) représente la simulation interne du mouvement. Celle-ci peut être utilisée pour améliorer les résultats de la reconnaissance de caractères. Cette idée d'utiliser la commande motrice, et plus

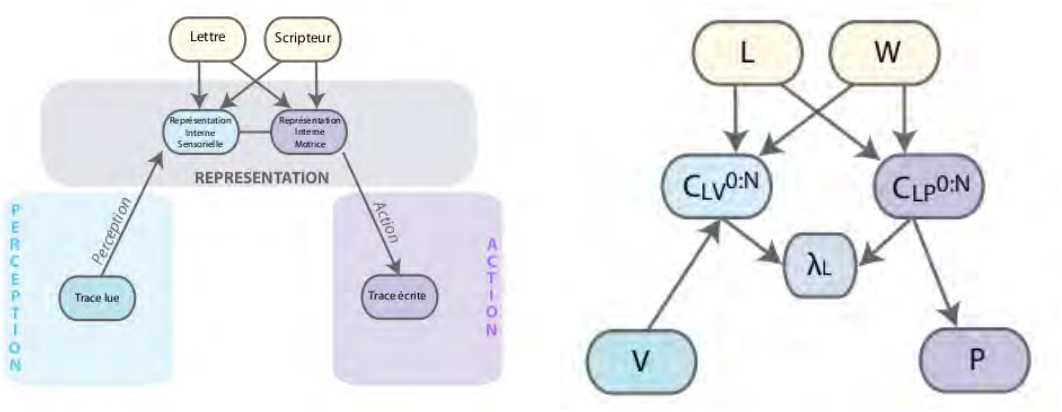


FIGURE 2.2 – Le modèle BAP s'appuie sur une représentation interne sensorielle et une représentation interne motrice des lettres. A droite, le réseau bayésien correspondant au modèle. Images empruntées à Gilet [2009].

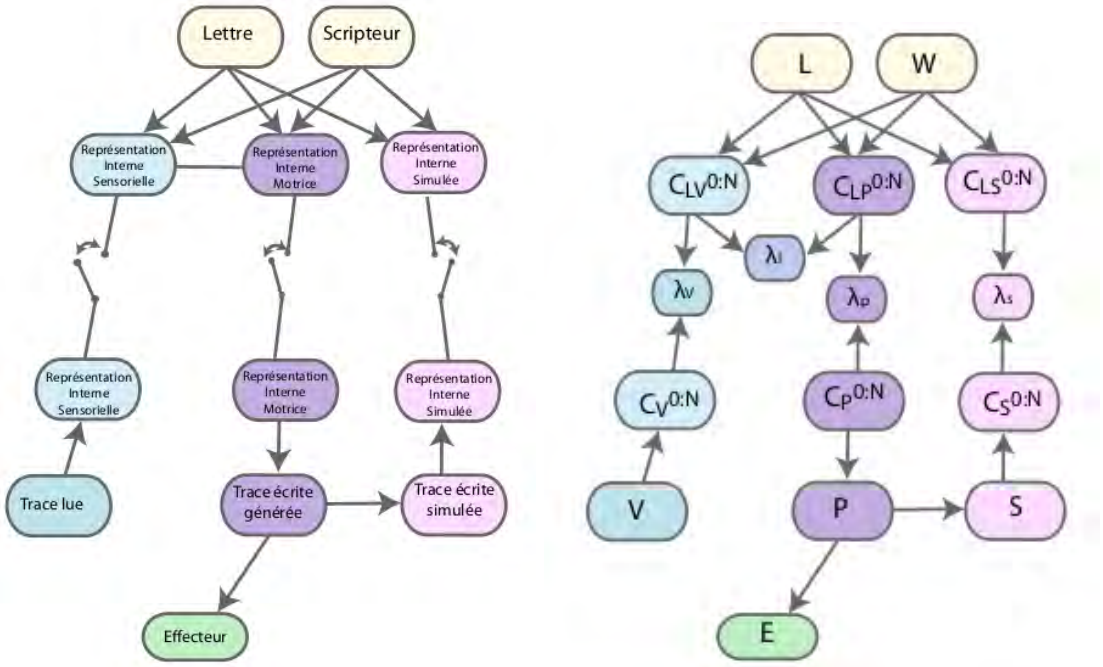


FIGURE 2.3 – Le modèle BAP complet. Images empruntées à Gilet [2009].

particulièrement une simulation interne de celle-ci pour améliorer la reconnaissance de l'écriture manuscrite est inspirée de nombreux travaux [Jeannerod, 2001; Proust, 2000; Longcamp, Tanskanen, et Hari, 2006; Knoblich et Seigerschmidt, 2002].

Une fois entraîné, il est possible de requêter ce modèle (comme il est possible de le faire pour tous les modèles bayésiens, voir Chapitre 14 et 20 de Russell et Norvig [2006]). Par exemple, la question "Quelle est la lettre correspondant à la trace perçue écrite par tel scripteur ?" est obtenue par $P(L|V_X^{0:M} = v_x^{0:M}, V_Y^{0:M} = v_y^{0:M}, W = w, \lambda_V = 1)$. Un autre exemple : "Quelle est la lettre correspondant à la trace perçue écrite par un scripteur non-connu ?", $P(L|V_X^{0:M} = v_x^{0:M}, V_Y^{0:M} = v_y^{0:M}, \lambda_V = 1)$. Enfin, il est possible de faire des lectures en utilisant la simulation motrice interne, ce qui dans certains cas améliore le taux de reconnaissance.

De ce modèle nous retiendrons deux aspects. Premièrement, le fait qu'il soit complet. En effet, il gère aussi bien la lecture que l'écriture des lettres. Deuxièmement, le fait qu'il utilise une simulation interne de l'écriture pour la reconnaissance. Cette propriété est intéressante car elle rend compte des observations neurobiologiques présentées dans les travaux cités précédemment. Nous avons tenté de reprendre ces deux aspects dans le travail présenté au chapitre 5.

2.1.2 Modélisation par réseaux de neurones

Plusieurs modèles de génération/reproduction de l'écriture sont basés sur des réseaux de neurones [Grossberg et Paine, 2000; Paine, Grossberg, et Van Gemmert, 2004; Kalveram, 1998; Schomaker, 1991; Gangadhar et al., 2007]. Nous allons en particulier nous intéresser au modèle de Gangadhar et al. [2007] car ce dernier est aussi un modèle oscillatoire.

Schomaker [1991] a identifié quatre étapes dans la génération d'une trace par un système basé sur un réseau de neurones :

1. Le système est d'abord initialisé, il est amené d'un état quelconque à un état initial permettant d'appliquer la séquence générant le mouvement.
2. Un signal déclenchant l'exécution de la séquence est attendu.
3. La séquence de production est exécutée.
4. Une phase de nettoyage, facultative, est ensuite exécutée. Elle a pour but de remettre le système dans un état standard.

Le réseau de Gangadhar et al. [2007] (voir figure 2.4) est composé de trois couches : la couche des entrées, la couche oscillatoire et la couche des sorties. Chaque nœud de la couche des entrées représente une trace différente (dans ce cas-ci, un allographe différent). Toutes les entrées sont à l'état bas quand le réseau est au repos. Une entrée est à l'état haut pendant la production d'une trace (un allographe) correspondant à cette entrée. La couche oscillatoire est composée de plusieurs sous-couches dans lesquelles des neurones oscillants sont connectés en anneaux et ont la même fréquence d'oscillation. La couche des sorties a deux sorties représentant la vitesse verticale et la vitesse horizontale de la trace produite. Le sous-réseau "horloge" ("timing network") contrôle les événements dans le réseau, permettant de produire la séquence des quatre étapes présentées plus haut.

Chaque neurone oscillant est décrit par :

$$\tau_x \dot{x} = -x + V - s + I \tag{2.1}$$

$$V = \tanh(\lambda x) \tag{2.2}$$

$$\tau_s \dot{s} = -s + V \tag{2.3}$$

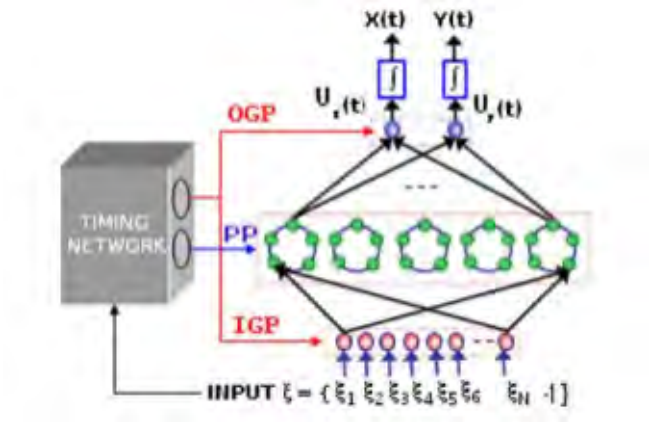


FIGURE 2.4 – Schéma du réseau de neurones utilisé par Gangadhar. Image empruntée à Gangadhar et al. [2007]

où V est la sortie oscillante, x et s des états internes du neurone : x excite s tandis que s inhibe x. Chaque anneau (constituant une sous-couche de la deuxième couche), composé de ces oscillateurs, reliés avec des paramètres bien choisis, finit par atteindre un cycle limite (stable), où chaque oscillateur est à une phase relative spécifique (relativement aux autres).

L'étape de préparation consiste à conduire le système de n'importe quel état à un état standard à partir duquel la production de la trace devient possible. Cela se fait en envoyant une décharge au premier neurone de chaque anneau et en attendant une durée prédéterminée : elle doit être suffisamment longue pour que chaque anneau s'approche suffisamment du cycle limite. Le moment de la décharge doit être effectué à une phase précise du cycle limite. Le sous-réseau "horloge" s'occupe de coordonner tous ces événements. Il s'occupe aussi d'activer les couches d'entrées et de sorties une fois l'état standard atteint.

Le réseau est entrainé par un algorithme de propagation inverse usuel [Haykin, 2007]. Lors de la phase d'apprentissage, il est amené à l'état standard avant que chaque trace à apprendre lui soit présentée. Une fois entrainé, le réseau est capable de produire les mouvements associés à chacun des allographes appris. Il permet aussi de produire des suites d'allographes attachés, en modifiant la séquence de contrôle envoyée par le réseau "horloge" (par exemple en enchaînant deux séquences). Cependant, il y a un délai de préparation entre chaque lettre, pour amener le réseau à son état standard, ayant pour conséquence l'arrêt du tracé pendant ce délai. Des solutions peuvent être apportées pour contourner ce problème, mais elles ont un impact négatif sur la géométrie de la trace. En effet, la pente de l'écriture est modifiée et la jonction entre deux lettres peut présenter une cassure. Des mécanismes sont alors mis en place pour corriger ces problèmes. Toutes ces procédures ne seront pas détaillées ici.

Ce modèle montre qu'une vision connexioniste de l'écriture n'est pas incompatible avec une vision oscillatoire de celle-ci. Cette propriété est compatible avec notre modèle de l'écriture qui sera présenté au chapitre 3 et conforte notre intuition que certains processus cognitifs, en particulier ceux liés à des tâches motrices comme l'écriture manuscrite, ont une nature fondamentalement oscillatoire.

2.1.3 Modélisation basée sur le contrôle optimal

Dans ce type de modèle, l'écriture est décrite par un système dynamique allant d'un état initial à un état but. La trajectoire générée est déterminée par un certain nombre de

contraintes. Le plus souvent, il s'agit de contraintes d'optimalité, qui tendent à ce que la trajectoire empruntée réduise une ou plusieurs fonctions de coût.

Après une brève présentation de la théorie du contrôle optimal, nous donnerons comme exemple de génération de l'écriture basée sur ce principe, le modèle d'Edelman et Flash [1987] qui explique comment passer d'un mot à une trajectoire grâce au contrôle optimal.

2.1.3.1 Contrôle optimal

Considérons un système dynamique S contrôlable dont l'état est décrit par $x \in \mathbb{R}^n$:

$$\dot{x}(t) = f(t, x(t), u(t)) \tag{2.4}$$

$u(t)$ étant une commande de l'espace des commandes admissibles U et f une fonction de $\mathbb{R}^+ \times \mathbb{R}^n \times U$ à valeur dans $\mathbb{R}^n$.

On se donne une cible x_c. Deux questions se posent alors :

- Cette cible est-elle atteignable ?
- Quelle commande dois-je appliquer au système pour atteindre cette cible, possiblement en respectant certains critères ?

On dit que S est contrôlable en temps fini si et seulement si :

$$\exists T \in \mathbb{R}^+, \forall x_c \in \mathbb{R}^n, \forall x_0 \in \mathbb{R}^n, \exists u \in U, \exists t_c < T, \mathcal{F}(x_o, x_c, u, S, t_c) \tag{2.5}$$

où $\mathcal{F}(x, u, S, t)$ est la fonction propositionnelle qui indique si S est à l'état x_c au temps t_c en ayant été soumis à une suite de commandes u depuis un état initial x.

Si S est contrôlable en temps fini, on peut alors définir un coût $C \in R$ à une trajectoire associée à la commande $u \in U$ sur l'intervalle $[0, T]$. Le problème de contrôle optimal consiste alors à minimiser C.

Principe du maximum de Pontryagin faible Soit C une fonction de coût définie sur $[0, T]$ telle que :

$$C = \int_0^T L(t, x(t), \dot{x}, ..., \overset{\cdot \times n}{x}(t)), \tag{2.6}$$

L étant le Lagrangien de S (décrivant la dépendance de C vis-à-vis des dérivées temporelles de x). On a alors une condition suffisante pour que C soit minimale :

$$\frac{\partial L}{\partial x} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) + ... + (-1)^n \frac{d^n}{dt^n}\left(\frac{\partial L}{\partial \overset{\cdot \times n}{x}}\right) = 0 \tag{2.7}$$

Ici on remarque que C ne dépend pas de u, ce qui signifie que u n'est pas directement soumise à des contraintes.

2.1.3.2 Le modèle d'Edelman-Flash

Pour Edelman et Flash [1987] l'écriture est une juxtaposition de morceaux de mouvements : on parle ainsi de modèle discret. L'écriture est une suite de courtes traces, qui peuvent être rangées en classes de formes, chaque classe étant appelée un prototype. Ces prototypes, au nombre de quatre, sont constitués du crochet, du U, du gamma et du cercle (voir figure 2.5). La lettre "a" écrite en script serait alors un ovale, un U à l'envers suivi d'un crochet. Il est à noter que dans l'écriture ces formes apparaissent le plus souvent après avoir subi des transformations (rotations, homothéties, ...).

FIGURE 2.5 – Les quatre prototypes constituant les bases de l'écriture selon Edelman et Flash. Image empruntée à Edelman et Flash [1987].

Une fois l'écriture partitionnée en suite de prototypes, chacun de ces morceaux est encodé selon le modèle de génération de mouvement de notre choix. Edelman et Flash [1987], donnent deux modèles de génération des prototypes s'appuyant sur la théorie du contrôle optimal et en font la comparaison : l'un est basé sur la minimisation de la dérivée troisième du mouvement (notée MJ_{1v}), l'autre sur la minimisation de sa dérivée quatrième (notée MS_1).

Minimum Jerk (minimum des secousses) MJ MJ cherche à minimiser les changements d'accélération. Reprenant le raisonnement de la section 2.1.3.1, la fonction coût est définie comme suit :

$$C = \int_0^T \dddot{x}^2(t) + \dddot{y}^2(t)dt \tag{2.8}$$

Minimiser C revient à minimiser séparément le terme en x et le terme en y. Plaçons nous dans le cas de x (pour y la démarche est identique). On cherche à minimiser :

$$C_x = \int_0^T \dddot{x}^2(t)dt \tag{2.9}$$

Le Lagrangien correspondant est donc $L_x = \dddot{x}^2$. La trajectoire optimale est alors un polynôme de degré 5 :

$$x(t) = \sum_{k=0}^{k=5} a_k t^k \tag{2.10}$$

Si on impose $x(0)$, $\dot{x}(0)$, $\ddot{x}(0)$, $x(T)$, $\dot{x}(T)$ et $\ddot{x}(T)$ alors x est entièrement déterminé.

Pour étendre le modèle aux mouvements courbes simples, on ajoute une contrainte supplémentaire en imposant au modèle de passer par un point P donné du plan (on note ce modèle étendu MJ_1). La position x est alors donnée par (en appliquant le principe du Maximum de Pontryagin [Bryson et Ho, 1975]) :

$$x(t) = \sum_{k=0}^{k=5} a_k t^k + p_x(t - t_1)_+^5 \tag{2.11}$$

avec $(t - t_1)_+$ égal à 0 si $t - t1$ est négatif. $t1$ est le temps de passage de la trajectoire au point P, il peut être obtenu par minimisation conjointe des a_k et de p_x. Il peut aussi être fourni comme contrainte supplémentaire. Sur les quatre prototypes, seul le crochet est suffisamment bien décrit par MJ_1. Pour étendre la compatibilité aux autres prototypes, une solution envisageable est d'augmenter le nombre de points de passage. Le problème de cette solution est qu'elle augmente le nombre de points nécessaires pour représenter la trajectoire. Une autre solution consiste à ajouter des contraintes supplémentaires aux

extrémités ou au point de passage. On peut par exemple choisir de rajouter des conditions de vitesse au point de passage p. La trajectoire du modèle MJ_{1v} est alors donnée par :

$$x(t) = \sum_{k=0}^{k=5} a_k t^k + p_{1x}(t - t_1)_+^4 + p_{2x}(t - t_1)_+^4 \tag{2.12}$$

et ses contraintes $x(0)$, $\dot{x}(0)$, $\ddot{x}(0)$, $x(T)$, $\dot{x}(T)$, $\ddot{x}(T)$, $x(t_1)$ et $\dot{x}(t_1)$.

Minimum Snap (MS) MS (étudié pour la première fois par Flash [1983]) est basé sur la minimisation de la dérivée quatrième de la trajectoire. Le même raisonnement que celui appliqué pour le *minimum-jerk* nous amène à remarquer que la trajectoire MS est décrite par deux polynômes de degré 7 (un selon chaque direction). Il y a donc 8 paramètres à fournir pour calculer les coefficients de chacun des polynômes. On peut fournir les conditions aux bornes jusqu'à la dérivé 3$^{\mathrm{e}}$. De même que pour MJ_1, on peut rajouter une contrainte de passage par un point. La position x est alors décrite par :

$$x(t) = \sum_{k=0}^{k=7} a_k t^k + p_x(t - t_1)_+^7 \tag{2.13}$$

On utilise alors la coordonnée de x en t_1 comme paramètre supplémentaire.

Extraction de paramètres et comparaison entre trace reconstruite et trace réelle Edelman et Flash ont donné une méthode permettant de calculer les paramètres de leurs modèles (MS_1 et MJ_{1v}) à partir de traces prototypiques exécutées par des humains [Edelman et Flash, 1987]. Pour obtenir ces paramètres (les coefficients des polynômes), il faut extraire des traces enregistrées les conditions aux bornes (point de passage, positions initiales et finales, vitesses initiales et finales ...). Dans les deux cas, le point de passage était désigné par l'expérimentateur. Dans le cas de MS_1, les pentes initiales et finales utilisées pour calculer $\dddot{x}$ et $\dddot{y}$ sont fournies manuellement.

Après reconstruction, ils ont comparé les traces originales et les traces reconstruites (ainsi que leurs dérivées) avec leur modèle au moyen d'un coefficient de corrélation. Les deux modèles donnent des résultats similaires assez convaincants (un peu meilleurs pour MS_1, au prix d'un effort supplémentaire de l'expérimentateur).

Pour évaluer quantitativement notre modèle POMH, nous l'avons comparé avec le modèle MJ_{1v} sur chacune des traces prototypiques. Les résultats de cette comparaison sont présentés au chapitre 3.

2.1.4 Autres modèles discrets

Dans le modèle présenté précédemment, l'écriture (ou plus généralement le mouvement humain) est vu comme une juxtaposition de trajectoires élémentaires. Ce concept est largement repandu et prend sa source avec Lashley [1917] et est repris par Eden [1962] et van der Gon, Denier, Thuring, et Strackee [1962] dans le cadre de l'écriture. Les trajectoires complexes seraient ainsi une succession de tels mouvements (dits unitaires ou balistiques). L'idée de ne plus seulement juxtaposer ces mouvements unitaires mais de les superposer a été mise en avant par Morasso et Ivaldi [1982].

Nombreux sont les modèles qui s'appuient sur ces principes. Deux d'entre eux seront abordés ici : le modèle de Wada et Kawato [1995] qui représente une trace écrite par une suite de mouvements balistiques juxtaposés et le modèle de Plamondon et Guerfali [1998] où il est fait usage de la superposition de mouvements unitaires.

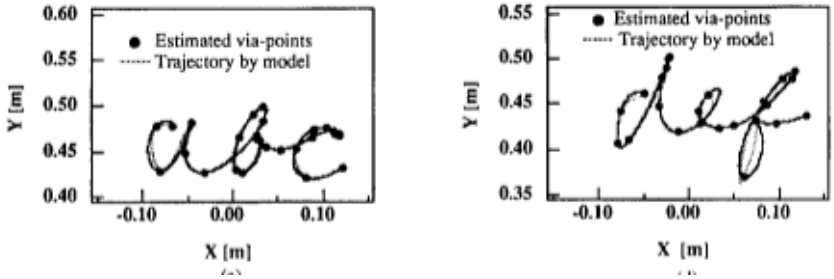


FIGURE 2.6 – Exemple de trace reconstruite par le modèle de Wada. Image empruntée à Wada et Kawato [1995].

2.1.4.1 Le modèle de Wada

Dans le modèle de Wada et Kawato [1995] l'écriture est vue comme une suite de trajectoires passant par des points de passage. Ces morceaux de trajectoire entre deux points de passage sont générés suivant la condition d'optimalité suivante : minimum de changement de couple (*minimum torque-change*) [Uno, Kawato, et Suzuki, 1989; Nakano, Imamizu, Osu, Uno, Gomi, Yoshioka, et Kawato, 1999; Kawato, Maeda, Uno, et Suzuki, 1990]. Le calcul du minimum de changement de couple dépend de l'effecteur puisqu'il s'agit de minimiser les mouvements autour de points de pivot modélisant les articulations du bras.

Pour reproduire une trace écrite et trouver les paramètres du modèle (la suite des positions datées des points de passage), un algorithme d'optimisation appelé FIRM est mis en œuvre. Il s'agit de chercher le nombre minimal de points de passage permettant de reproduire la trace grâce au minimum torque-change pour une erreur (entre la trace originale et la trace reconstruite) donnée. La figure 2.6 donne un exemple de reconstruction de deux mots par le modèle de Wada. On remarque que les points de passage extraits par l'algorithme d'optimisation sont répartis de manière irrégulière sur la trace.

2.1.4.2 Le modèle de Plamondon

Le modèle de Plamondon s'appuie sur la modélisation de mouvements balistiques basés sur le delta-lognormal, issu de la théorie cinématique du mouvement, qui tente de capturer les propriétés globales du réseau neuro-musculaire impliqué dans le mouvement [Plamondon et Guerfali, 1993; Plamondon, 1993, 1998; Plamondon, Feng, et Woch, 2003]. Les modèles basés sur ces principes semblent reproduire le plus efficacement les données cinématiques d'un mouvement rapide [Plamondon, Alimi, Yergeau, et Leclerc, 1993].

Dans ce modèle, une trace simple exécutée depuis une position arbitraire P_0 est caractérisée par 9 paramètres, C_0 et θ_0 représentant les propriétés géométriques de l'ensemble des articulations et des muscles impliqués dans le mouvement. Les paramètres D_1, D_2 et t_0 décrivent la commande. Les paramètres μ_1, μ_2, σ_1 et σ_2 représentent quant à eux les propriétés temporelles du réseau neuro-musculaire impliqué dans la génération du mouvement. L'amplitude de la vitesse de la trace est alors donnée par

$$|v(t)| = |D_{1_{(P_0,\theta_0,C_0)}}\Delta(t, t_0, \mu_1, \sigma_1^2) - D_{2_{(P_0,\theta_0,C_0)}}\Delta(t, t_0, \mu_2, \sigma_2^2)|, \tag{2.14}$$

où

$$\Delta(t, t_0, \mu_j, \sigma_j^2) = \frac{1}{\sigma_j\sqrt{2\pi}(t - t_0)} \exp\left(\frac{-(\ln(t - t_0) - \mu_j)^2}{2\sigma_j^2}\right) \tag{2.15}$$

et la direction de la vitesse $\angle v(t)$, selon une référence arbitraire, est donnée par

$$\angle v(t) = \theta_0 + C_0 \int_{t_0}^{t} |c(\tau)| d\tau. \tag{2.16}$$

Une trace écrite plus complexe peut ensuite être exprimée comme la superposition de mouvements simples représentés par l'équation précédente :

$$v(t) = \sum_{i=1}^{n} v_i(t - t_{0_i}). \tag{2.17}$$

La trace écrite est alors vue comme une suite de groupes de 9 paramètres représentant chacun un des mouvements unitaires constituant la trace complète.

Pour calculer cette suite de groupes de paramètres pour une trace donnée, l'algorithme suivant est appliqué [Guerfali et Plamondon, 1998] :

- calculer l'amplitude et la direction de la vitesse curviligne de la trace en utilisant les $X(t)$ et $Y(t)$ fournis par la tablette graphique lors de la saisie de la trace,
- utiliser les données de contact fournies par la tablette graphique pour séparer le mot en plusieurs traces,
- pour chacune de ces composantes, extraire les mouvements unitaires, cachés, par ajustement de la vitesse curviligne vue comme une somme de termes en delta-lognormal (cf. équation 2.17) décalés dans le temps, et par ajustement de la vitesse angulaire qui en résulte,
- sauver les neuf paramètres qui représentent le mieux chacune des composantes.

La figure 2.7 donne un aperçu de la reconstruction d'une trace originale 'elle' par le modèle de Plamondon. Enfin, la figure 2.8 montre comment le modèle représente une trace écrite par un ensemble de mouvements unitaires superposés, décalés dans le temps.

2.2 Modélisation oscillatoire de l'écriture

Eden [1962] a suggéré une conception sinusoïdale de l'écriture. Cette vision fut reprise par Hollerbach [1981], qui a exploré plus en détail cette voie en proposant un modèle oscillatoire de l'écriture ainsi qu'un modèle masse-ressort du bras lui permettant d'asseoir cette vision oscillatoire. Il pensait que les tentatives de modélisation de l'écriture effectuées jusqu'alors (voir [Yasuhara, 1974; Denier et Thuring, 1965; MacDonald, 1966; Mermelstein et Eden, 1964]) ne consistant qu'à faire de l'ajustement de courbe sur des profils d'accélération issus d'expérimentations ne permettaient pas de comprendre comment le système moteur générait l'écriture. Au delà de l'écriture, Hollerbach était intéressé par la modélisation biologiquement plausible du mouvement animal et humain : "Because of the speed of fast cursive writing, handwriting trajectories are open-loop and hence are relatively pure manifestation of central programming" [Hollerbach, 1981]. L'utilisation d'oscillateurs dans la modélisation du mouvement a depuis été utilisée avec succès en biologie avec l'introduction du concept des Central Pattern Generators (CPG) [Delcomyn, 1980; Ijspeert, 2008] et dans le domaine de la locomotion en robotique [Arena, 2000; Hoinville, Henaff, et Delaplace, 2007; Crespi et Ijspeert, 2006].

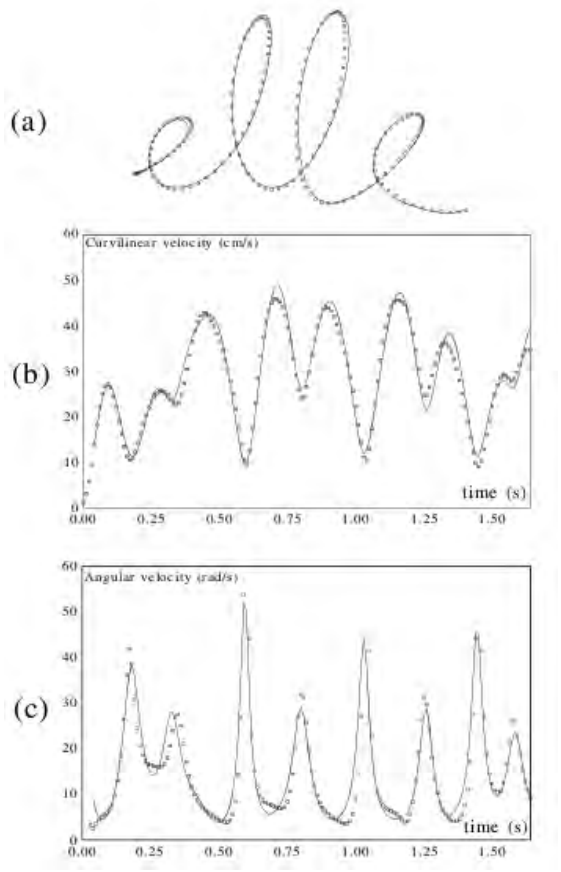


FIGURE 2.7 – Exemple de trace reconstruite par le modèle de Plamondon, la vitesse curviligne et la vitesse angulaire correspondante. En continu, l'original et en pointillé, la reconstruction. Image empruntée à Plamondon et Guerfali [1998].

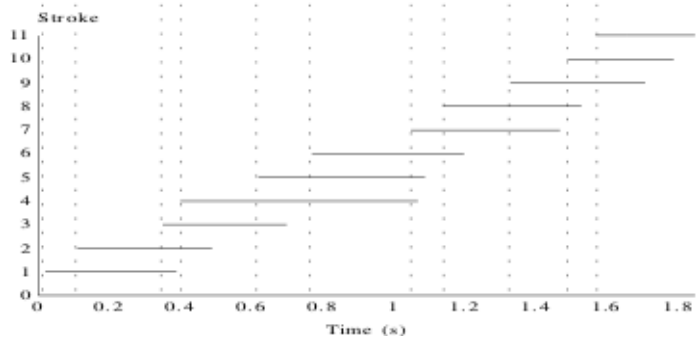


FIGURE 2.8 – Exemple de décomposition d'une trace écrite en superposition de traces simples par le modèle de Plamondon. Image empruntée à Plamondon et Guerfali [1998].

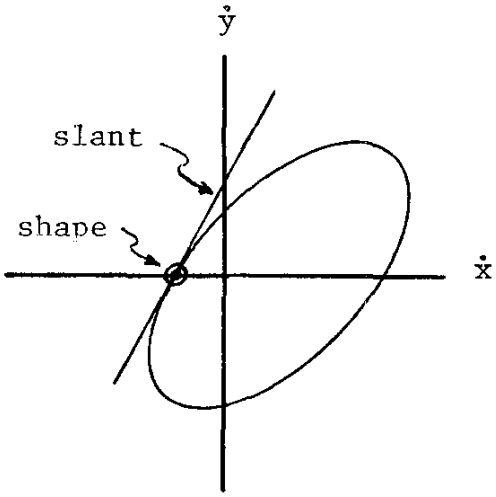


Figure 2.9 – Représentation visuelle des équations d'Hollerbach, son excentricité ainsi que son orientation dépendent de a, b et $\phi = \phi_x - \phi_y$ dans le cas où les oscillateurs ont la même fréquence. Image empruntée à Hollerbach [1981].

2.2.1 Modèle d'Hollerbach

Selon Hollerbach [1981], l'écriture peut être vue comme la combinaison de deux oscillateurs dans le plan, l'un oscillant selon la direction verticale, l'autre selon la direction horizontale. À cela s'ajoute un mouvement constant vers la droite :

$$\begin{aligned} \dot{x} &= a\sin(\omega_x t + \phi_x) + c \\ \dot{y} &= b\sin(\omega_y t + \phi_y) \end{aligned} \tag{2.18}$$

où a (resp. b) est l'amplitude de la vitesse horizontale (resp. verticale), ω_x, ω_y, ϕ_x et ϕ_y sont respectivement les fréquences et les phases associées à ces directions. t représente le temps et c l'amplitude du déplacement constant vers la droite ($c > 0$).

Afin de mieux appréhender les différents paramètres du modèle, il peut être intéressant de regarder le tracé de ces équations dans l'espace des vitesses [Abelson, di Sessa, et Rudolph, 1975]. La figure 2.9 montre le résultat des équations 2.18 à paramètres constants dans l'espace des vitesses. Elles prennent la forme d'une ellipse dont l'excentricité et l'inclinaison dépendent des paramètres a, b et $\phi = \phi_x - \phi_y$. ϕ correspond à la phase relative entre les deux oscillateurs ; elle fait sens si ω_x et ω_y sont très proches. Le sens de parcours de l'ellipse est déterminé par le signe de ϕ et la vitesse de parcours par ω (en supposant $\omega = \omega_x = \omega_y$).

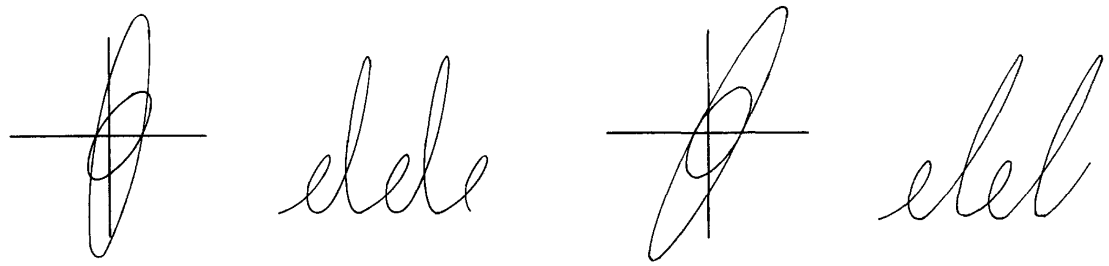
Les paramètres du modèle (i.e. a, b, ω_x, ω_y, ϕ_x et ϕ_y) sont constants par morceaux : il changent au moment où la vitesse verticale s'annule. Hollerbach a fait ce choix car il a remarqué que la forme des lettres semblait déterminée par la valeur de la vitesse horizontale au point d'annulation de la vitesse verticale, exprimée par :

$$\Psi = \dot{x}(t_{y_o}) = c - a\sin\phi. \tag{2.19}$$

Quand Ψ est proche de zéro, on obtient des formes aiguës en haut de lettres comme "u" ou "i". Quand Ψ est négatif, on obtient des boucles telles que dans "e". Enfin quand Ψ est positif, on obtient des formes arrondies telles que pour "n" ou "m". Pour quelques exemples, voir la figure 2.10. Ψ correspond au point "shape" sur la figure 2.9.

(a) $\Psi = 1.2$ (b) $\Psi = -43$ (c) $\Psi = 37$

FIGURE 2.10 – Dans le modèle d'Hollerbach, la forme des lettres dépend de l'amplitude et du signe de Ψ.



(a) Seul a a été modifié. Les "l" sont bien plus grands mais leur pente n'est pas homogène avec celle des "e".

(b) Ici, b à été corrigé afin de rétablir la pente tout en conservant la forme de la lettre.

FIGURE 2.11 – Modulation de l'amplitude : changer l'amplitude uniquement change la pente de l'écriture. Pour conserver la pente et la forme des lettres il convient de corriger b. Dans les deux cas le point d'annulation de la vitesse verticale est placé au même endroit pour les deux ellipses de l'espace des vitesses (dans les deux cas Ψ est identique) : la forme de la lettre est préservée. Figures empruntées à Hollerbach [1981].

Par ailleurs il est possible de moduler la taille des lettres par un changement des amplitudes a et b (voir figure 2.11a). Un changement d'amplitude seul entraîne un changement de pente. La pente β est donnée par l'équation :

$$\tan\beta = \frac{b}{a\cos\phi} \text{ where } \phi = \phi_x - \phi_y. \tag{2.20}$$

Comme on peut le voir la pente dépend de ϕ, a et b ; ainsi, pour la préserver deux paramètres au moins doivent être modifiés, la variation du second compensant la variation du premier.

Supposons que l'on veuille maintenant modifier la taille de la lettre en conservant sa pente et sa forme, c'est à dire que Ψ et β doivent rester inchangés (contraintes k_1 et k_2). On peut exprimer ces contraintes sous la forme :

$$\begin{aligned} k_1 &= c - a\sin\phi \\ k_2 &= \frac{b}{a\cos\phi} \end{aligned} \tag{2.21}$$

Ces deux contraintes imposent b :

$$b = k_2(c - k_1)\cot\phi \tag{2.22}$$

La figure 2.11b montre le résultat sur l'exemple "elel". On notera que les tangentes aux points d'annulation de la vitesse verticale des deux ellipses sont identiques contrairement au cas précédent (figure 2.11a).

2.2.2 Modèles basés sur le modèle d'Hollerbach

Les travaux d'Hollerbach permirent à d'autres d'aller plus loin dans cette voie. Singer et Tishby proposèrent un encodage discret des paramètres d'Hollerbach [Singer et Tishby, 1994]. Stettiner et Chazan [1994] suivis par Chen, Agazzi, et Suen [1997] tentèrent quant-à-eux d'améliorer le modèle d'Hollerbach en essayant de contourner ce qui leur semblait être une de ses insuffisances majeures : la dépendance des paramètres du modèle aux points d'annulation des profils de vitesse.

2.2.2.1 Encodage dynamique de l'écriture manuscrite

En 1994, Singer et Tishby [1994] proposèrent un moyen d'encoder l'écriture de manière discrète. Leur encodage permet une réduction par 100 fois du nombre de bits utilisés par rapport à l'encodage initial (positions échantillonnées à temps constant).

Nous allons décrire les étape présentées par Singer et Tishby permettant de passer de l'écriture à un encodage discret de l'écriture. Ils s'appuient sur les équations suivantes :

$$\begin{aligned} \dot{x} &= a_x(t)\sin(\omega_x(t)t + \phi(t)) + c \\ \dot{y} &= b_y(t)\sin(\omega_y(t)t) \end{aligned} \tag{2.23}$$

La principale différence, par rapport aux equations 2.18, est l'absence de phase en y. ϕ représente donc la différence de phase entre les deux oscillateurs.

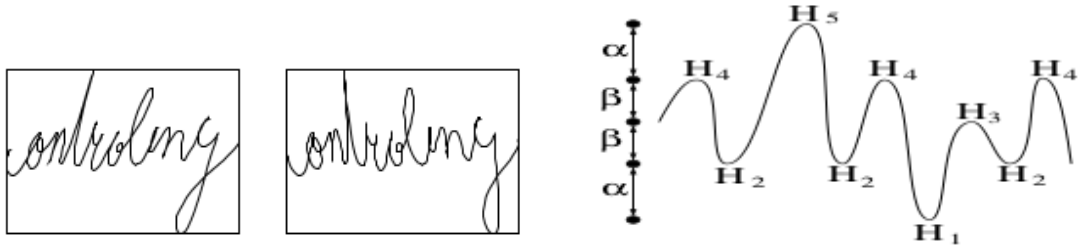
Correction de l'orientation de l'écriture L'écriture étant échantillonnée sans contraindre le scripteur, il se peut que l'écriture ne soit pas droite. L'angle α de cette dérive, permettant par rotation de corriger la pente de l'écriture, est évaluée par une méthode d'estimation statistique [Wald, 1940]. L'ensemble des points x_i et y_i du signal est divisé aléatoirement en ensembles de paires $\{(x_{2i}, y_{2i}), (x_{2i+1}, y_{2i+1})\}$. L'estimateur de $\tan\alpha$, $\hat{W}$ satisfait alors $\hat{W} = \frac{\sum_k y_{2k+1} - y_{2k}}{\sum_k x_{2k+1} - x_{2k}}$.

Compensation de pente Ensuite il faut corriger l'inclinaison de l'écriture. Si on note $\tilde{\dot{x}}$ la vitesse en x après correction de pente, on peut exprimer la vitesse horizontale sous cette forme : $\dot{x} = \tilde{\dot{x}} + A(t)\dot{y}$. Si on suppose A constant alors on peut estimer A par $\hat{A} = \frac{\sum_{t=1}^{N} \dot{x}\dot{y}}{\sum_{t=1}^{N} \dot{y}\dot{y}}$, où N est le nombre de points échantillonnés. Malheureusement, pour certains scripteurs, cette hypothèse n'est pas vérifiée, il faut donc estimer la pente localement comme ceci : $\hat{A}(t0) = \frac{\sum_{t=1}^{N} \dot{x}\dot{y}H(t_0 - t)}{\sum_{t=1}^{N} \dot{y}\dot{y}H(t_0 - t)}$ où H est une fenêtre de Hanning de largeur 5 cycles de $\dot{y}$. Finalement, on obtient $\tilde{\dot{x}}$ par $\tilde{\dot{x}}(t) = \dot{x}(t) - \hat{A}(t)\dot{y}(t)$. La figure 2.12a montre un exemple de ce procédé.

Estimation de c Supposé constant par Hollerbach, c est estimé ici par $\hat{c} = \frac{1}{N}\sum_{i=1}^{N} \dot{x}(i)$. L'hypothèse de la constance de c peut être vérifiée en estimant c sur une fenêtre glissante. De plus, $\dot{x}$ et $\dot{y}$ sont divisés par $\hat{c}$ à des fins de normalisation.

Estimation de a_x, a_x, ω_x et ω_y Les paramètres sont constants entre deux points consécutifs d'annulation de la vitesse. Les vitesses angulaires ω_x et ω_y sont calculées pour correspondre au temps entre deux zéros de vitesse consécutifs.

Notons $t_i^{x/y}$ les i^{e} zéros de la vitesse horizontale/verticale, et $L_i^{x/y}$ la progression horizontale/verticale après soustraction du décalage constant. Les amplitudes sont estimées



(a) Effet de la compensation de pente sur l'écriture.

(b) Exemple des positions verticales en fonction du temps : elles semblent ajustées autour de 5 niveaux.

FIGURE 2.12 – Images empruntées à Singer et Tishby [1994]

par :

$$\int_{t_i^{x/y}}^{t_{i+1}^{x/y}} \hat{a_{x/y}} sin\left(\frac{\pi}{t_{i+1}^{x/y} - t_i^{x/y}}(t - t_i^{x/y})\right) dt = L_i^{x/y} \Rightarrow \hat{a_{x/y}} = \frac{L_i^{x/y}\pi}{2(t_{i+1}^{x/y} - t_i^{x/y})} \tag{2.24}$$

Discrétisation des amplitudes Tout d'abord, les amplitudes horizontales et verticales sont supposées indépendantes.

En ce qui concerne la vitesse verticale, on remarque que les amplitudes sont quantifiées selon 5 niveaux (voir figure 2.12b) satisfaisant certaines contraintes de symétrie (voir Singer et Tishby [1994] pour plus de détails). On note H_1,...,H_5 ces niveaux et $L(t)$ les niveaux réellement observés. Par ailleurs, on suppose que les niveaux réels sont distribués selon une loi normale centrée en ces niveaux et de variance σ (identique pour tous les niveaux) : $L(t) = H_{I_t} + \xi$, $\xi \sim N(0, \sigma)$. I_t est une indicatrice qui au t^e zéro associe l'index du niveau (1, 2, 3, 4 ou 5). On cherche donc à estimer $\{H_i, \sigma\}$ avec les données d'observations qui sont incomplètes (on ne connait pas les I_t). Nous sommes dans le cas d'une situation d'estimation par maximum de vraisemblance avec données incomplètes. Cela se résout par l'algorithme d'espérance-maximisation [Dempster, Laird, et Rubin, 1977].

Dans le cas de la vitesse horizontale, selon les auteurs, il y a trois types de lettres, fines (i), normales (n) et grosses (o). Cependant il nous semble difficile de différencier i et n, puisqu'une série de "i" symétrisée par rapport à l'axe des x donne une série de "n".

Discrétisation de la phase et de la vitesse angulaire On peut utiliser la phase $\theta = \int_0^t \omega(t)dt$ pour reparamétriser les équations 2.23 :

$$\begin{aligned} \frac{dx}{d\theta} &= a_x(\theta)\sin(\omega(\theta + \phi(\theta)) + c\left[\frac{dt}{d\theta}\right] \\ \frac{dx}{d\theta} &= b_y(t)\sin(\theta) \end{aligned} \tag{2.25}$$

en supposant $\omega_x \approx \omega_y \triangleq \omega$.

Singer et Tishby ont remarqué que la phase relative ϕ prenait valeur dans $\{\pm\frac{\pi}{2}, 0\}$. Par ailleurs, ils ont remarqué que l'on pouvait supposer ω constant. Finalement en corrigeant les amplitudes on obtient une paramétrisation discrète de l'écriture : $\phi(\theta) \in \{-\frac{\pi}{2}, 0, \frac{\pi}{2}\}$, $a_x(\theta) \in \{a_x^1, a_x^2, a_x^3\}$ et $a_y(\theta) \in \{a_y^1, a_y^2, a_y^3, a_y^4, a_y^5\}$.

2.2.2.2 Modèle oscillatoire à paramètres non constants par morceaux

Selon Stettiner et Chazan [1994], l'approche de Singer et Tishby, utilisant les points de vitesse nulle comme points de changement de paramètres, est problématique pour modéliser statistiquement les paramètres car une même lettre a plusieurs représentations possibles (en particulier en ce qui concerne l'ordonnancement des zéros de vitesses).

Dans leur modèle, s'inspirant de Plamondon et Lamarche [1986], offrant une vision systémique, les vitesses $\hat{V}$ sont définies comme suit :

$$\hat{V}_{x/y}(t) = \frac{A}{\sqrt{1-\zeta(t)^2}} \sin\left(\omega(t)\sqrt{1-\zeta(t)^2}t + \phi\right) \exp\left(\zeta(t)\omega(t)\right) \tag{2.26}$$

où A est le facteur de gain, ϕ une phase relative fixe, ζ est le coefficient d'amortissement et ω la fréquence propre du système. Tous les paramètres sont continus et changent à intervalles réguliers. Dans ces intervalles, les paramètres sont supposés varier linéairement. Les paramètres sont donc des fonctions affines par morceaux.

Leur but étant de modéliser statistiquement les paramètres, ils ont besoin d'extraire les paramètres de traces réelles. A partir de ce modèle ils définissent le problème d'optimisation non-linéaire suivant. On définit P le nombre d'intervalles sur lesquels sont basés les paramètres, θ la suite des paramètres est définie par

$$\theta = (A, \phi, (\zeta(p), \omega(p)), p = 1...P)\,, \tag{2.27}$$

et le modèle du profil de vitesse :

$$V = \hat{V} + v \tag{2.28}$$

où v est le vecteur erreur. La trace est alors modélisée par :

$$S = \int V = \int \left(\hat{V} + v\right) = \tilde{V}(\theta) + \tilde{v}. \tag{2.29}$$

On cherche alors θ au sens des moindres carrés :

$$\hat{\theta} = \underset{\theta}{Min}(v^T v) \tag{2.30}$$

Ce problème peut être traité avec la Méthode de Newton Modifiée (Modified Newton Method, MNM, présentée par Luenberger et Ye [2008]).

Les auteurs proposent ensuite d'utiliser les paramètres extraits par le modèle afin de créer un modèle statistique de l'écriture. Les lettres sont représentées par des distributions à densités mélangées de gaussiennes (Multivariate Mixture Gaussian). Cette représentation leur a permis de faire de la reconnaissance de caractères pour : pour 4 scripteurs et 26 lettres, 99.4% de réussite sur la reconnaissance de caractère et 100% de réussite sur la reconnaissance du scripteur.

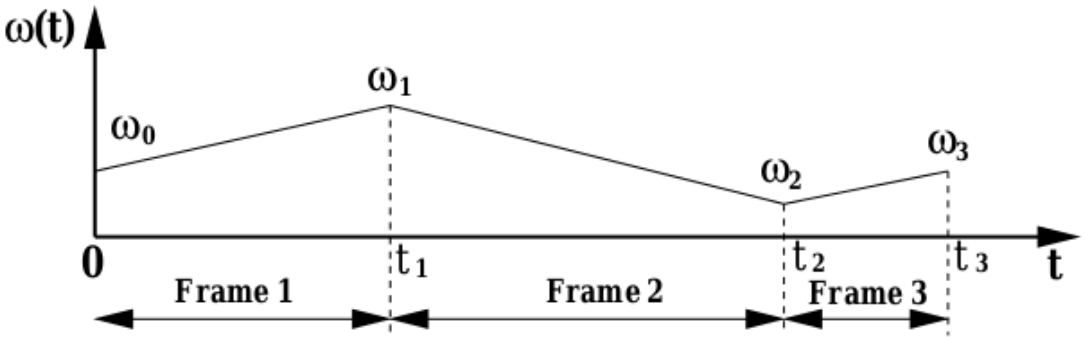
Allant encore plus loin dans la relaxation des contraintes imposées sur les paramètres de l'écriture et cherchant à réduire la quantité d'information nécessaire au codage de l'écriture, Chen et al. [1997] ont proposé d'utiliser des intervalles à taille variable plutôt qu'à taille constante comme intervalles associés aux paramètres, affines sur ces intervalles (un exemple est montré en figure 2.13a). Selon eux, dans le modèle de Stettiner et Chazan [1994], le fait que la taille des intervalles doive être déterminée de manière empirique (non systématique) est problématique. De plus, il n'y a aucune raison que la taille des segments de l'écriture, ainsi que les instants de changements abruptes de variation de paramètres, soient nécessairement répartis de manière uniforme.

Reprenant les notations précédentes, on note le signal d'erreur (équivalent à v) :

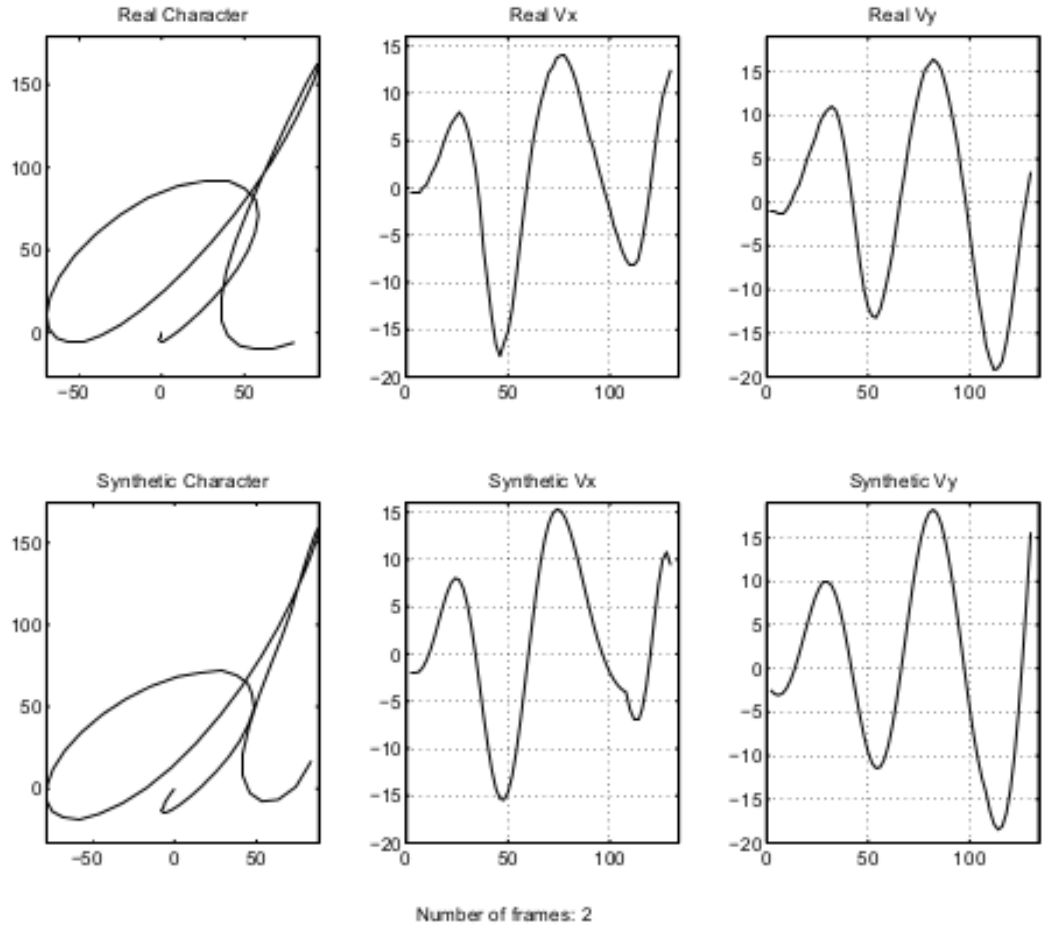
$$E(t) = V(t) - \hat{V}(t). \tag{2.31}$$

Dans ce modèle, la taille de chaque intervalle est déterminée algorithmiquement (par un algorithme de recherche binaire) en même temps que l'estimation des paramètres. Au début un petit bout de trajectoire est utilisé comme intervalle initial. Son erreur $E(t)$ est calculée. Si cette erreur est inférieure à un seuil s, la taille de l'intervalle est augmentée et l'erreur est de nouveau calculée. Si l'erreur $E(t)$ est plus petite que s, la taille de l'intervalle est réduite jusqu'à ce que l'on trouve t_0 tel que $E(t_0) \leq s$ et $E(t_0+1) > s$. La taille du premier intervalle est alors déterminée, et on recommence avec un nouvel intervalle sur la suite de la trajectoire.

Pour accélérer ce processus d'optimisation (ie. le calcul de E(t)), basé sur la méthode de Powell [Powell, 1962; Fletcher et Powell, 1963], il est supposé que l'espace des paramètres est borné. La figure 2.13b montre un exemple de trace reconstruite en utilisant les paramètres extraits par l'algorithme précédent : seuls deux intervalles sont nécessaires pour modéliser un "d".



(a) La fonction ω exprimée en fonction du temps est affine par morceaux. Les intervalles sont de tailles variables et déterminés alogrithmiquement.



(b) Comparaison entre un "d" original et un "d" reconstruit en utilisant les paramètres extraits par l'algorithme : seuls deux intervalles sont nécessaires.

Figure 2.13 – Figures empruntées à Chen et al. [1997]

Chapitre 3

POMH

3.1 Le modèle POMH

Contrairement au modèle d'Hollerbach, le modèle POMH est symétrique. En effet, comme nous l'avons vu, dans le modèle d'Hollerbach, les changements de paramètres, supposés constants par morceaux, interviennent tous aux moments d'annulation de la vitesse verticale. A l'inverse, pour POMH, les paramètre liés à l'oscillation verticale (resp. horizontale) changent aux moments d'annulation de la vitesse verticale (resp. horizontale). Cette symétrie permet, entre autre, de fournir une méthode heuristique, directe et peu coûteuse, d'extraction de paramètres depuis des traces réelles.

3.1.1 Équations cinématiques

On se place dans un repère canonique du plan. Comme dans le modèle d'Hollerbach, l'évolution de chaque coordonnée est décrite par un oscillateur. Les mouvements en x et en y sont exprimés comme suit :

$$\dot{x} = a_x(t)\sin(\omega_x(t)t + \phi_x(t)) \tag{3.1}$$
$$\dot{y} = a_y(t)\sin(\omega_y(t)t + \phi_y(t)) \tag{3.2}$$

où a_x, ω_x, ϕ_x, a_y, ω_y, ϕ_y sont des fonctions (positives dans le cas de a_x et a_y) constantes par morceaux. Les changements dans ces fonctions interviennent aux moments d'annulation de la vitesse horizontale (resp. verticale) pour a_x, ω_x et ϕ_x (resp. a_y, ω_y et ϕ_y).

Notons $t_{x,0}, \ldots, t_{x,N_x}$ les moments d'annulation de la vitesse en x et $t_{y,0}, \ldots, t_{y,N_y}$ les moments d'annulation de la vitesse en y. Le mouvement est alors entièrement spécifié par les équations (3.1) et (3.2) et les séries suivantes :

$$(t_{x,i}, a_{x,i}, \omega_{x,i}, \phi_{x,i}) \qquad 0 \leq i \leq N_x \tag{3.3}$$
$$(t_{y,i}, a_{y,i}, \omega_{y,i}, \phi_{y,i}) \qquad 0 \leq i \leq N_y \tag{3.4}$$

La figure 3.1 montre l'exemple d'une trace "umbrella" rendue par le modèle.

3.1.2 Extraction de paramètres

POMH permet l'extraction de paramètres à partir de traces enregistrées. L'information contenue dans ces traces peut alors être approchée par deux séries de type (3.3) et (3.4).

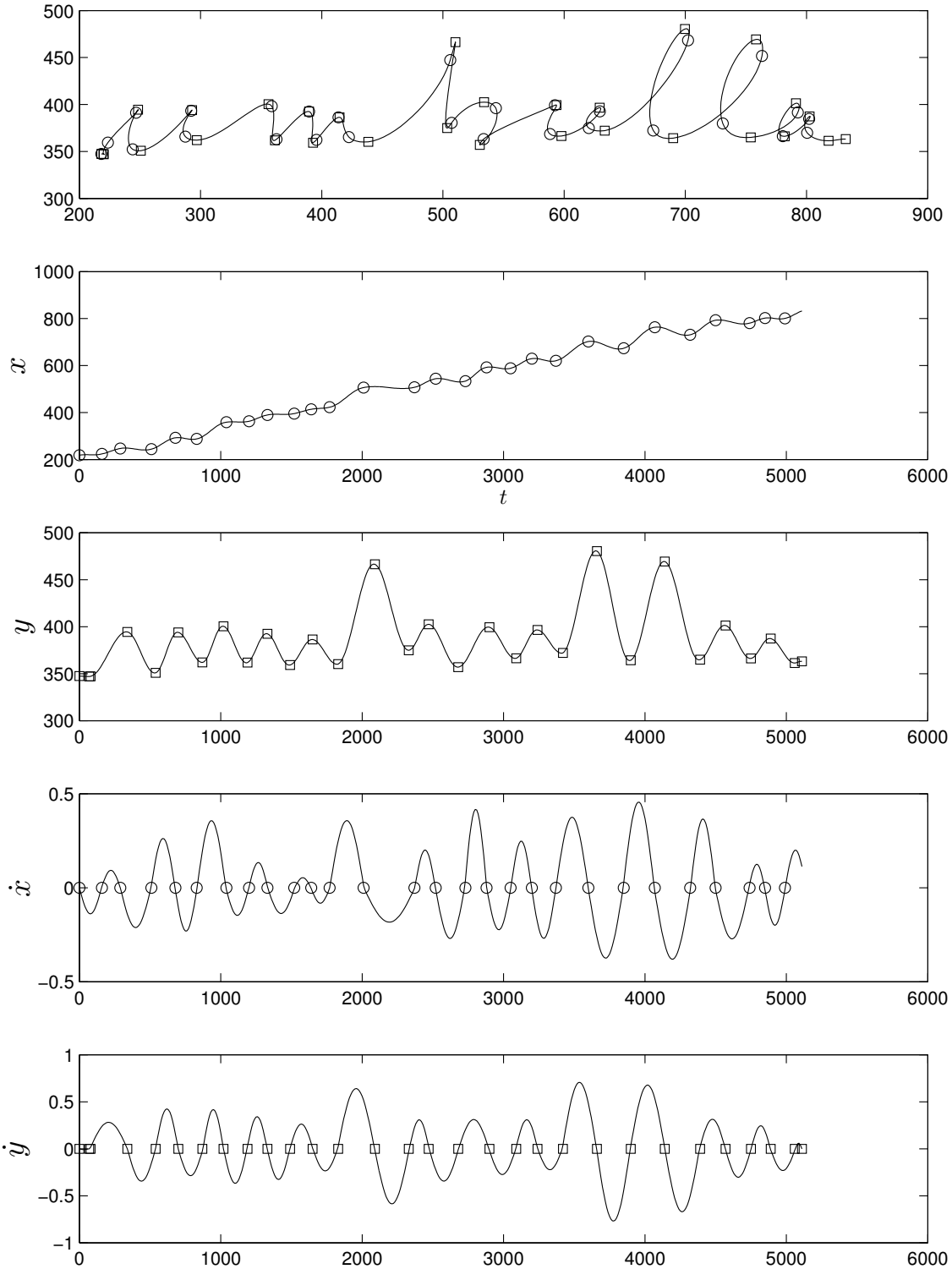


FIGURE 3.1 – Une trace produite par le modèle POMH (bandeau supérieur), les profils de position correspondants en x et en y (bandeaux du milieu) et les profils de vitesse (bandeaux du bas) en fonction du temps. Les cercles (resp. les carrés) représentent les moments d'annulation de la vitesse en x (resp. y).

3.1.2.1 Calcul de ω et ϕ

Entre deux zéros successifs $t_{x,i}$ et $t_{x,i+1}$ de la vitesse en x, $\dot{x}$ parcourt une demi-période de fonction sinus, donc :

$$t_{x,i+1} - t_{x,i} = \frac{\pi}{\omega_{x,i}} \quad \text{et} \quad \omega_{x,i} t_{x,i} + \phi_{x,i} = 0 \tag{3.5}$$

De même en y on a :

$$t_{y,i+1} - t_{y,i} = \frac{\pi}{\omega_{y,i}} \quad \text{et} \quad \omega_{y,i} t_{y,i} + \phi_{y,i} = 0 \tag{3.6}$$

3.1.2.2 Calcul du paramètre d'amplitude a

Considérons la fonction suivante :

$$f : x \mapsto a \sin(\omega x + \phi) \tag{3.7}$$

où a, ω et ϕ sont indépendants de x. Calculons la moyenne et la variance de f entre deux zéros successifs :

$$M = \frac{\omega}{\pi} \int_{\frac{-\phi}{\omega}}^{\frac{\pi - \phi}{\omega}} f(x)dx = \frac{2a}{\pi} \tag{3.8}$$

$$\begin{aligned} V &= \frac{\omega}{\pi} \int_{\frac{-\phi}{\omega}}^{\frac{\pi - \phi}{\omega}} (f(x) - M)^2 dx \\ &= \frac{a^2 \left(-8 + \pi^2\right)}{2\pi^2} \end{aligned} \tag{3.9}$$

Ces résultats montrent que l'amplitude d'un signal sinusoïdal sur une demi-période (zéro à zéro) peut être estimé grâce à la moyenne ou à la variance de ce signal sur la même demi-période, indépendamment de la phase et de la fréquence du signal.

3.1.2.3 Algorithme complet d'extraction des paramètres (FHA)

Supposons que l'échantillon d'écriture manuscrite enregistré soit représenté par une suite chronologique finie de positions datées :

$$S = (t_i, x_i, y_i)_{0 \leq i \leq N, N \in \mathbb{N}^*, \forall i > 0, t_i > t_{i-1}} \tag{3.10}$$

Voici les étapes utilisées pour extraire les paramètres des séries (3.3) and (3.4) de cette trace enregistrée :

Étape 1 $x = (x_i)_{0 \leq i \leq N}$ est dérivée selon $t = (t_i)_{0 \leq i \leq N}$:

$$\dot{x} = \left(\frac{x_i - x_{i-1}}{t_i - t_{i-1}} \right)_{0 < i \leq N} \tag{3.11}$$

$y = (y_i)_{0 \leq i \leq N}$ est dérivée selon $t = (t_i)_{0 \leq i \leq N}$:

$$\dot{y} = \left(\frac{y_i - y_{i-1}}{t_i - t_{i-1}} \right)_{0 < i \leq N} \tag{3.12}$$

Étape 2 Puisque tous nos calculs sont faits entre les zéros consécutifs de vitesse, nous ajoutons des vitesses initiales et finales nulles aux séries de vitesses. Cette hypothèse bien que parfois irréaliste nous épargne des calculs particuliers pour gérer le début et la fin de la trace (plus de détails à ce sujet en section 3.5.2).

Étape 3 Un algorithme de calcul des zéros est appliqué aux séries dérivées préalablement filtrées (en utilisant une fenêtre rectangulaire de taille 6, il s'agit donc d'un filtre passe-bas). Cela empêche l'algorithme de détection de zéros de trouver des groupes de zéros à causes des irrégularités dues à l'acquisition et exacerbées par la dérivation. Cette étape nous donne donc les deux séries $t_{x,0}, \ldots, t_{x,N_x}$ et $t_{y,0}, \ldots, t_{y,N_y}$.

Étape 4 Les séries de paramètres sont alors calculées :

$$\forall i, 0 \leq i \leq N_x :$$

$$\omega_{x,i} = \frac{\pi}{t_{x,i+1} - t_{x,i}} \tag{3.13}$$

$$\phi_{x,i} = -\frac{\pi t_{x,i}}{t_{x,i+1} - t_{x,i}} \tag{3.14}$$

$$a_{x,i} = \frac{\pi}{2} mean\,(\dot{x})_{[t_{x,i}, t_{x,i+1}[} \tag{3.15}$$

$$\forall i, 0 \leq i \leq N_y :$$

$$\omega_{y,i} = \frac{\pi}{t_{y,i+1} - t_{y,i}} \tag{3.16}$$

$$\phi_{y,i} = -\frac{\pi t_{y,i}}{t_{y,i+1} - t_{y,i}} \tag{3.17}$$

$$a_{y,i} = \frac{\pi}{2} mean\,(\dot{y})_{[t_{y,i}, t_{y,i+1}[} \tag{3.18}$$

Étape 5 (Synthèse) Les vitesses $\dot{x}$ et $\dot{y}$ peuvent être reconstruites en utilisant les équations (3.1) et (3.2) et les séries calculées précédemment. La trace (x, y) est finalement obtenue par intégration de ces séries de vitesses.

La figure 3.2 montre un exemple de trace enregistrée (en bleu) à laquelle on a superposé la trace reconstruite (en rouge) en utilisant l'algorithme précédent. Non seulement cette trace reconstruite est parfaitement lisible, mais les propriétés individuelles de la formation des lettres sont aussi capturées par POMH, malgré le fait que la trace soit assez complexe, incluant des changements abrupts de dérivées, correspondant à des caractéristiques topologiques de l'écriture (par exemple, des pics, des points de rebroussement, ...). De plus le modèle peut gérer les levées de crayons (voir figure 3.17). Plus important encore, le modèle capture la dynamique du mouvement : les deux traces peuvent être rejouées et l'on observe une excellente synchronisation.

3.1.3 Nombre minimum de paramètres

Reprenant les considérations de Teulings [1996], section 4.1, intéressons-nous au nombre de paramètres nécessaires pour reconstruire une trace. Selon le théorème d'échantillonnage de Shannon, le nombre minimum de paramètres nécessaires pour reconstruire un signal temporel de durée T et limité en fréquence à W est de $2WT$ [Jerri, 1977]. Si on suppose que les deux composantes x et y sont indépendantes alors il nous faut $4WT$ paramètres pour décrire exactement la trace.

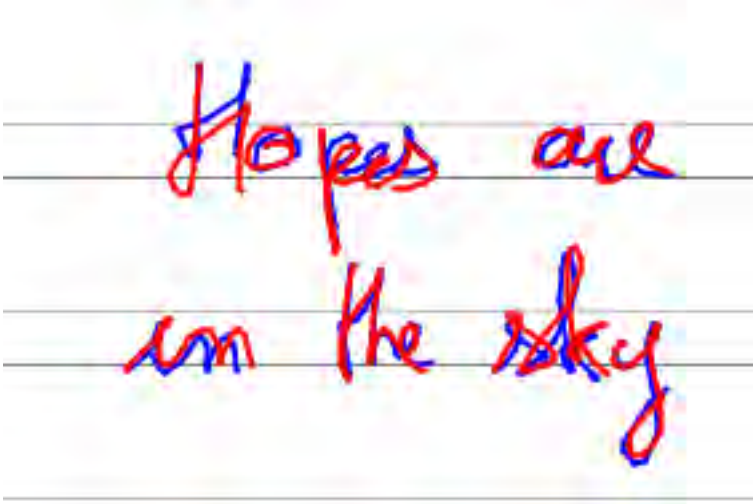


FIGURE 3.2 – Comparaison entre la trace originale (ligne bleue) et la trace reconstruite par POMH (ligne rouge).

Considérons la trace donnée en exemple figure 3.1, elle dure approximativement 5.1s. On prend 7Hz pour W [Teulings et Thomassen, 1979; Teulings et Maarse, 1984], donc le nombre de paramètres nécessaire est de 143. Notre modèle a capturé 28 moments d'annulation de la vitesse horizontale et 27 moments d'annulation de la vitesse verticale. D'après les équations 3.3 et 3.4 on utilise 4 paramètres par zéros. Donc au total nous utilisons 220 paramètres, et nous devrions être capable de reproduire exactement la trace initiale, ce qui n'est pas le cas : notre modèle est-il mauvais ?

En réalité, dans les séries décrites par 3.3 et 3.4, plusieurs paramètres sont redondants où inutiles pour la reconstruction. Une trace peut être reconstruite en utilisant seulement ces éléments :

- Les valeurs initiales $(t_{x,0}, a_{x,0}, \omega_{x,0}, \phi_{x,0})$ et $(t_{y,0}, a_{y,0}, \omega_{y,0}, \phi_{y,0})$
- Les séries $(a_{x,i}, \omega_{x,i})$, $0 \leq i \leq N_x$ et $(a_{y,i}, \omega_{y,i})$, $0 \leq i \leq N_y$

En effet, l'algorithme de reconstruction peut détecter les moments où les signaux $\dot{x}$ et $\dot{y}$ en cours de reconstruction s'annulent, ce qui rend inutile le stockage des $t_{x,i}$ et $t_{y,i}$. Par ailleurs en ces zéros, les ϕ peuvent être calculés.

Finalement, le nombre de paramètres nécessaire dans notre exemple est de 119. Notre modèle est donc parcimonieux, mais ne capture pas tout puisqu'il faudrait 143 paramètres pour pouvoir rendre exactement compte de la trace initiale. Notons que le nombre de paramètres utilisés par notre modèle dépend uniquement du nombre de zéros trouvés. Cela signifie que le nombre de paramètres extraits peut ou non se rapprocher du nombre de paramètres idéal, pour une trace donnée, en fonction de la performance de l'algorithme de calcul des zéros. Nous verrons dans la partie consacrée aux erreurs fréquentes (voir section 3.5) que celles-ci sont essentiellement dues à ce problème de recherche des zéros.

3.2 Comparaison avec le modèle d'Edelman et Flash (EFM)

Nous nous proposons ici d'évaluer la qualité de la reproduction issue des paramètres extraits par l'algorithme 3.1.2.3. Dans ce but, POMH est comparé au modèle d'Edelman et Flash présenté en détail en section 2.1.3.2 ; nous noterons ce modèle EFM. A notre connaissance,

c'est la première fois qu'un modèle oscillatoire de l'écriture est confronté à EFM sur des traces curvilignes (Plamondon et al. [1993] ont comparé 23 modèles de génération de traces sur un mouvement rectiligne rapide et de courte durée ; parmi ces modèles, le modèle d'Hollerbach et le modèle d'Edelman et Flash).

3.2.1 Expérience

3.2.1.1 Participants

Quatre volontaires bénévoles, trois hommes et une femme ont participé à cette expérience. Deux participants se sont déclarés droitiers et deux autres gauchers. Ils ont affirmé que leur acuité visuelle était bonne et qu'ils ne souffraient pas de troubles les empêchant de détecter ou de voir une trace écrite.

3.2.1.2 Matériel

La tâche graphique à été effectuée sur une tablette graphique associée à un ordinateur. Il s'agit d'une Wacom DTZ-1200W/G avec un écran LCD de 261.1×163.2 mm d'une résolution de 1280×800 pixels inséré dans un cadre en plastique noir (405×270×17 mm) pouvant être orienté à souhait comme une feuille de papier. L'écran affichait un fond blanc avec des lignes grises espacées de 150 mm. Le stylet associé avait approximativement la même taille (175 mm de long et un diamètre de 14 mm en moyenne) et le même poids (17 g) qu'un stylo à bille normal. Les participants étaient assis sur une chaise haute ajustable, face à la tablette posée sur une table. Il leur a été demandé d'adopter une posture d'écriture confortable. Sitôt que le stylet touchait la surface de l'écran, les cordonnées spatiales x et y de la trajectoire tracée étaient échantillonnées à une fréquence de 100Hz avec une résolution spatiale de 0.2 mm. La trace produite était affichée en temps réel sur l'écran dans le but d'offrir aux participants un retour visuel proche de celui qu'ils auraient en écrivant avec un crayon sur une feuille. Quand le stylet quittait la surface de la tablette, la trace écrite était enregistrée dans un fichier afin de permettre une analyse ultérieure.

3.2.1.3 Procédure

L'ensemble des traces requises était composé des quatre formes prototypiques présentées par Edelman et Flash [1987] (voir section 2.1.3.2 pour plus de détails) : un crochet, un U, une boucle (gamma à l'envers) et un cercle. Pour chaque forme, il a été demandé aux participants d'en écrire 60 exemplaires à vitesse spontanée. Parmi ces 60 exemplaires, le deuxième tiers (20 exemplaires) a été utilisé dans l'analyse suivante (les 20 premiers ont été supprimés afin de permettre aux participants de s'habituer à la forme et à la tablette, les 20 derniers car ils ont indiqué éprouver une fatigue après 45 exemplaires tracés). Il n'a pas été demandé aux participants d'immobiliser le crayon sur la tablette avant de commencer leur trace si bien que la vitesse du début d'enregistrement de la trace n'est pas nécessairement nulle, ceci afin que le mouvement des participants soit le plus naturel possible. L'ensemble de l'expérience durait environ 15 min.

3.2.1.4 Analyse des données

Pour chaque type de trace produite, quatre variables dépendantes ont été étudiées : x, y, $\dot{x}$ et $\dot{y}$ (les positions et les vitesses). A la suite d'Edelman et Flash [1987], une estimation du degré d'ajustement entre la trace produite et la trace reconstruite a pu être mesurée par la moyenne des 20 coefficients de corrélation entre les profils enregistrés de position et de vitesse et leur pendant simulé (i.e., $r_{x,\hat{x}}$, $r_{y,\hat{y}}$, $r_{\dot{x},\hat{\dot{x}}}$ and $r_{\dot{y},\hat{\dot{y}}}$).

Pour le calcul des coefficients de corrélation la formule classique de Pearson-R

$$r_{a,b} = \frac{a^T b}{\sqrt{a^T a \; b^T b}}, \tag{3.19}$$

a été choisie plutôt que celle donnée par Edelman et Flash parce que cette dernière, bien que présentée comme donnant des valeurs comprises entre 0 et 1, peut donner, en réalité, des valeurs supérieures à 1. L'indice de corrélation mesure la ressemblance topologique entre la trace produite et la trace reconstruite. L'indice de corrélation varie de 1 pour des signaux identiques à -1 pour des signaux opposés, en passant par 0 quand il n'y a aucune ressemblance entre les deux traces.

3.2.1.5 Dissimilarité au sens de Minkovski

Un évaluation globale de la ressemblance entre POMH et EFM à été réalisée en utilisant la p-dissimilarité de Minkovski. Cette métrique capture la distance entre les deux modèles dans un espace à 4 dimensions, dans lequel chaque dimension correspond aux quatre coefficients de corrélation pour chaque variable dépendante a et son pendant simulé b. Pour chaque participant, EFM et POMH représentent des points dans cet espace à 4 dimensions et la distance de Minkovsky , $d(EFM, POMH)$, mesure l'écart entre les deux. La distance de Minkowski est une généralisation de la distance Euclidienne :

$$d(EFM, POMH) = \left(\sum_{i=1}^{p} |r_{ab\_EFM} - r_{ab\_POMH}|^p \right)^{\frac{1}{p}}, \tag{3.20}$$

où p est le nombre de dimensions de l'espace. Dans notre cas 1.414 est la distance maximale possible, car la dissimilarité maximale correspond au cas où l'un des modèles s'ajuste parfaitement ($r_{a,b} = 1$) et l'autre pas du tout ($r_{a,b} = 0$) :

$$d_{max}(EFM, POMH) = \left(\sum_{i=1}^{4} |1 - 0|^4 \right)^{\frac{1}{4}} = 1.414 \tag{3.21}$$

3.2.1.6 Analyse statistique

Une analyse de 2 (Modèle = POMH,EFM) Friedman ANOVA a été effectuée sur chaque variable dépendante pour tester s'il y avait une différence entre l'ajustement de EFM et celui de POMH. Une analyse additionnelle de 4 (Type=cercle,gamma,crochet,U) Friedman ANOVA a été effectuée sur la distance de Minkovsky pour tester si la ressemblance dépend de la forme de la trace.

3.2.2 Résultats

3.2.3 Inspection visuelle de résultats particuliers

Les figures 3.3 à 3.6 montrent pour chaque prototype un exemple particulier tracé par un des participants. Sur chaque figure, le bandeau du haut montre la trace, le bandeau du milieu montre les profils de position en x et y et le bandeau du bas les profils de vitesse en x et y. A première vue, POMH (en gris foncé) et EFM (en gris clair) reproduisent assez fidèlement la trace originale. Les coefficients de corrélation présentés pour chaque profil excèdent tous 0.97.

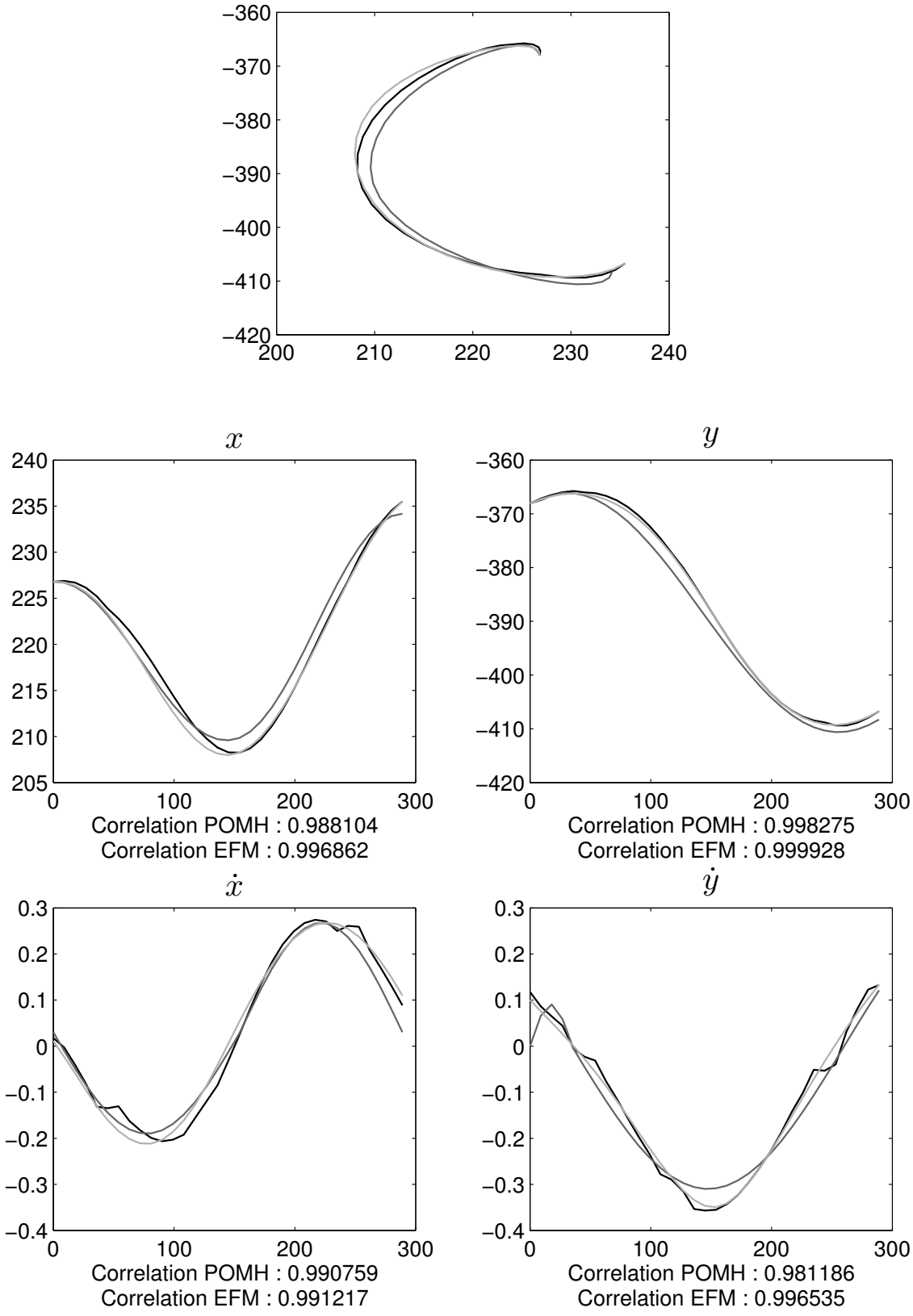


FIGURE 3.3 – Comparaison entre la trace originale, en noire, la trace reconstruite par POMH, en gris foncé, et celle reconstruite par EFM, en gris clair, pour le prototype 'crochet'.

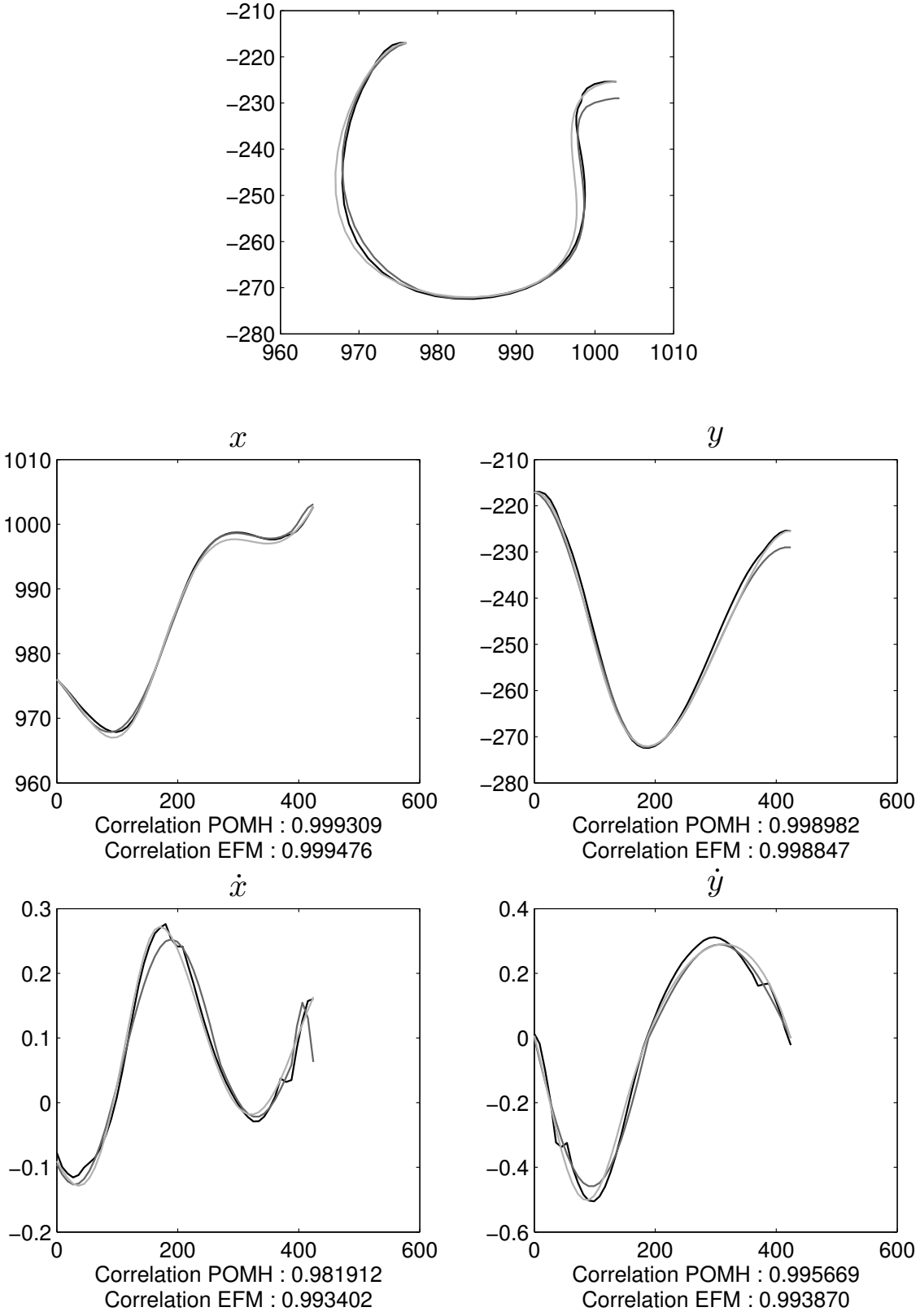


FIGURE 3.4 – Comparaison entre la trace originale, en noire, la trace reconstruite par POMH, en gris foncé, et celle reconstruite par EFM, en gris clair, pour le prototype 'U'.

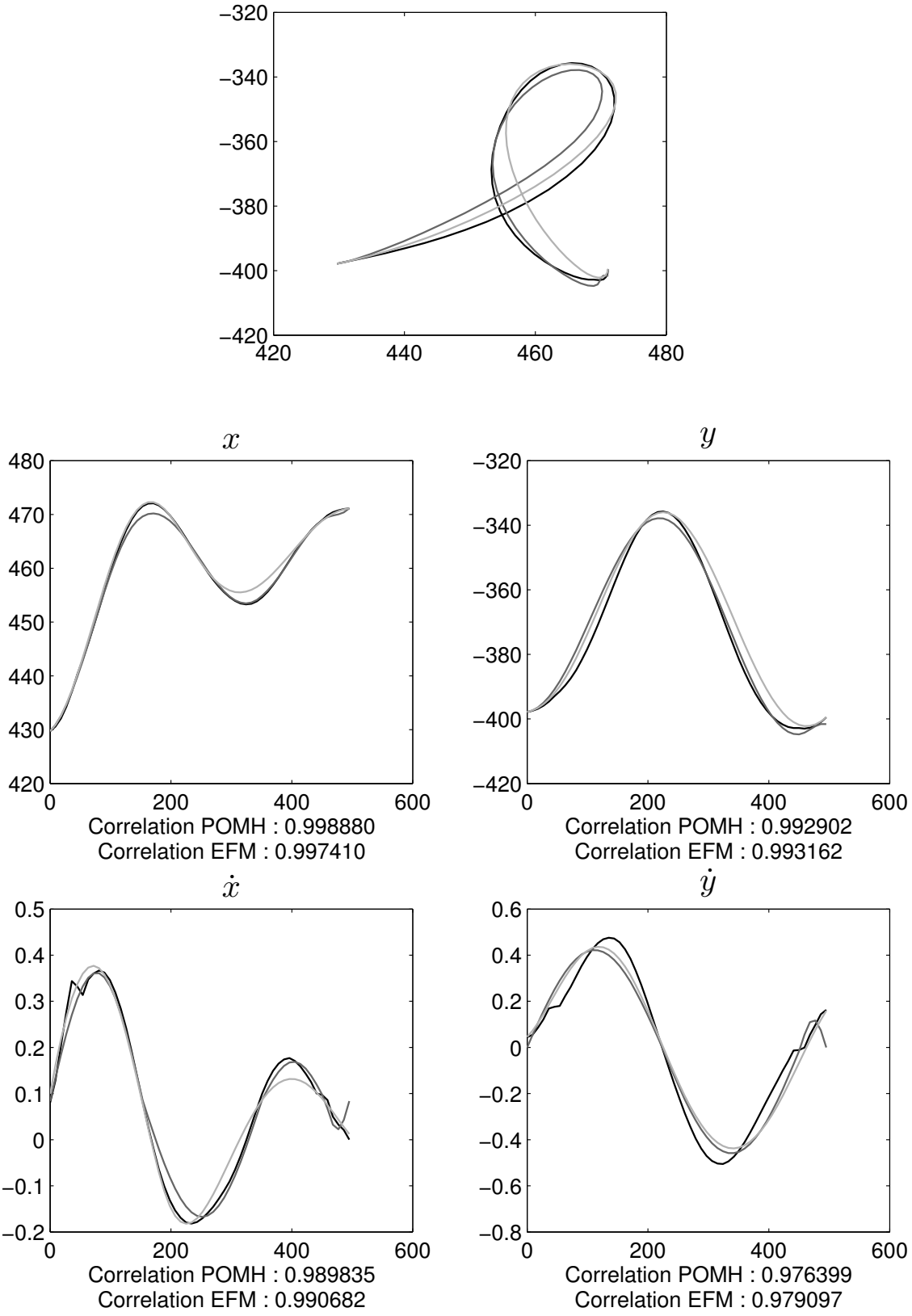


FIGURE 3.5 – Comparaison entre la trace originale, en noire, la trace reconstruite par POMH, en gris foncé, et celle reconstruite par EFM, en gris clair, pour le prototype 'gamma'.

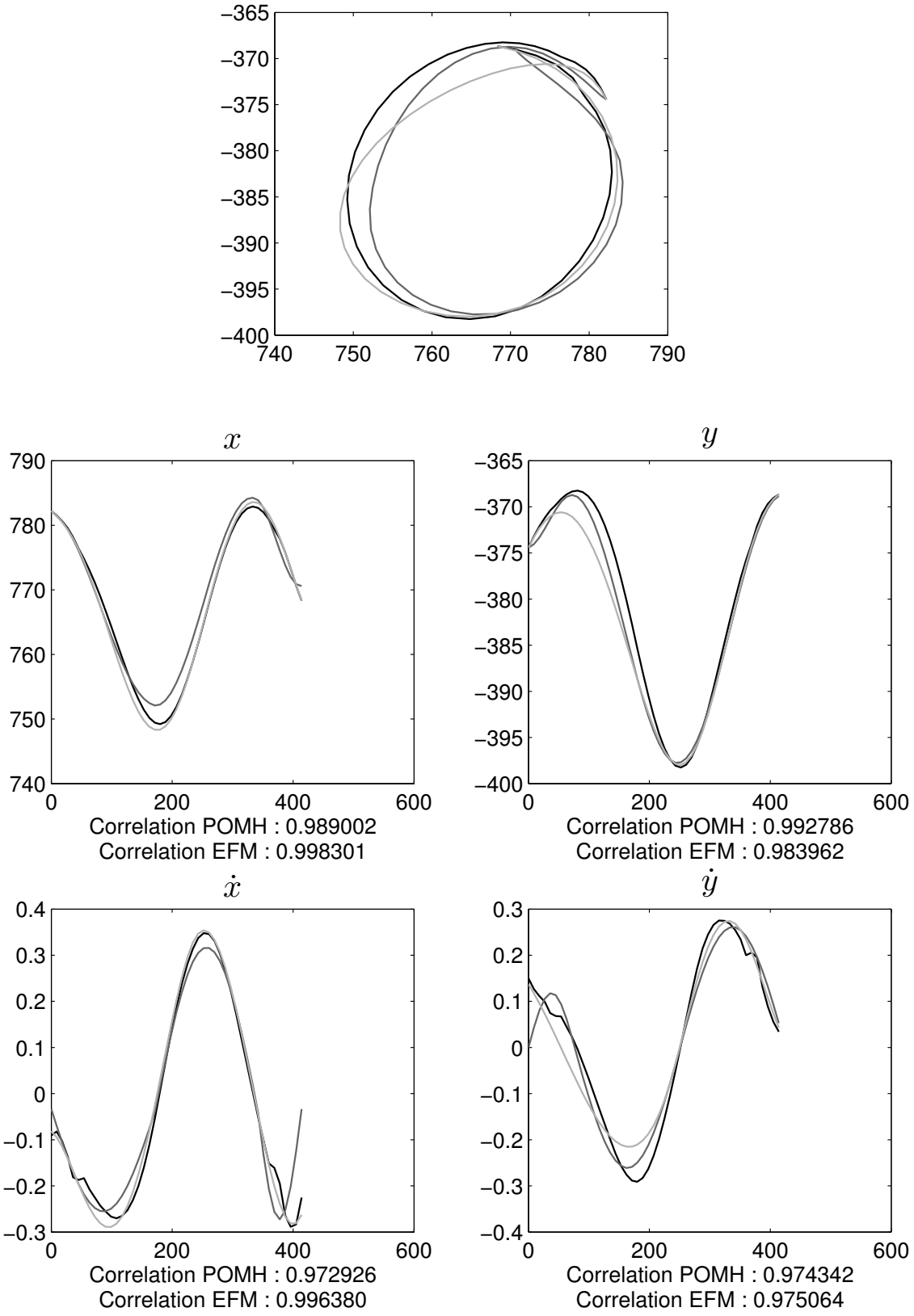


FIGURE 3.6 – Comparaison entre la trace originale, en noire, la trace reconstruite par POMH, en gris foncé, et celle reconstruite par EFM, en gris clair, pour le prototype 'cercle'.

TABLE 3.1 – Moyennes et écarts-type des coefficients de corrélation pour les quatre variables dépendantes (x, y, $\dot{x}$ et $\dot{y}$). La dernière colonne donne le résultat de l'analyse ANOVA, une astérisque indique la détection d'une différence statistique ($p < 0.05$).

	POMH	Edelman-Flash	Difference	Friedman ANOVA
	mean(SD)		mean	p
Cercle				
x	0,975(0,016)	0,922(0,083)	0,053	0,317
$\dot{x}$	0,962(0,007)	0,883(0,059)	0,079	0,046*
y	0,987(0,005)	0,910(0,070)	0,077	0,046*
$\dot{y}$	0,956(0,008)	0,881(0,062)	0,075	0,046*
Gamma				
x	0,990(0,001)	0,930(0,078)	0,061	0,046*
$\dot{x}$	0,956(0,006)	0,882(0,050)	0,074	0,046*
y	0,988(0,002)	0,885(0,098)	0,103	0,046*
$\dot{y}$	0,954(0,005)	0,819(0,107)	0,135	0,046*
Crochet				
x	0,900(0,130)	0,907(0,128)	-0,007	0,046*
$\dot{x}$	0,899(0,099)	0,896(0,053)	0,004	1,000
y	0,950(0,055)	0,926(0,087)	0,024	0,371
$\dot{y}$	0,812(0,125)	0,840(0,107)	-0,029	0,046*
U				
x	0,813(0,282)	0,853(0,171)	-0,040	1,000
$\dot{x}$	0,876(0,126)	0,911(0,093)	-0,036	0,046*
y	0,917(0,100)	0,942(0,066)	-0,025	0,317
$\dot{y}$	0,789(0,142)	0,873(0,090)	-0,084	0,046*

3.2.3.1 Coefficients de corrélation

Les coefficients de corrélation sont donnés dans le Tableau 3.1. Pour tous les prototypes et pour toutes les variables dépendantes, la différence de corrélation entre POMH et EFM est inférieure à 0.1. Une différence positive (resp. négative) indique que POMH s'est mieux (resp. moins bien) ajusté à la trace réelle que EFM. Une astérisque montre les différences statistiques découvertes par ANOVA ($p < 0.05$).

L'analyse ANOVA effectuée sur le cercle indique une différence statistique pour 3 des quatre variables dépendantes ($\dot{x}$, y et $\dot{y}$) et ce en faveur de POMH. En ce qui concerne le gamma inversé, ANOVA indique une différence statistique sur les quatre variables dépendantes en faveur de POMH. L'étude ANOVA pour le crochet repère une différence statistique en faveur de EFM pour x et $\dot{y}$, cependant, les différences associées sont inférieures aux deux études précédentes. Il en est de même pour le U où ANOVA détecte une différence significative en faveur de EFM pour x et $\dot{y}$ mais avec des valeurs plus faibles que pour les différences détectées pour les deux premiers prototypes.

Sur l'ensemble des cas, 6 fois sur 16 EFM a obtenu de meilleurs résultats, dont 4 fois avec une différence statistiquement significative. Sur les 10 fois où POMH à obtenu de meilleurs résultats, 7 sont significatifs.

3.2.3.2 Mesure de la p-dissimilarité de Minkovsky

La distance de Minkovsky moyenne était de 0.242 (SD = 0.022) pour le cercle, 0.298 (SD = 0.065) pour gamma, 0.151 (SD=0.063) pour le crochet et 0.0228 (SD=0.041) pour le U.

Ces valeurs sont à comparer avec la distance maximale 1.414 (voir l'équation 3.21). Une analyse 4(Prototype) de Friedman ANOVA effectuée sur ces valeurs de similarité n'a pas révélé d'effet des prototypes sur les résultats ($F(3) = 7.5$, $p < 0.057$).

D'une manière générale, tous ces résultats indiquent que les qualités d'ajustement de EFM et POMH sont relativement similaires.

3.3 Performances de l'algorithme d'extraction de POMH

On se propose ici de comparer l'algorithme présenté en section 3.1.2.3 à une méthode d'optimisation multi-points de départ basée sur l'algorithme des Régions de Confiances Réflexif (Trust Region Reflective) tel qu'il est implémenté dans la fonction `lsqnonlin` de `MATLAB`.

3.3.1 Formulation du problème

On se place au début de l'étape 4 de l'algorithme de POMH (3.1.2.3). Il s'agit de calculer $(a_{xi}, \omega_{xi}, \phi_{xi})_{1\leq i\leq N_x}$ et $(a_{y_i}, \omega_{y_i}, \phi_{y_i})_{1\leq i\leq N_y}$ en utilisant $(t_{0xi})_{1\leq i\leq N_x}$, $(t_{0yi})_{1\leq i\leq N_y}$ et les deux dérivées temporelles $\dot{x}$ and $\dot{y}$ calculées aux étapes précédentes de l'algorithme. Dans un souci de concision, on note θ la variable de notre problème d'optimisation, avec :

$$\theta = ((a_{xi}, \omega_{xi}, \phi_{xi})_{1\leq i\leq N_x}, (a_{y_i}, \omega_{y_i}, \phi_{y_i})_{1\leq i\leq N_y}) \tag{3.22}$$

Ce problème peut être vu tant comme le résultat d'une méthode d'estimation du maximum de vraisemblance que comme un problème d'ajustement de courbe (curve-fitting problem). Le problème est formulé de la manière suivante :

$$\begin{aligned} \underset{\theta}{\operatorname{argmin}} \quad f(\theta) = \quad & \sum_{j=0}^{N_x-1} \sum_{i=\iota_x(j)}^{\iota_x(j+1)-1} \left(\dot{x}_i - a_{xj}\sin(\omega_{xj}t_i + \phi_{xj})\right)^2 \\ & + \sum_{j=0}^{N_y-1} \sum_{i=\iota_y(j)}^{\iota_y(j+1)-1} \left(\dot{y}_j - a_{y_j}\sin(\omega_{y_j}t_i + \phi_{y_j})\right)^2, \end{aligned} \tag{3.23}$$

où ι_x et ι_y sont les fonctions qui retournent les indices correspondants aux points de vitesse nulle dans les séries $\dot{x}$ et $\dot{y}$. On note θ_s la solution de ce problème.

De θ_s, on reconstruit les dérivées $\dot{x}$ and $\dot{y}$ ($\dot{x}$ and $\dot{y}$ sont des vecteurs colonne de taille N). On note $\mathcal{C}_s = (\dot{x}\ \dot{y})$. $\mathcal{C}_s$ est basé sur la même série temporelle utilisée pour échantillonner la trace enregistrée $\mathcal{C}_o$. Une mesure de l'erreur entre la trace enregistrée et la trace synthétisée est donnée par :

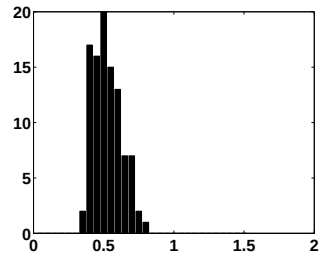
$$err(\mathcal{C}_s) = \frac{\|\mathcal{C}_s - \mathcal{C}_o\|_2}{\|\mathcal{C}_o\|_2}. \tag{3.24}$$

Notons que résoudre le problème (3.23) est équivalent à trouver $\mathcal{C}_s$ qui minimise $err(\mathcal{C}_s)$. On considère qu'une solution approchée du problème donnant une erreur inférieure à 0.5 est suffisante.

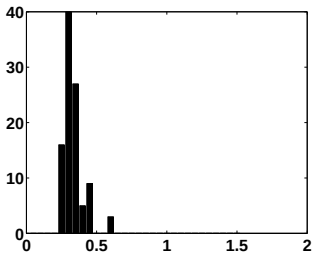
3.3.2 Application d'une méthode d'optimisation usuelle

Puisque le problème (3.23) est un problème d'ajustement non linéaire au sens des moindres carrés, nous avons choisi d'appliquer l'algorithme Trust-Region-Reflective qui est une méthode à régions de confiances (subspace trust-region). Cet algorithme est basé sur une méthode de Newton (interior-reflective Newton Method) présentée dans Coleman et Li [1993] et Coleman et Li [1994]. On utilise ici l'implémentation `MATLAB` de cet algorithme accessible par la fonction `lsqnonlin`.

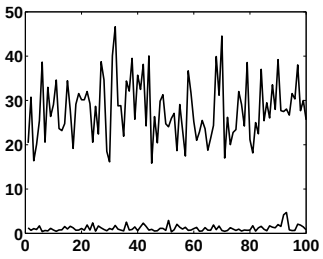
On peut appliquer cet algorithme en utilisant des stratégies différentes. Le but des prochains paragraphes est de déterminer la plus efficace.



(a) Distribution des erreurs dans le cas où l'on considère plusieurs sous-problèmes.



(b) Distribution des erreurs dans le cas où l'on considère le problème comme un tout.



(c) Temps d'éxécution (en secondes) sur 100 itérations. La courbe du haut représente le cas des sous-problèmes et celle du bas le cas du problème considéré en entier.

FIGURE 3.7 – Comparaison entre le cas où on considère le problème comme un ensemble de sous problèmes et le cas où l'on considère le problème comme un tout.

3.3.2.1 Doit-on diviser le problème en sous-problèmes ?

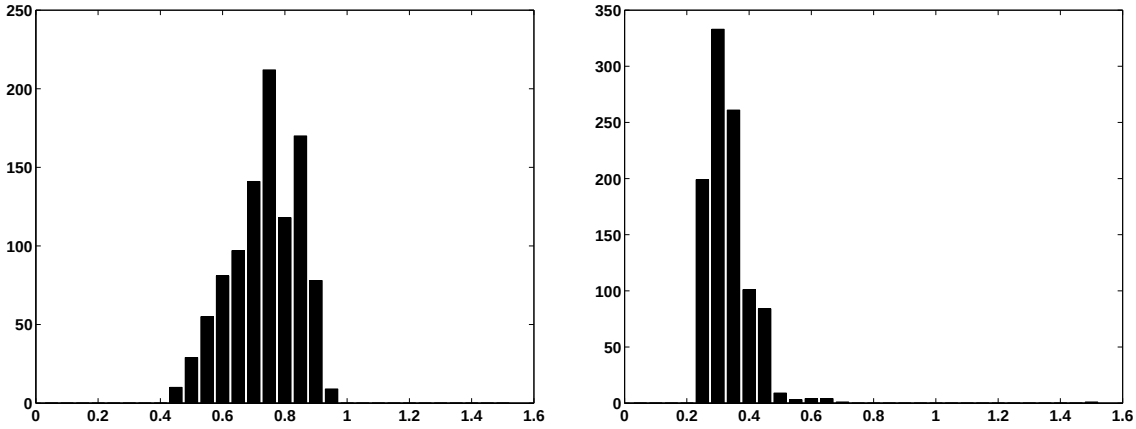
Certaines variables du problème (3.23) sont indépendantes. Cela est dû au fait que dans POMH, les paramètres sont constants par morceaux, chaque morceau étant indépendant des autres. Par ailleurs, dans notre modèle, il n'y a pas de lien entre les composantes : x et y sont indépendantes. Le problème (3.23) peut donc être divisé en un ensemble $\mathcal{S}$ de sous-problèmes :

$$\mathcal{S} = \left\{ \sum_{i=\iota_d(j)}^{\iota_d(j+1)-1} (d_i - a_{dj}\sin(\omega_{dj}t_i + \phi_{dj}))^2 : d \in \{(x),(y)\}, j \in [\![0, N_d - 1]\!] \right\} \tag{3.25}$$

Au lieu d'appliquer l'algorithme sur l'ensemble du problème, on peut être tenté de l'appliquer sur chacun des éléments de $\mathcal{S}$. La figure 3.7 montre les résultats de l'algorithme lancé 100 fois, sur une même trace, dans le cas où l'on considère le problème comme un tout et dans le cas où l'on considère chaque sous problème (tous autres paramètres égaux par ailleurs). On remarque que dans le premier cas (3.7b) la distribution des erreurs est relativement faible et inférieure à 0.5 (avec une moyenne de 0.33), dans la grande majorité des cas, tandis que le temps d'exécution tourne autour de la seconde (3.7c). En revanche, dans le second cas (3.7a), les erreurs sont distribuées avec une plus forte dispersion (de 0.4 à 0.7) autour d'une moyenne de 0.52. Par ailleurs, le temps d'exécution (3.7c) est en moyenne de 28 secondes soit presque 30 fois plus grand que dans le premier cas. Pour ces raisons, il convient de considérer l'application de l'algorithme uniquement sur l'ensemble du problème.

3.3.2.2 Solutions initiales

Lorsqu'on utilise une méthode de recherche locale, ce qui est le cas de la méthode de Newton utilisée ici, nous devons lui fournir une solution initiale à partir de laquelle elle commencera sa recherche. Le choix de cette solution initiale a un impact crucial sur la convergence de la méthode vers une solution satisfaisante. L'importance de ce choix nous amène à essayer différentes stratégies : (i) on peut appliquer la méthode à plusieurs points de départ choisis uniformément dans notre domaine de travail ; (ii) on peut l'appliquer à plusieurs points de départ générés selon une distribution normale dont les paramètres peuvent être calculés



(a) Solutions initiales distribuées uniformément. (b) Solutions initiales distribuées normalement

FIGURE 3.8 – Distribution des erreurs pour 1000 exécutions de l'algorithme d'optimisation pour deux types de distributions pour les solutions initiales (en abscisses les erreurs, en ordonnées le nombre d'exécutions pour un intervalle d'erreur donnée).

statistiquement a priori ; (iii) on peut l'appliquer à une unique solution initiale définie comme la moyenne de la distribution issue de l'étude statistique précédente, dans ce cas, la méthode n'est lancée qu'une fois.

On exécute cet algorithme 1000 fois,sur une même trace, dans les deux premiers cas (i) et (ii). Les résultats sont présentés en figure 3.8 ; pour le cas (iii), dans lequel seulement une situation initiale est considérée, on obtient une erreur de 0.8. La distribution des erreurs pour le cas (ii), où les solutions initiales sont choisies selon une distribution normale, est plus favorable que la distribution des erreurs dans la cas (i), où les solutions initiales sont choisies uniformément. La moyenne de cette distribution est d'ailleurs largement inférieure à 0.8. Il semble donc que la stratégie (ii) doive être retenue.

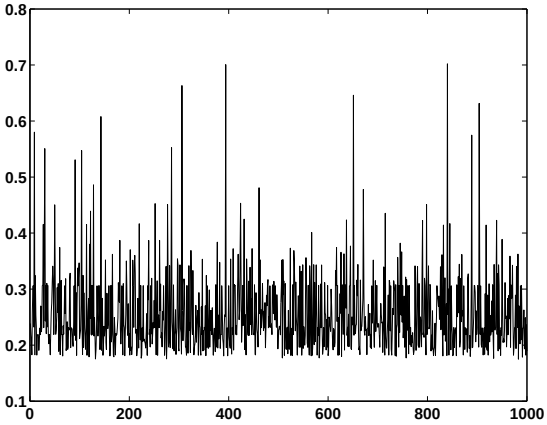
3.3.2.3 Combien d'exécutions de l'algorithme TRR ?

Dans les sous-sections précédentes il a été montré que le meilleur moyen d'appliquer l'algorithme TRR (Trust Region Reflective) au problème était de l'appliquer à l'ensemble du problème, et de l'appliquer plusieurs fois avec des solutions initiales choisies aléatoirement selon une distribution normale dont les paramètres sont déterminés statistiquement. Une question reste : combien de fois doit-on appliquer l'algorithme pour avoir une solution satisfaisante ?

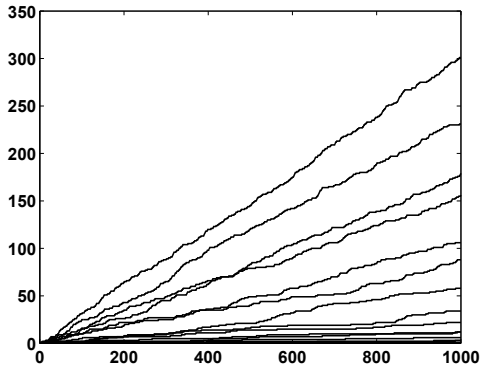
Supposons que 1000 exécutions de l'algorithme nous apportent toujours des solutions approchées suffisantes au problème 3.23. Notons s la meilleure d'entre-elles. Combien de fois devons nous exécuter l'algorithme TRR pour être sûrs d'avoir une solution approchée ps suffisamment bonne comparée à s telle que $\|err(s) - err(ps)\| < \epsilon$? Cette question fait sens car, comme nous pouvons le voir figure 3.9a, les meilleures solutions locales trouvées semblent être presque égales et réparties uniformément sur le nombre d'itérations.

Supposons maintenant qu'à chaque exécution, on ait la probabilité p d'obtenir une solution approchée suffisante ps. Notons n le nombre de fois qu'il faut exécuter l'algorithme TRR afin d'avoir une probabilité q d'obtenir une solution approchée suffisante ps. La condition $1 - (1-p)^n > q$ doit être satisfaite ce qui donne :

$$n \geqslant \left\lceil \frac{\ln(1-q)}{\ln(1-p)} \right\rceil . \tag{3.26}$$



(a) Erreurs obtenues pour 1000 exécutions de l'algorithme TRR appliqué à une trace donnée.



(b) Nombre de bonnes solutions *ps* rencontrées en fonction du nombre d'exécutions de l'agorithme TRR pour 15 traces différentes.

FIGURE 3.9 – Erreurs obtenues pour 1000 exécutions de l'algorithme TRR sur différentes traces.

Cependant, si l'on considère la figure 3.9b, même si l'hypothèse que toutes les solutions approchées suffisamment satisfaisantes sont distribuées uniformément sur le nombre d'itérations de TRR, la probabilité p en découlant semble être très différente d'une trace étudiée à une autre. On ne peut donc pas statuer a priori sur la valeur de n.

3.3.2.4 Conclusion

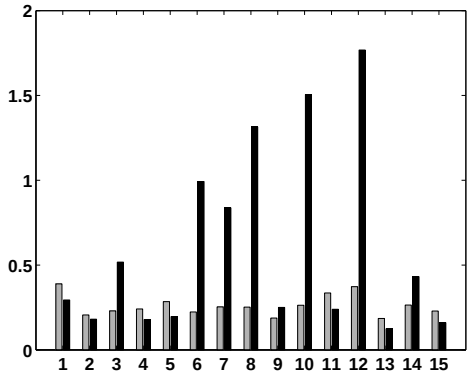
Dans cette section nous avons examiné différentes possibilités afin d'appliquer efficacement l'algorithme TTR pour résoudre le problème 3.23. Cet algorithme doit être appliqué sur le problème vu dans son ensemble. De plus, il doit être appliqué plusieurs fois avec des solutions initiales réparties normalement autour d'une moyenne calculée statistiquement. Enfin, le nombre d'exécutions nécessaires pour obtenir une solution suffisante ne peut être déterminé a priori.

Dans les sections suivantes, les tests seront effectués pour un nombre fixe d'exécutions noté N. D'autre part on suppose que l'étude statistique nécessaire pour déterminer les paramètres de la distribution normale utilisée pour sélectionner les solutions initiales ne consomme pas de temps. Enfin, l'algorithme TRR et la façon de l'appliquer sera noté MS_N (pour méthode multi-départ (multi-start) utilisant N solution initiales). Nous appellerons aussi l'algorithme présenté en étape 4 de la section 3.1.2.3, FHA, pour approximation rapide de fonction harmonique (Fast Harmonic Approximation).

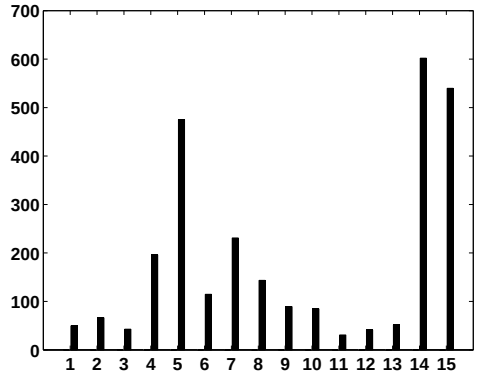
3.3.3 Comparaison entre MS_{50} and FHA

Dans cette section nous comparons donc la méthode présentée en section 3.3.2 avec FHA présenté en section 3.1.2.3. Les deux algorithmes ont été exécutés sur 15 traces. La figure 3.10 résume les résultats : les résultats de FHA sont représentés en gris, ceux de MS_{50} en noir.

En raison du choix aléatoire des solutions initiales fournies à MS_{50}, cette méthode montre une grosse variabilité en terme de précision (voir figure 3.10a) ; ainsi, dans 6 cas sur 15, MS_{50} donne une solution dont l'erreur est supérieure à 0.5. A l'opposé, FHA montre

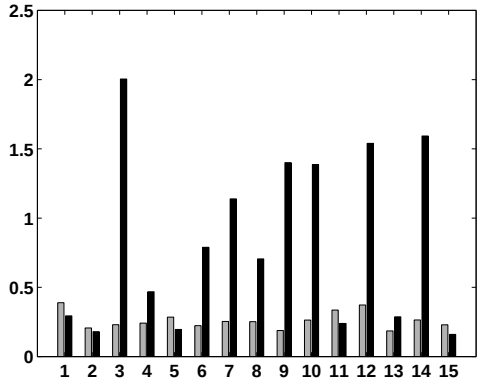


(a) Comparaison en erreur des solutions trouvées par FHA (en gris) et MS_{50} (en noir).

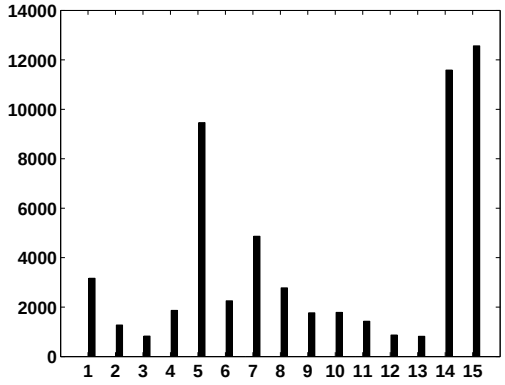


(b) Comparaison temporelle entre les deux méthodes (en secondes). Pour FHA les temps de calcul sont compris entre 0.01s et 0.03s (ce qui explique qu'ils ne soient pas visibles).

FIGURE 3.10 – Comparaison entre les deux méthodes sur 15 traces. FHA in gris, MS_{50} en noir.



(a) Comparaison en erreur des solutions trouvées par FHA (en gris) et MS_{1000} (en noir).



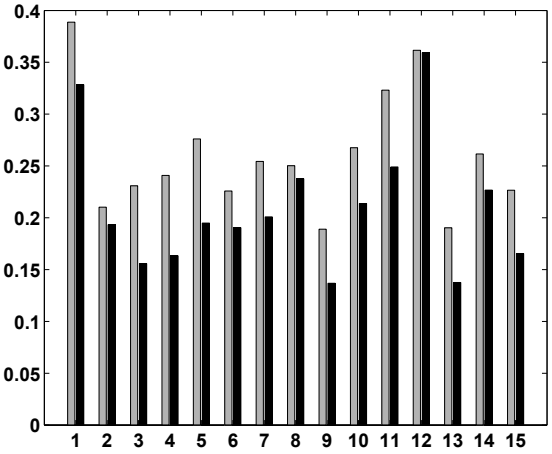
(b) Comparaison des performances temporelles entre FHA et MS_{1000} (en secondes). Pour FHA les temps de calcul sont compris entre 0.01s et 0.03s (ce qui explique qu'ils ne soient pas visibles).

FIGURE 3.11 – Comparaison entre les deux méthodes sur 15 traces. FHA in gris, MS_{1000} en noir.

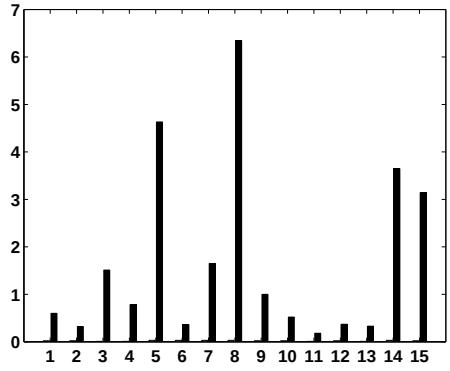
des résultats plus réguliers et ses solutions ont toujours une erreur inférieure à 0.5. En ce qui concerne le temps d'exécution, FHA montre des performances constantes tandis que les performances MS_{50} dépendent grandement des solutions initiales utilisées. Dans la plupart des cas notre méthode heuristique FHA est de 50 à 1000 fois plus rapide que MS_{50}.

Les mêmes commentaires validant l'efficacité de notre méthode heuristique FHA sont applicables aux résultats obtenus si l'on compare FHA avec MS_{1000} (en utilisant 1000 solutions initiales au lieu de 50), voir figure 3.11. Par ailleurs, la précision des solutions obtenues par MS_{1000} ne semble pas sensiblement meilleure que celle obtenue par MS_{50} alors que les temps d'exécution, eux, augmentent beaucoup.

La raison pour laquelle l'algorithme FHA fonctionne si bien est qu'il est directement



(a) Comparaison des solutions trouvées par FHA (in gris) et FHAo (en noir)



(b) Comparaison des performances temporelles entre FHA et FHAo (en seconde). Pour FHA, les temps d'exécution sont compris entre 0.01s et 0.03s.

FIGURE 3.12 – Comparaison entre FHA en gris et FHAo en noir.

relié aux propriétés de notre modèle où les paramètres changent aux moments d'annulation de la vitesse. Cela permet de calculer simplement et rapidement les fréquences et les phases.

La figure 3.12 montre des résultats correspondant à l'utilisation de la solution approchée calculée par FHA comme solution initiale pour TRR (on note cette méthode FHAo, FHA optimisé). On remarque que dans ce cas les solutions trouvées par TTR sont meilleures que celles de FHA. Cependant, les temps de calcul restent beaucoup plus élevés.

3.4 Exemples plus complexes

Après avoir comparé POMH avec le modèle d'Edelman-Flash, et après avoir comparé FHA, l'algorithme utilisé par POMH à une méthode d'optimisation répendue, nous appliquons POMH à des traces plus complexes telles que des lettres, des mots, des signatures et des phrases. Pour nombre de cas, un aperçu visuel est donné. Dans certains des cas, afin d'avoir une appréciation plus quantifiée du degré de similitude entre la trace originale et la trace reconstruite par POMH, les coefficients de corrélation sur chacun des signaux composant la trace (positions et vitesses) sont donnés. Plus rarement, quand les données étaient disponibles, une étude plus approfondie de cette similitude à été effectuée au moyen d'expériences.

3.4.1 Lettres

Les figures 3.13 et 3.14 montrent deux exemples de traces enregistrées pour les lettres 'b' et 'h' ainsi que les traces correspondantes restituées par POMH. On remarque que les traces sont visuellement proches. Les corrélations associées aux profils de position et aux profils de vitesse sont supérieures à 0.96.

3.4.1.1 Méthode

Cette expérience reprend les conditions expérimentales présentées en section 3.2.1.

Procédure Il a été demandé aux quatre participants d'écrire 6 lignes de 'b' puis 6 lignes de 'h'. Chaque trace a été enregistrée individuellement. Pour chaque lettre et chaque

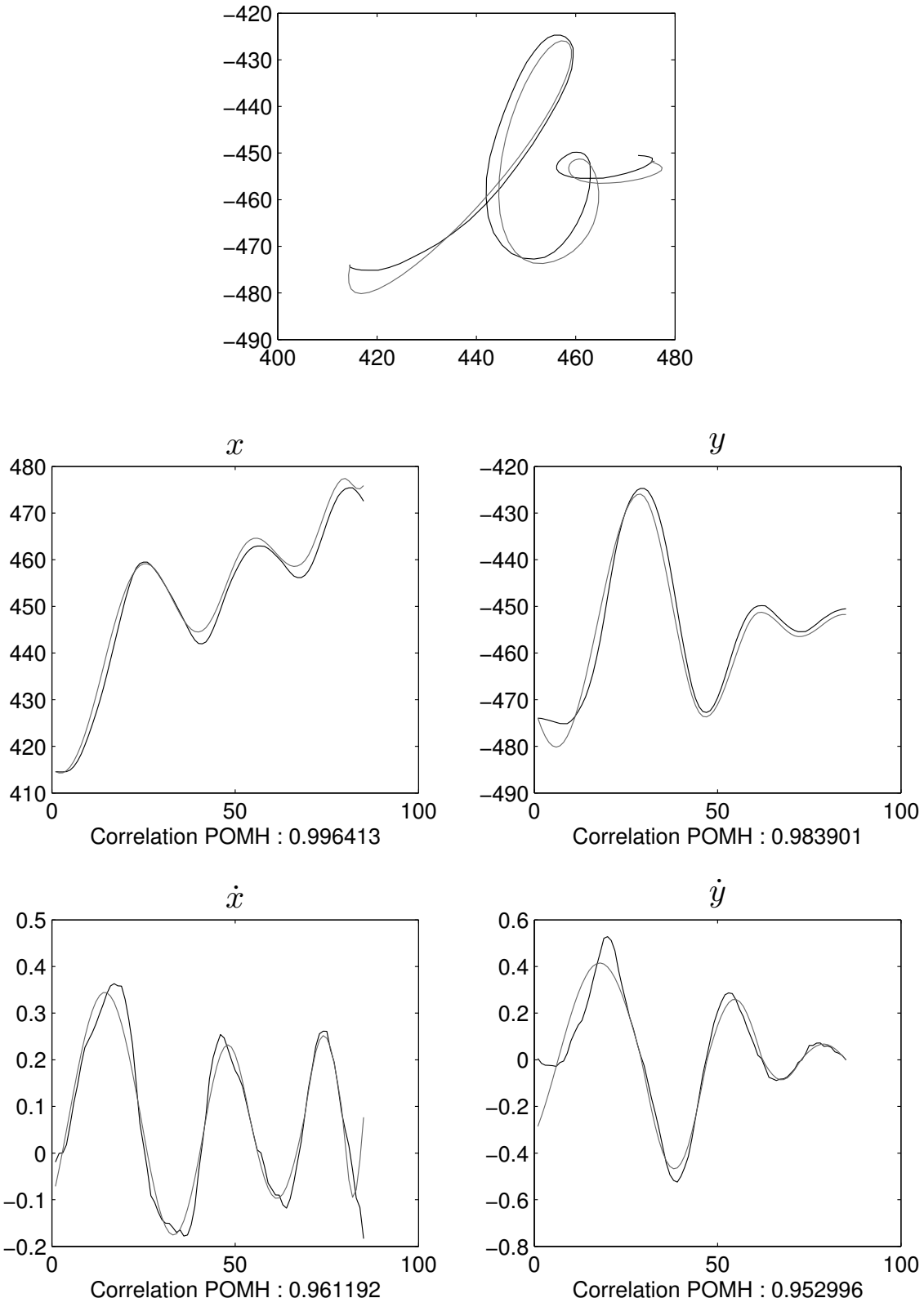


FIGURE 3.13 – Comparaison entre la trace originale en noir et la trace reconstruite en gris pour la lettre 'b'.

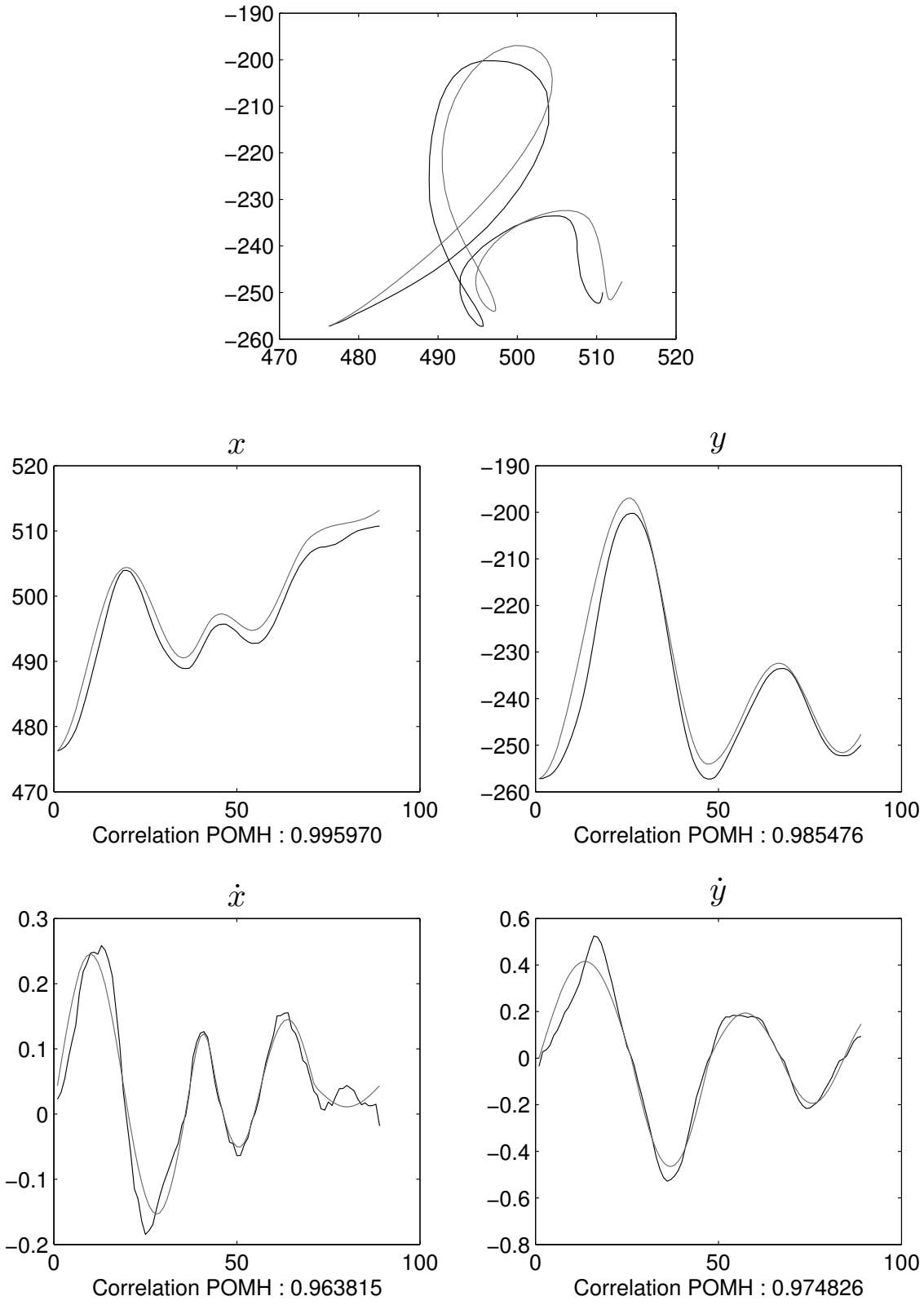


FIGURE 3.14 – Comparaison entre la trace originale en noir et la trace reconstruite en gris pour la lettre 'h'.

TABLE 3.2 – Corrélations moyennes (positions et vitesses) sur 35 traces pour chaque scripteur et chaque lettre.

	'b'				'h'				Moyenne
Participant	x	y	$\dot{x}$	$\dot{y}$	x	y	$\dot{x}$	$\dot{y}$	
David	0,997	0,969	0,970	0,941	0,946	0,979	0,867	0,955	0.953
Fabien	0,992	0,989	0,959	0,973	0,985	0,944	0,737	0,908	0.936
Sylvie	0,978	0,976	0,905	0,949	0,920	0,988	0,858	0,957	0.941
Vivianne	0,987	0,982	0,965	0,966	0,988	0,963	0,948	0,955	0.969
Moyenne	0,988	0,979	0,950	0,957	0,960	0,969	0,852	0,944	0.950

scripteur, 35 exemplaires sont choisis aléatoirement afin d'obtenir le même nombre de traces à étudier pour chaque cas.

Analyse des données L'analyse porte sur les traces sélectionnées. Pour chaque lettre et chaque scripteur on calcule les coefficients de corrélation pour les profils de position et de vitesse.

3.4.1.2 Résultats

Le tableau 3.2 présente les corrélations moyennes entre les profils de position et de vitesse des traces originales et des traces synthétisées par POMH.

La plupart des valeurs de corrélation se situent autour de 0.95. Les positions ont une corrélation un peu meilleure que les vitesses bien que ces premières soient calculées à partir de ces dernières. On peut en déduire qu'une erreur dans la dynamique de la trace peut ne pas être visible dans sa géométrie.

On remarque que pour trois scripteurs sur quatre, POMH a du mal a capturer le profil des vitesses horizontales pour la lettre 'h'. Plus généralement, les corrélations sont plus faible pour le 'h' que pour le 'b'.

3.4.2 Mots

Aborder la question de l'étude des mots demande en tout premier lieu de se poser la question des levers de crayon. En effet, lorsque l'on écrit un mot, il est plus que probable que celui-ci soit composé de plusieurs traces.

Afin d'appliquer POMH sur des mots, 3 stratégies ont été explorées :

- Obliger le scripteur à ne pas lever le crayon.
- Appliquer POMH à chaque trace constituant le mot, de manière indépendante.
- Enregistrer les positions du crayon même quand il ne touche pas la tablette, la trace devenant la trajectoire totale utilisée pour tracer le mot.

Le premier cas est le plus facile à mettre en œuvre. POMH est appliqué à une trace continue dont le début et la fin sont déterminés par le lever de crayon, rien ne change par rapport aux cas précédents. Cependant, obliger le scripteur à ne pas lever le crayon lorsqu'il écrit un mot l'empêche d'utiliser son écriture naturelle. Pire, cela peut entraîner des hésitations, des pauses, des déformations de lettres voire, cas extrême, des coupures de traces (s'il n'a pu s'empêcher de lever le crayon). Or les hésitations dans la trace originale sont un problème pour POMH. Il est possible de contourner ces problèmes en choisissant

TABLE 3.3 – Corrélations moyennes entre profils de position et de vitesse pour 100 mots.

	x	y	$\dot{x}$	$\dot{y}$
Moyennes	0.9983	0.9715	0.9261	0.9606
Écart-types	0.0064	0.0241	0.0432	0.0218

des mots qui les minimisent, par exemple, 'cellule' et 'umbrella' (figure 3.1) ne posent pas trop de problèmes au scripteur.

Le deuxième cas permet de corriger les erreurs de position à la fin de chaque trace constituant le mot (chaque trace est reconstruite à partir de son point de départ), ce qui est visuellement satisfaisant (voir figure 3.2). Cependant, cette méthode ne permet pas de bien décrire la dynamique du mot ; pire, l'hypothèse faite dans l'étape 2 de l'algorithme présenté en section 3.1.2.3 (les vitesse sont nulles en début et fin de trace) est clairement fausse.

Le troisième et dernier cas donne visuellement la moins bonne impression, en effet, POMH travaillant sur les vitesses, l'erreur entre la vitesse originale et la vitesse synthétisée a tendance à s'accumuler lors de la reconstruction de la trace. En revanche, la dynamique de la trace est préservée même lors des levers de crayon. Finalement, le point le plus problématique de cette méthode devient la détermination du début et de la fin d'un mot de manière automatique. Ceci doit être fait manuellement par l'expérimentateur lors de la saisie des traces.

La figure 3.15 montre un exemple de mot reconstruit par POMH ainsi que les valeurs de corrélation associées aux profils de position et de vitesse. Dans ce cas précis, le scripteur ne lève pas le crayon (nous sommes donc dans le cas 1), les corrélations sont toutes supérieures à 0.9, la vitesse horizontale étant comme souvent le facteur limitant.

Procédure Il a été demandé à un participant d'écrire 100 mots en anglais. Pour chaque mot, le participant ne devait pas lever le crayon.

Analyse des données Pour chaque mot on calcule les coefficients de corrélation pour les profils de position et de vitesse.

Résultats Le tableau 3.3 présente les corrélations moyennes (et les écarts-type) entre les profils de position et de vitesse des traces originales et des traces synthétisées par POMH. Malgré le fait que le scripteur ait hésité, les corrélations moyennes restent supérieures à 0.92, les vitesses horizontales se montrant les moins corrélées et exhibant la plus forte variabilité dans la corrélation.

3.4.3 Phrases et signatures

La figure 3.16 montre un exemple de signature reconstruite par POMH. Les corrélations sont supérieures à 0.97. L'inspection visuelle du résultat est satisfaisante.

Les considérations de la section précédente concernant les levers de crayon à l'intérieur d'un mot sont applicables à la phrase. Le premier cas où l'on oblige le scripteur à ne pas lever le crayon n'a bien sûr ici plus aucun sens. Il reste donc deux stratégies :

- considérer indépendamment toutes les traces écrites (entre chaque lever de crayon),
- considérer une phrase comme une unique grande trace et, dans ce cas, on doit enregistrer les positions quand le crayon est levé.

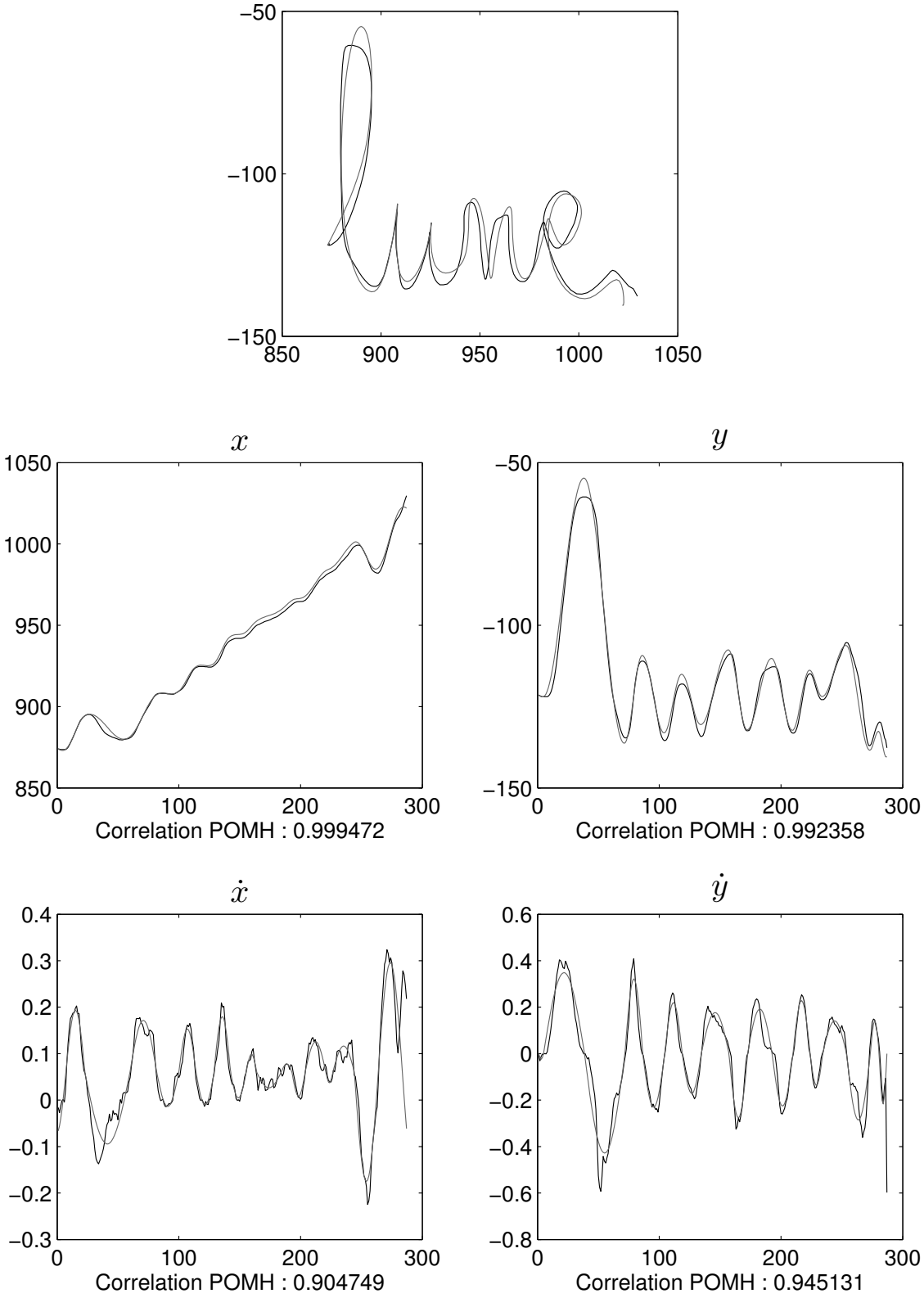


FIGURE 3.15 – Comparaison entre la trace enregistrée et la trace reconstruite par POMH pour le mot "lune". Cas où le scripteur ne lève pas le crayon.

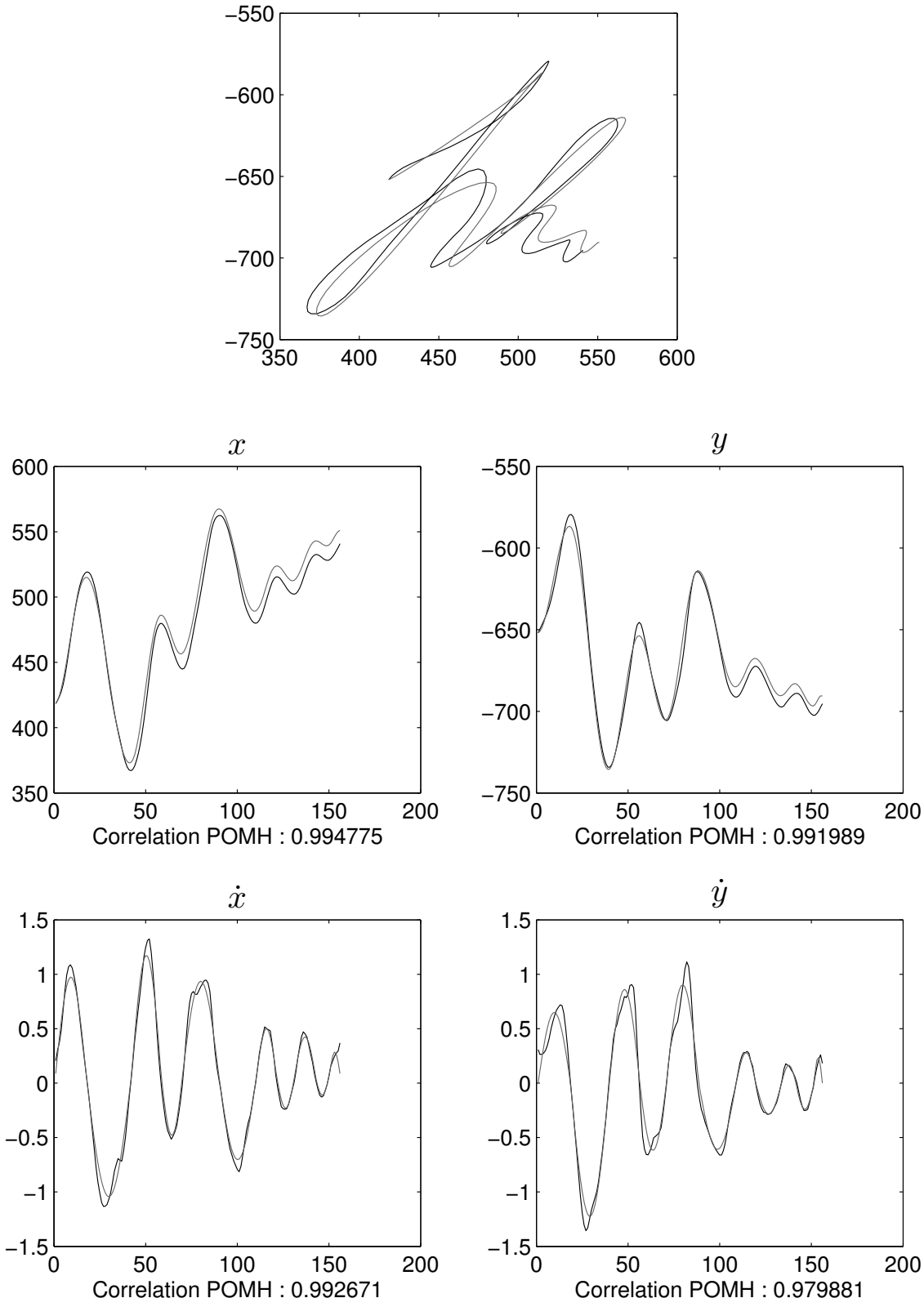


FIGURE 3.16 – Comparaison entre une trace originale (en noir) et cette même trace reconstruite par POMH (en gris) pour un cas particulier de signature.

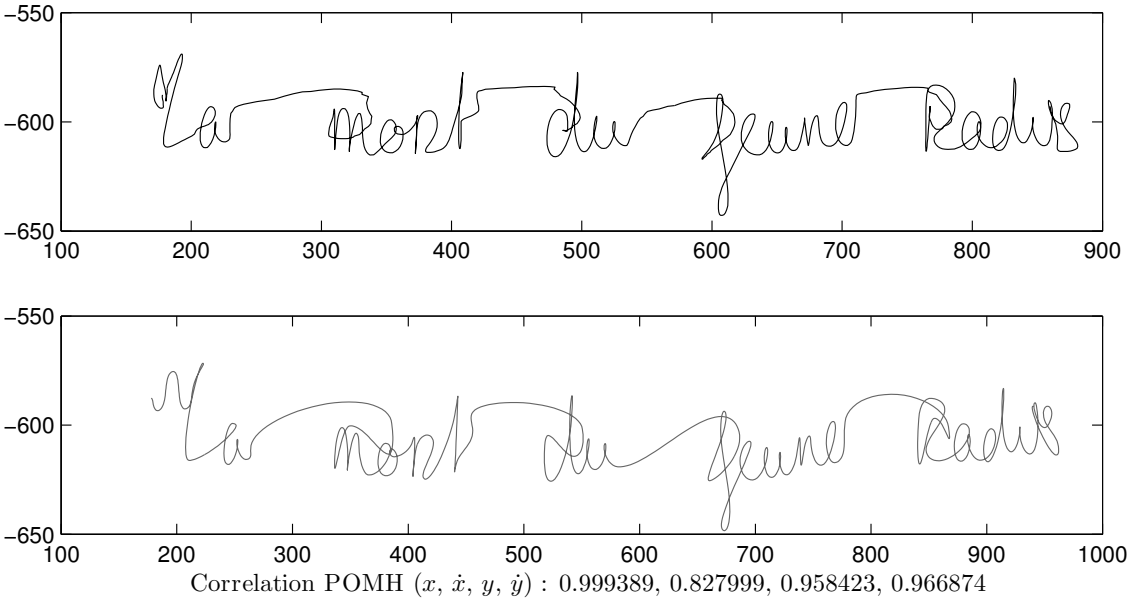


FIGURE 3.17 – Comparaison entre la trace originale et la trace reconstruite pour la phrase : "la mort du jeune radis". Les levers de crayon sont correctement intégrés par POMH.

La figure 3.2 montre un exemple de phrase ("Hopes are in the sky") étudiée par POMH avec la première stratégie. Le fait qu'à chaque début de trace (entre un poser et un lever de crayon) reconstruite, les positions soient réinitialisées, confère au résultat une impression visuelle très satisfaisante.

La figure 3.17 montre un exemple de phrase ("La mort du jeune radis") reconstruite par POMH en utilisant la deuxième stratégie. Les corrélations sont supérieures à 0.95 sauf pour les positions en y où la corrélation est de 0.83. On remarque que les traces reconstruites aux moments où le crayon est levé (entre chaque mot) ont tendance à être arrondies, ce qui pourrait expliquer ce coefficient de corrélation plus faible en y. La comparaison visuelle entre la trace originale et la trace reconstruite par POMH n'en reste cependant pas moins satisfaisante.

Il n'a pas été effectué d'étude statistique pour la signature et la phrase. Cependant, dans le cas de la phrase une tendance semble se détacher (en utilisant la seconde stratégie) : il y a une dérive progressive se traduisant par une élongation de la trace reconstruite par rapport à la trace originale, ce phénomène étant cumulatif avec la longueur de la trace initiale. Cette dérive est d'ailleurs visible sur la figure 3.17.

3.4.4 Autres langues et autres mouvements

POMH semble généralisable à d'autres langues et d'autres types de mouvements. On ne donne ici que des exemples visuels sans aucune évaluation quantitative. La figure 3.18 montre le comportement de POMH sur une phrase écrite en arabe. La figure 3.19 montre le comportement de POMH sur un idéogramme chinois. Dans ce dernier cas, les corrélations sont supérieures à 0.90.

FIGURE 3.18 – Comparaison entre la trace originale et la trace reconstruite d'un morceau de phrase écrite en arabe. Les accents ont été enlevés.

3.5 POMH : problèmes et améliorations possibles.

3.5.1 Axes et déplacements horizontal/vertical ?

Lorsque nous avons présenté POMH en début de chapitre, nous avons choisi les axes canoniques comme directions des deux oscillateurs modélisant le mouvement de l'écriture. Ce choix a été effectué par simplicité. Nous tentons ici d'aller plus loin en proposant d'autres axes pour ces oscillateurs. Par ailleurs, nous allons aussi étudier plus en détail l'influence de l'ajout d'une constante de déplacement selon la direction de l'écriture sur la performance du modèle.

Le choix des axes canoniques s'accorde bien avec le support habituel d'écriture, cependant il s'accorde mal avec les mouvements naturels des doigts et du poignet. Il s'accorde aussi mal avec la pente de l'écriture. Nous allons maintenant tester POMH sur des systèmes d'axes qui correspondent mieux à l'anatomie du scripteur et voir si cela permet à POMH d'avoir une représentation plus précise des traces qu'il représente. Pour cela, nous choisissons deux nouveaux types d'axes.

Le premier type d'axes est constitué de deux axes définis, pour l'un, par la diagonale naturelle que l'on trace naturellement (correspondant plus ou moins à un mouvement de poignet) et, pour l'autre, la perpendiculaire au premier (correspondant au mouvement des doigts tenant le crayon). Ces deux axes sont définis expérimentalement pour chaque scripteur auquel on demande de tracer ces deux diagonales (ils sont ensuite calculés par régression linéaire. Ce système d'axes est intéressant car il correspond grossièrement au mouvement de chacune des "deux" articulations impliquées dans l'écriture. On appelle ce repère le repère "naturel".

Le second type d'axes prend la direction de l'écriture comme premier axe (ici la direction horizontale, la même que dans le cas du système canonique) et la pente de l'écriture comme deuxième axe. Il est à noter que la pente de l'écriture n'est pas alignée avec la diagonale naturelle utilisée dans le repère précédent. Ce repère sera appelé le repère "concordant". La figure 3.20 montre le repère "naturel" et le repère "concordant" sur un exemple. Sur celui-ci, la pente de l'écriture n'est pas constante ; le repère concordant est alors déterminé par la pente moyenne.

Nous voulons aussi étudier l'effet du recentrage autour de zéro des profils de vitesse lors de l'extraction des paramètres (voir section 3.1.2.3). Cela revient à prendre en considération un déplacement constant qui s'ajoute au mouvement oscillatoire, comme le fait Hollerbach dans son modèle. On note c ce déplacement constant du plan.

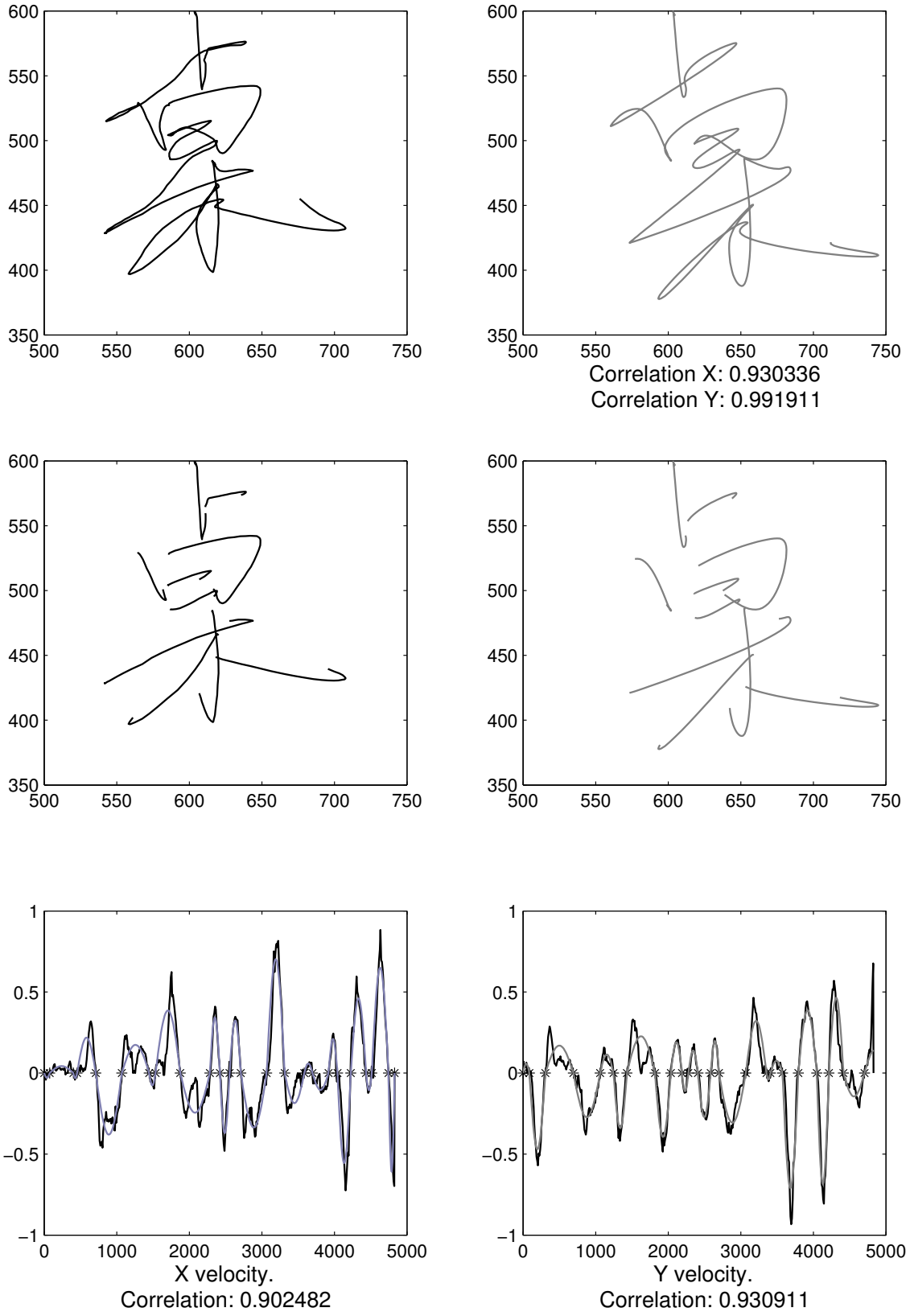


FIGURE 3.19 – Comparaison entre la trace originale et la trace reconstruite d'un idéogramme chinois. Le premier bandeau montre l'ensemble du mouvement réalisé pour la trace du idéogramme tandis que le deuxième montre la trace laissée par ce même mouvement sur le support. Le dernier bandeau présente les profils de vitesse en x et en y. En noir la trace originale et en gris la trace reconstruite.

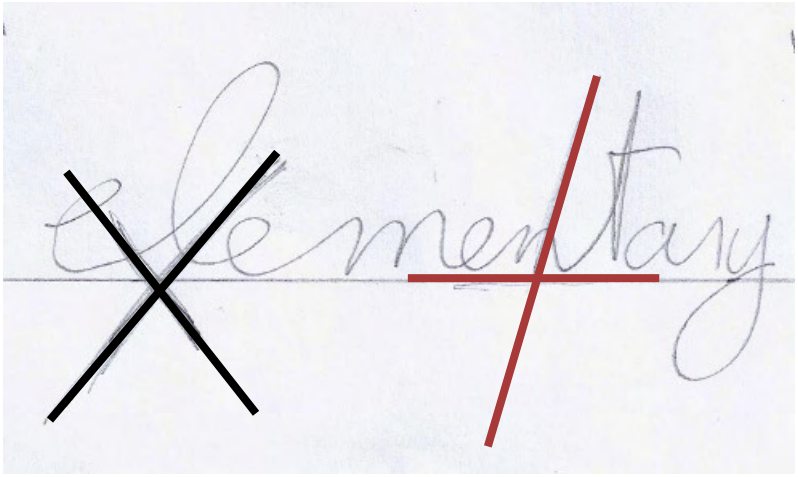


FIGURE 3.20 – En noir le repère "naturel", en rouge le repère "concordant".

TABLE 3.4 – Corrélations moyennes (positions et vitesses) pour 100 traces pour 3 types de repères différents en utilisant où non la constante de déplacement c.

Repère	x	y	$\dot{x}$	$\dot{y}$	Moyenne
avec déplacement horizontal constant ($c \neq 0$)					
Canonique	0.9983	0.9715	0.9261	0.9606	0.9641
Naturel	0.9980	0.9692	0.9126	0.9580	0.9595
Concordant	0.9984	0.9699	0.9197	0.9617	0.9624
sans déplacement horizontal constant ($c = 0$)					
Canonique	0.9959	0.9715	0.8629	0.9606	0.9477
Naturel	0.9974	0.9623	0.8856	0.9531	0.9496
Concordant	0.9933	0.9699	0.8519	0.9617	0.9442

Procédure Il a été demandé à un participant d'écrire 100 mots en anglais. Pour chaque mot, le participant ne devait pas lever le crayon. On lui a préalablement fait tracer sa diagonale naturelle et sa perpendiculaire.

Analyse des données Pour chaque enregistrement, POMH est appliqué dans chacun des repères. Puis, chaque résultat est comparé à la trace enregistrée (la comparaison s'effectue dans l'espace canonique, toujours en utilisant le coefficient de corrélation).

Résultats Le tableau 3.4 montre les résultats de la comparaison de l'application de POMH avec ou sans constante de déplacement (le c dans l'équation 2.18), pour trois repères différents. On remarque d'abord que la présence de la constante c permet d'améliorer la performance : la vitesse en x étant comme toujours le facteur limitant, il se trouve qu'elle est un peu meilleure quand c est présent.

En ce qui concerne les différents repères, le repère canonique semble le mieux adapté pour POMH puisqu'il donne des résultats un peu meilleurs que les deux autres repères. Pour ce scripteur il semble que le choix du repère n'a pas d'importance sur les performances de POMH. Ce scripteur n'a pas une pente d'écriture stable ni une écriture très penchée, il serait intéressant de voir avec d'autres scripteurs ayant des caractéristiques d'écriture différentes si le repère à toujours si peu d'importance.

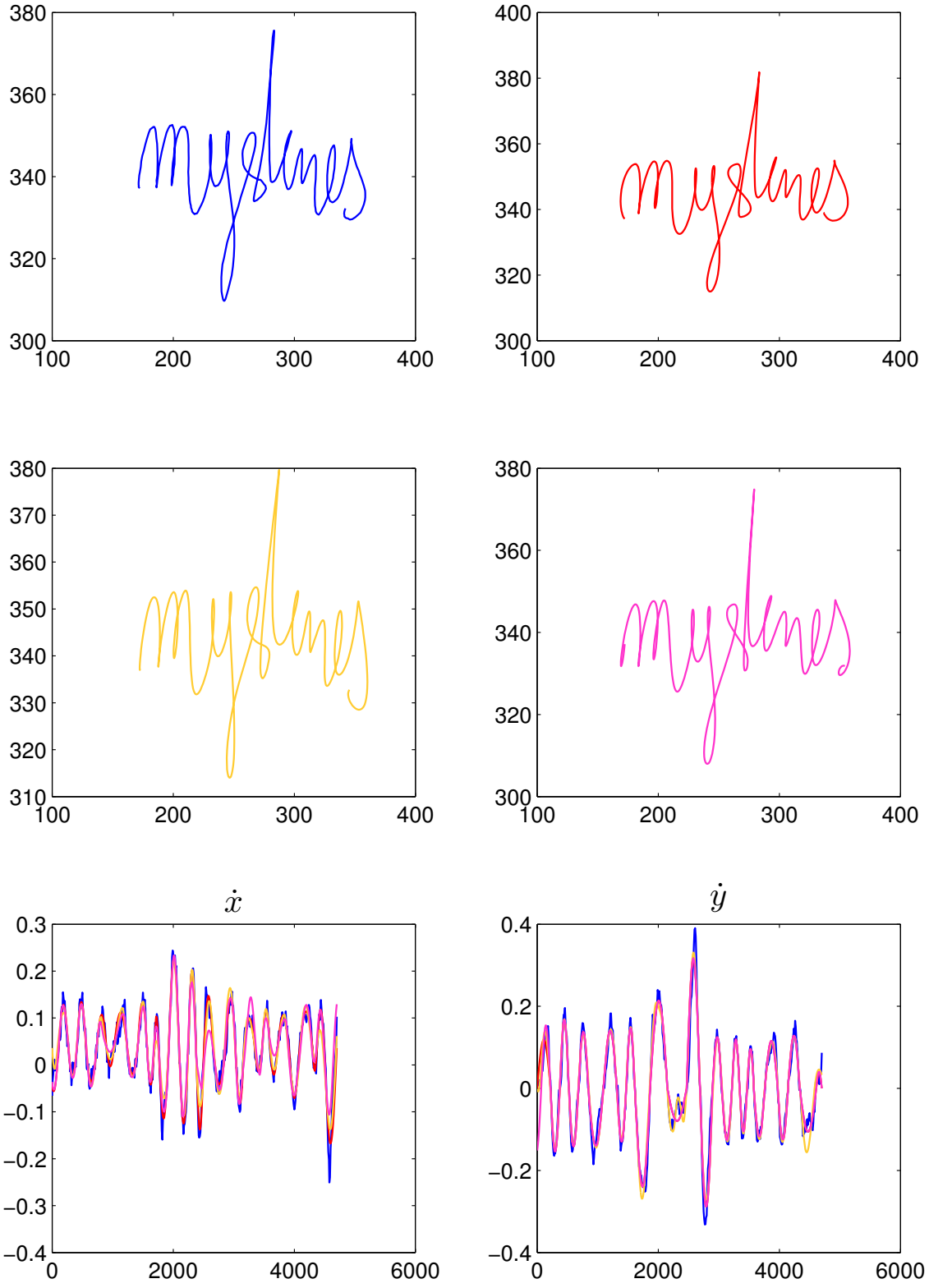


FIGURE 3.21 – Comparaison entre la trace originale en haut à gauche et les traces reconstruites par POMH appliqué en utilisant une constante de déplacement continu et en utilisant différents repères : le repère canonique (en haut à droite), le repère "naturel" (au milieu à gauche) et le repère "concordant" (au milieu à droite). En bas les vitesses en x et en y.

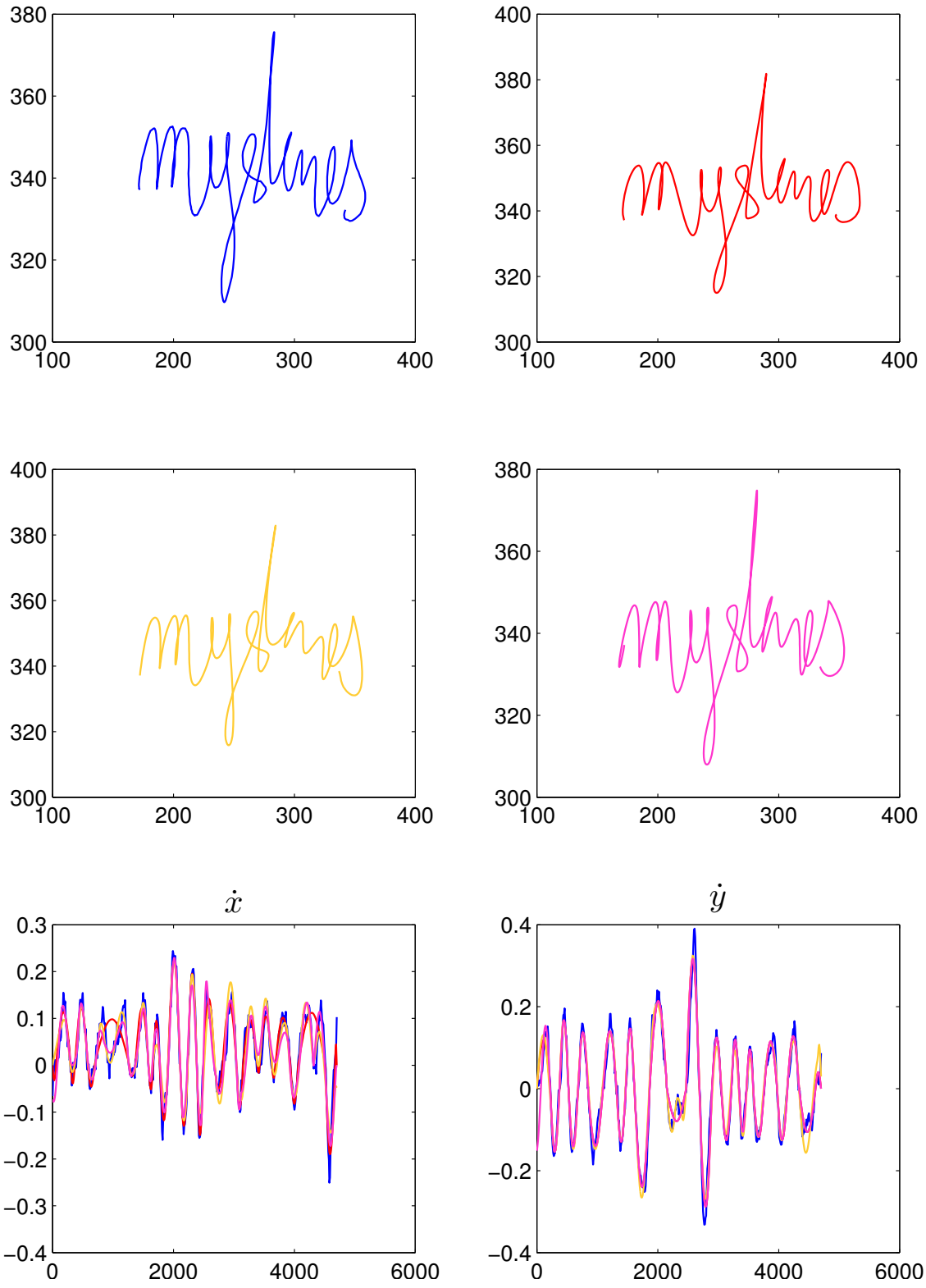


FIGURE 3.22 – Comparaison entre la trace originale en haut à gauche et les traces reconstruites par POMH appliqué sans utiliser de déplacement continu et en utilisant différents repères : le repère canonique(en haut à droite), le repère "naturel" (au milieu à gauche) et le repère "concordant" (au milieu à droite). En bas les vitesses en x et en y.

Les figures 3.21 et 3.22 montrent pour le mot "mysteries" les traces reconstruites par POMH en utilisant différents types d'axes et en utilisant ou non la constante de déplacement horizontale. On peut faire deux remarques. Premièrement, on voit que la trace originale est à peine lisible, surtout la fin du mot. Cela est dû au fait que l'on a imposé au scripteur d'écrire l'ensemble du mot sans lever le crayon alors qu'il n'en a pas l'habitude. Deuxièmement, il faut comprendre que l'effet de l'utilisation du déplacement constant, dans l'application de POMH, a pour but de déplacer le profil des vitesses horizontales vers le bas (si l'axe des x est aligné avec le déplacement horizontal). Si l'on regarde la trace rouge des figures étudiées, à la fin de la lettre 'm', on remarque que dans le cas où il n'y a pas de prise en compte de ce décalage vertical du profil de vitesse horizontal, la trace est déformée : on voit que dans ce cas POMH saute une période car il n'a pas détecté le zéro. Puisque prendre en compte le déplacement constant selon la direction horizontale revient à décaler le profil vers le bas, le zéro est alors correctement détecté et donc le problème disparait. Il s'agit d'un des problèmes majeurs de POMH : une oscillation de vitesse ne descendant pas suffisamment bas peut ne pas être détectée. La prise en compte de c permet de corriger nombre de ces problèmes mais cela n'est pas toujours suffisant. Plusieurs stratégies pourraient être envisageables ; on pourrait par exemple augmenter la constante de déplacement c de manière artificielle (en ne tenant plus compte de la quantité de déplacement réelle) mais judicieuse (sans en faire trop pour éviter le risque de ne plus détecter des zéros qui l'étaient auparavant). On pourrait aussi ajouter des zéros artificiels en utilisant le fait que les périodes semblent ne pas beaucoup varier.

3.5.2 Début et Fin

Ce problème à déjà été abordé précédemment mais nous le détaillons ici.

Dans l'algorithme 3.1.2.3 on suppose que les vitesses sont nulles en début et fin de trace. Comme nous l'avons déjà remarqué, cette hypothèse est au mieux une bonne approximation, au pire totalement fausse. Cette approximation est faite car il faut calculer la période et l'amplitude de la première demi-période, incomplète.

La figure 3.23 montre un exemple de profil de vitesse posant des problèmes à POMH. Au début, le profil reconstruit par POMH, en rouge, et le profil trace originale, en bleu, correspondent bien car cette dernière débute à une vitesse nulle. Par contre, à la fin, la trace se termine alors que sa vitesse est élevée. POMH suppose en revanche que la vitesse est nulle, d'où la différence de forme entre le profil original et le profil généré par POMH.

Afin de corriger ce problème, une idée explorée est de compléter la trace en ajoutant au début (resp. à la fin) le symétrique de la trace comprise entre le début (resp. la fin) et le premier (resp. le dernier) zéro. La courbe verte sur la figure 3.23 montre que cette méthode est très efficace si la vitesse est proche de son maximum (voir fin du profil) mais en revanche échoue totalement si la vitesse est proche de zéro (voir début du profil). En effet, dans ce dernier cas, le premier zéro n'est pas détecté à cause du filtrage et seul le deuxième zéro est pris en considération. POMH considère alors deux périodes comme une seule. Une analyse statistique non présentée ici a montré qu'en moyenne cette stratégie donne de moins bons résultats que la stratégie originale (consistant à supposer les vitesses nulles au début et à la fin).

Une autre méthode, qui n'a pas été implémentée, suppose que la demi-période incomplète a la même période que la période suivante (resp. précédente). On peut alors placer le zéro manquant au bon endroit au début (resp à la fin) du profil de vitesse. On complète alors celui-ci par interpolation. Cette méthode est nommée "le report de période". Il peut arriver que la période à reporter soit plus petite que la portion de période à compléter (fi-

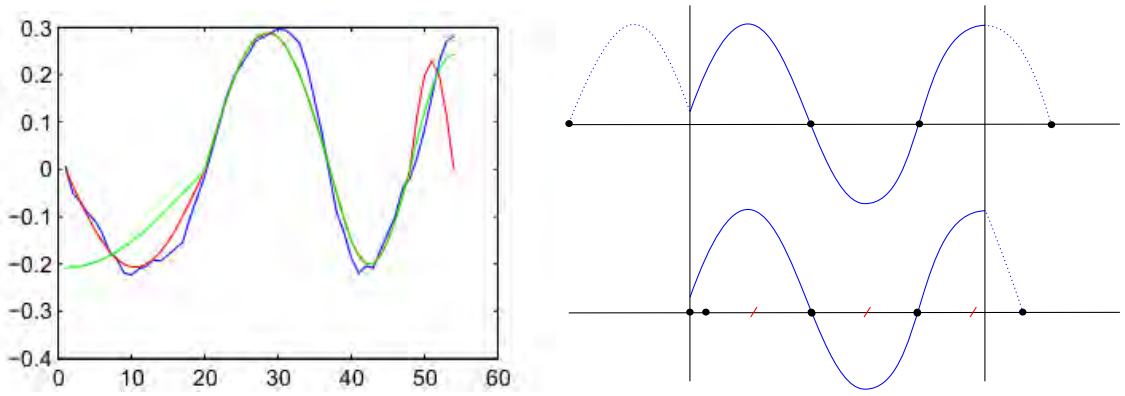


FIGURE 3.23 – A gauche, le profil des vitesses horizontales pour un oval. En bleu le profil original. En rouge le profil reconstruit par POMH appliqué au profil original auquel on a ajouté des zéros au début et à la fin. En vert le profil reconstruit par POMH appliqué au profil complété par symétrie. La figure de droite montre comment le profil original est complété pour les deux stratégies de complétion présentées dans le texte (en haut par symétrie, en bas par report de période).

gure 3.23, à droite en bas, au début du profil), dans ce cas on applique la stratégie initiale : on met juste un zéro au début du profil.

3.5.3 Trace hésitante

Nous l'avons vu, POMH parvient à reconstruire des traces de toutes complexités. Il y a cependant un cas où la reconstruction est mauvaise : le cas où le scripteur exécute un tracé hésitant (comme un enfant qui apprendrait à écrire). Nous n'entrerons pas dans les détails ici, cependant il pourrait être intéressant d'utiliser ce phénomène pour donner une mesure de la dysgraphie d'un individu. Des travaux complémentaires seraient à réaliser, mais, la corrélation moyenne entre une trace originale et sa reconstruction par POMH pourrait intervenir dans un test cherchant à évaluer la qualité de l'écriture d'un individu. Ce test pourrait venir en complément du test BHK, déjà utilisé actuellement, dans le diagnostic de la dysgraphie.

3.6 Variations autour de POMH

3.6.1 Forme des oscillations

Dans son modèle, Hollerbach [1981] préconise d'utiliser des oscillations harmoniques par simplicité, mais il affirme qu'on peut tout aussi bien les remplacer par d'autres formes. Cette remarque est pertinente si l'on se place du point de vue du mathématicien qui cherche à faire de l'ajustement de courbes sur les profils de vitesse : la forme s'adaptant le mieux donnera les meilleurs résultats lors de la comparaison avec la trace originale.

Pour montrer le fait que d'autres formes d'oscillations sont effectivement convenables, nous avons fait une comparaison entre trois type de forme d'oscillation : harmonique, triangulaire et parabolique. On s'attend à ce que l'oscillation triangulaire donne les moins bon résultats puisqu'elle ne peut pas suivre les arrondis des profils de vitesse.

Cent traces écrites par un scripteur ont été reconstruites par POMH en utilisant ces trois types de forme. Le tableau 3.5 montre que nos prédictions sont conformes, les oscillations triangulaires ont le moins bon résultat avec une corrélation moyenne de 0.91. La

TABLE 3.5 – Corrélations moyennes (positions et vitesses) pour 100 traces pour les 3 types de formes d'oscillation utilisées lors de la reconstruction : harmonique, triangulaire et parabolique.

Forme	x	y	$\dot{x}$	$\dot{y}$	Moyenne
Harmonique	0.9986	0.9728	0.9438	0.9691	0.9711
Triangulaire	0.9772	0.8285	0.8786	0.9425	0.9067
Parabolique	0.9704	0.9716	0.8937	0.9470	0.9457

reconstruction basée sur les formes harmoniques (donc POMH dans sa version originale) obtient les meilleurs résultats avec une corrélation moyenne de 0.97.

Chose étrange, dans le cas des oscillations à forme triangulaire, une bonne corrélation de la vitesse verticale aboutit à une mauvaise corrélation de la position verticale.

3.6.2 Application de POMH à des traces non naturelles.

Si POMH s'applique bien à des traces écrites avec une dynamique naturelle, qu'en est-il d'une trace qui aurait été réalisée par un agent non naturel (robot). Deux cas ont été testés, celui où cet agent a un mouvement à vitesse tangentielle constante et celui où il a une vitesse tangentielle aléatoire (ce dernier cas, peu réaliste en pratique, n'est étudié que par curiosité).

Ces deux types de trace sont calculés à partir de la géométrie d'une trace enregistrée, les points d'échantillonnage de la nouvelle trace (non naturelle) sont répartis sur la géométrie par calcul. La figure 3.24 montre deux nouvelles dynamiques calculées à partir d'une trace enregistrée. Sur le bandeau du bas, sont représentés les profils des vitesses verticales (en bleu) et leur reconstruction par POMH en (rouge).

Dans le cas où la vitesse tangentielle est constante, on remarque que le profil de vitesse est en forme de créneau. POMH produisant un profil de vitesse de forme harmonique, l'ajustement entre ces deux profils est donc médiocre. Si l'on superpose la géométrie de de ces traces avec la géométrie des traces reconstruites par POMH, le ressemblance est toujours évidente. Par contre si l'on joue temporellement l'écriture de ces traces, il y a des différences temporelles importantes entre les traces non naturelles originales et les traces reconstruites par POMH. POMH peut donc être utilisé afin de modifier la dynamique d'une trace et la rendre plus naturelle.

3.7 Discussion

Nous inspirant du travail original d'Hollerbach [1981], notre but était de fournir un modèle de l'écriture. Deux aspects devaient être pris en considération : déterminer quand et comment les paramètres du modèle étaient mis à jour et faire en sorte que le modèle soit suffisamment général pour rendre compte d'une grande diversité de traces manuscrites. Notre contribution principale repose sur la symétrie des équations 3.1 et 3.2 et sur la création d'un algorithme d'extraction des paramètres efficace, ne reposant pas sur des algorithmes d'optimisation coûteux. En particulier, le calcul de l'amplitude repose sur un résultat mathématique faisant intervenir la moyenne de la trace sur une demi-période. Nous discutons maintenant des propriétés mathématiques de notre modèle, de ses performances à reconstruire une trace manuscrite réelle, de sa plausibilité biologique et enfin des directions futures à donner à celui-ci.

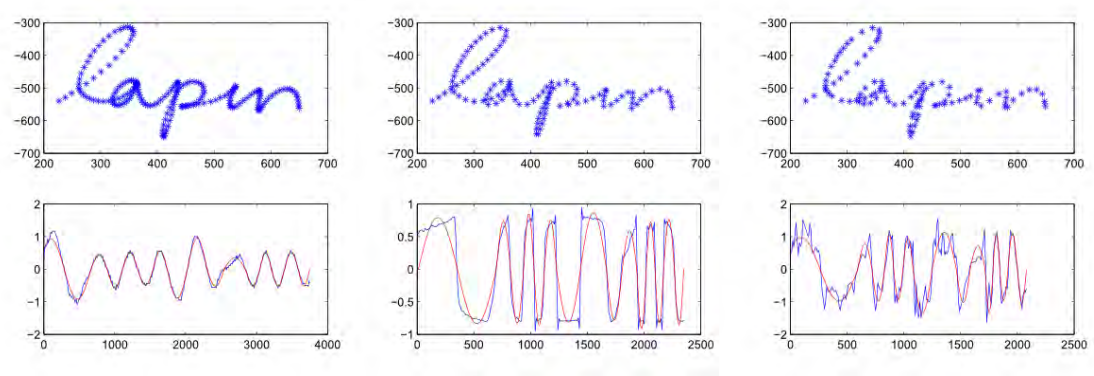


FIGURE 3.24 – A gauche la dynamique de la trace originale. Au milieu la trace avec une dynamique à vitesse tangentielle constante. A droite la même trace avec une dynamique oscillatoire. Les points sont temporellement espacés de 10ms. Sur le bandeau du bas le profil de vitesse verticale associé (en bleu) et la reconstruction par POMH (en rouge).

3.7.1 Un modèle simple et pertinent de l'écriture manuscrite

Pour que les mouvements en x et en y soient réellement oscillatoires, les paramètres associés aux oscillations ne doivent pas varier continûment (sinon, le choix d'une fonction harmonique comme support du modèle serait arbitraire). Ils ne doivent même pas changer trop souvent (sinon, il s'agirait d'un ajustement ad-hoc constitué d'une succession de petits bouts d'ellipses). Ainsi, sur chaque axe, la trace est définie, entre deux points d'annulation de la vitesse, par trois paramètres constants : l'amplitude, la phase et la période. Ces choix permettent une extraction en temps réel de ces paramètres en utilisant un ordinateur et une tablette graphique ordinaire.

La parcimonie de POMH est en partie due à sa symétrie. Contrairement à Hollerbach [1981], dont les paramètres du modèle changent, comme nous l'avons vu, aux moments d'annulation de la vitesse verticale, nous avons choisi de faire changer ces paramètres en x (resp. en y) aux moments d'annulation de la vitesse horizontale (resp. verticale). De fait, les axes x et y sont spatialement symétriques, et donc les oscillateurs associés sont fonctionnellement symétriques (et donc interchangeables). De là, un algorithme simple et unique, basé sur des opérations trigonométriques élémentaires, a pu être appliqué pour reconstruire les profils de vitesse de chacun des oscillateurs.

L'une des premières conséquences de cette symétrie est, pour POMH, la possibilité d'un transfert d'information entre les mises à jour successives des paramètres du modèle (temporellement parlant), constituant une sorte de mémoire partielle de la trace venant d'être écrite. Tandis que les paramètres sont extraits pour la trajectoire en x à un moment particulier, le modèle continue d'utiliser les paramètres ayant été extraits précédemment pour la trajectoire en y, et vice-versa. En conséquence, la trajectoire n'est pas produite comme une succession de morceaux de traces : les paramètres nouvellement extraits pour une dimension de la trace courante sont mélangés avec ceux hérités de la mise à jour précédente sur l'autre dimension. Un tel transfert d'information, sur la base du quart de cycle, est à l'origine de l'aspect lisse de la trajectoire reconstruite (ce que ne permet pas forcément le modèle d'Hollerbach). Cela pourrait aussi être un mécanisme au travers duquel se réalise la co-articulation entre morceaux de trace, un sujet qui reste ouvert dans la littérature consacrée à l'écriture manuscrite.

Une seconde conséquence est que la symétrie rend notre modèle insensible à l'inversion de x et de y. En fait, il n'y a pas nécessité d'un système d'axes fixé a priori. Dans le cas

présent, nous avons décidé d'utiliser le système d'axes canonique relativement à la feuille de papier, mais le modèle fonctionne avec n'importe quel autre système d'axes (nous en avons vu des exemples en section 3.5.1). De fait, POMH fonctionne mieux sur certaines traces avec un système de coordonnées ayant subit une rotation, voire même avec un système non-orthogonal.

Plus globalement, POMH met l'accent sur une leçon bien connue des systèmes dynamiques : des comportement complexes, ici les traces manuscrites, peuvent être générés grâce à des algorithmes étonnamment simples. Les propriétés du comportement moteur, telles que l'indépendance d'un système de coordonnées spatiales ou l'aspect lisse de la trace, sont fournis gratuitement, sans avoir été spécifiés dans le modèle.

3.7.2 Topologie de la trace et dynamique du mouvement

Afin d'évaluer les performances de POMH, nous l'avons comparé au modèle, bien établi et pourtant différent, d'Edelman et Flash [1987]. Utilisant les traces prototypiques proposées par ces auteurs, nos résultats statistiques ont montrés que POMH était capable de reproduire les traces générées par les humains aussi bien que leur modèle. Nos résultats ont été obtenus à partir de données brutes issues de l'acquisition, sans lissage (sauf pour l'extraction des zéros), contrairement à Edelman et Flash. Ces résultats ont montré que POMH était capable de gérer toute la diversité topologique des traces cursives (boucles, points de rebroussement, croisements, inflexions) qui peuvent être simplement mais directement décrits par des inversions de phase relative et des changements d'amplitude. De plus, le modèle POMH présente, par essence, la propriété de pouvoir reproduire exactement la cinématique du mouvement. Si l'on rejoue trace originale et trace reconstruite par le modèle, on observe les mêmes propriétés spatio-temporelles.

De plus, POMH présente un équilibre acceptable entre précision d'ajustement par rapport à la trace originale et nombre de paramètres utilisés (voir section 3.1.3). Tant dans la pratique que dans la théorie, il n'est pas toujours judicieux de modifier les modèles pour rendre compte de toutes les variations des traces produites par les humains [Burnham et Anderson, 2002]. Certaines variations ne sont que du bruit et n'apportent aucune informations sur le mouvement intrinsèque [Pitt et Myung, 2002]. Au delà du problème de surajustement, il faut garder à l'esprit que le comportement humain, et plus particulièrement le comportement moteur, exhibe une variabilité irréductible : deux trajectoires ne sont jamais produites de manière identique. On peut alors défendre le fait que la reproduction d'une trajectoire par un modèle mathématique n'a pas à être plus précise qu'une reproduction de trajectoire par un humain. Nos résultats semblent être dans la zone de variabilité attendue d'un scripteur humain.

3.7.3 Plausibilité biologique

En plus de sa symétrie, sa parcimonie et son efficacité, une quatrième propriété de POMH est sa plausibilité biologique. L'hypothèse d'un mouvement basique oscillatoire adoptée ici est en accord avec les travaux de Dooijes [1983] et Kunesch, Binkofski, et Freund [1989] sur le spectre fréquentiel du mouvement de la mine de crayon observé dans un repère cartésien. De plus, malgré ses spécificités, l'écriture manuscrite suit les même lois que d'autres mouvements cycliques et continus, dont la dynamique est capturée par des modèles oscillatoires [Athènes et al., 2004].

Par ailleurs, POMH exhibe deux autres propriétés de l'écriture manuscrite humaine. Premièrement, comme nous l'avons déjà souligné, en extrayant les paramètres indépendamment sur x et sur y, de sorte que les paramètres sont renouvelés à tour de rôle, le modèle

traduit l'une des propriétés biologiques les plus documentée du mouvement humain : la co-articulation. Chez les humains, la dynamique d'une trace en cours de génération est influencée par ce qui a été tracé précédemment et par ce qui sera tracé après [Thomassen et Schomaker, 1986]. Deuxièmement, l'indépendance de POMH vis-à-vis d'un système de coordonnées spatiales, rend compte d'une deuxième propriété prééminente du mouvement humain : l'équivalence motrice [Lashley, 1942; Teulings, 1996]. L'équivalence motrice est la faculté d'obtenir le même mouvement quelle que soit la configuration articulaire. La redondance du système moteur, qui permet l'équivalence motrice, interdit une correspondance directe entre l'effecteur mobilisé et le système de référence (de contrôle). Ainsi, les humains peuvent changer l'orientation de la feuille de papier ou changer leur manière de tenir le crayon [Sassoon, 1993]. Ils peuvent aussi générer la même trace en utilisant leur épaule, leur coude, leur poignet, leurs doigts, ou même leur cou, leur pied ou leurs dents [Bernstein, 1935].

Cette propriété suggère que le mouvement est spécifié par le système nerveux central dans un espace plus abstrait que les espaces liés aux effecteurs. Avec POMH, les traces sont reconstruites en utilisant des phases, des amplitudes et des pulsations, qui sont des paramètres relativement abstraits du mouvement oscillatoire et qui ne tiennent pas compte des propriétés ni de l'espace, ni de l'appareil moteur. Il est important de souligner que l'équivalence motrice se manifeste principalement sur des trajectoires assez longues pour montrer des propriétés topologiques. Il est donc crucial que POMH puisse reconstruire de longues traces, telles que de vrais mots ou de vrais phrases, en plus d'être capable de reproduire des traces simples et des traces prototypiques.

Les diverses tâches auxquelles font appel l'écriture manuscrite requièrent des mécanismes capables de générer des mouvements continus et des mouvements discrets [Guiard, 1993, 1997]. Au niveau de la sortie motrice, les fonctions harmoniques permettent à POMH de pouvoir produire aussi bien des traces contenant des arrêts et des points dégénérés que des mouvements continus et harmoniques tel que les gribouillis. POMH générant les trajectoires en mettant à jour les paramètres de manière intermittente, les trajectoires ne sont pas vues comme une concaténation de mouvements discrets. Cette vision semble confirmée par des preuves empiriques, basées sur les propriétés dynamiques, cinématiques, topologiques et neuronales des mouvements discrets et continus [Buchanan, Park, et Shea, 2006; Guiard, 1993; Huys, Studenka, Rheaume, Zelaznik, et Jirsa, 2008; Jirsa et Kelso, 2005; van Mourik et Beek, 2004; Perdikis, Huys, et Jirsa, 2011].

Souvent, les mouvements continus sont vus comme la concaténation de mouvements basiques et discrets. La production de mouvements discrets impliquerait moins de "calculs" que la production des mouvements continus et serait moins sensible aux perturbations que cette dernière. Pourtant des études neurobiologiques contredisent cette vision. Utilisant la technique fMRI, Schaal, Sternad, Osu, et Kawato [2004] ont observé que contrairement aux mouvements continus qui n'utilisent que les aires primaires du cerveau, les mouvements discrets font appel à nombre d'autres régions, dont le cervelet. Par ailleurs, Spencer, Zelaznik, Diedrichsen, et Ivry [2003] ont montré que des patients présentant des lésions du cervelet, sont handicapés pour produire des mouvements discrets mais pas pour produire des mouvements continus. Plutôt que d'être considérés comme une base servant à la construction de mouvements complets, les mouvements discrets devraient plutôt être considérés comme un cas limite de mouvements oscillatoires (par exemple comme des mouvements oscillatoires interrompus) [Schöner, 1990], ou comme une catégorie à part de mouvements, disjointe des mouvements continus [Jirsa et Kelso, 2005].

Extraire les paramètres aux moments d'annulation de la vitesse est aussi pertinent. En effet, des études sur un mouvement oscillatoire humain ont montré qu'à l'intérieur d'une période, il y a deux points spécifiques, localisés aux environs des moments d'inversion du

mouvement [Beek, 1989; Byblow, Carson, et Goodman, 1994]. En ces points, un resserrement local des trajectoires dans l'espace des phases laisse supposer que l'information relative au mouvement y est disponible et permet de l'organiser [Fink, Jirsa, Foo, et Kelso, 2000]. Kostrubiec, Soppelsa, Albaret, et Zanone [2011] ont montré que les patrons de coordination bimanuels, nouveaux pour un individu et d'abord difficiles à réaliser, étaient facilités lorsque les effecteurs étaient mis à la bonne position en ces points spécifiques. Ces découvertes tendent toutes à montrer que les caractéristiques globales d'un cycle du mouvement sont définies aux points où la vitesse s'annule.

Chapitre 4

Dynamique de l'écriture

Au chapitre précédent, nous avons proposé un modèle oscillatoire de l'écriture. Ce modèle se base sur des équations faisant intervenir six paramètres évoluant de façon constante par morceaux au cours du temps.

Le but de ce chapitre est de tenter de comprendre les interactions entre les deux oscillateurs du modèle oscillatoire de l'écriture. En particulier, nous voulons comprendre si tous les tracés sont possibles, avec autant de facilité, ou au contraire, si un sous-ensemble de traces est privilégié.

Plusieurs travaux ont, par le passé, observé l'existence de contraintes dans des mouvements oscillatoires [Haken, 1983; Haken et al., 1985; Athènes et al., 2004; Sallagoïty et al., 2004]. En s'aidant de l'approche dynamique de la motricité, ils ont pu découvrir et/ou expliquer certaines coordinations préférentielles dans ces mouvements. Nous basant nous aussi sur cette approche, présentée en première section, nous décrivons une expérience ayant pour but d'étendre ces travaux à l'écriture manuscrite. Enfin, nous exposerons des hypothèses sur l'impact de la vitesse de production d'une trace sur sa géométrie et donnerons des protocoles d'expérimentation permettant de tester celles-ci.

4.1 Dynamique de coordination motrice

4.1.1 Approche dynamique de la motricité

La présentation qui suit, de l'approche dynamique de la motricité, résume celle faite par Danna aux sections 2.1 et 2.2 de son manuscrit de thèse [Danna, 2011]. Nous la complétons avec des éléments du cours d'introduction de Delignières [2004].

Cette approche s'appuie sur la théorie des systèmes dynamiques et en particulier sur les systèmes complexes. Un système dynamique est défini comme un système qui évolue au cours du temps de manière déterministe (c'est à dire que la suite des états futurs est entièrement déterminée par l'état présent). Un système complexe est constitué d'un grand nombre d'éléments en interaction. Ces interactions et ces éléments étant trop nombreux pour pouvoir être étudiés individuellement, on s'attache à étudier leur comportement global.

Cette étude globale conduit à rechercher l'émergence de configurations (spatio-temporelle) macroscopiques du système, prédictibles et spontanées issues de la multitude des interactions du système. A ces configurations globales, appelées patrons préférentiels, on associe une mesure de stabilité qui décrit la capacité du système à reformer spontanément le patron duquel une perturbation l'a éloigné [Haken, 1983].

Afin de rendre compte du comportement global du système, c'est à dire des configurations macroscopiques précédemment évoquées, on utilise des variables dites collectives

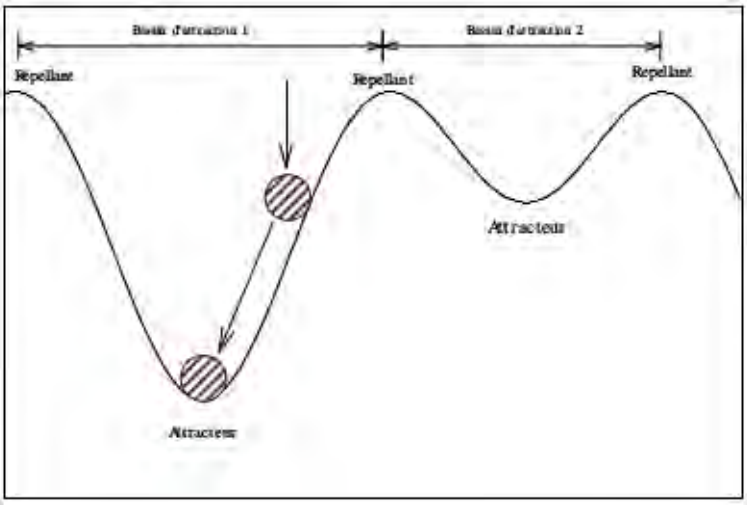


Figure 4.1 – Paysage des attracteurs. Si on suppose que l'état du système est représenté par une bille sur un profil de ce type, alors l'évolution du système dynamique peut être visualisée comme le mouvement de la bille soumise à l'action de la gravité. Les vallées représentent donc les attracteurs et les bosses les répulseurs. Image empruntée à Delignières [2004].

.

encore appelées paramètres d'ordre. Ces variables sont construites par le théoricien pour décrire le comportement global du système. Dans le cas où l'on étudie la coordination entre deux oscillateurs, il est usuel d'avoir recours à la différence de phase.

On appelle attracteur, la configuration adoptée par le système sous l'influence de contraintes internes et externes. Un attracteur est caractérisé par sa stabilité. Par opposition on appelle répulseur, les configurations non stables. Le paysage des attracteurs permet de visualiser la stabilité d'une configuration (coordination dans le cas de deux oscillateurs) ou au contraire son instabilité. La figure 4.1 montre le lien entre paramètre d'ordre, évolution du système, attracteurs et répulseurs.

Un facteur ou une contrainte pouvant modifier la topologie du paysage des attracteurs (ie. supprimant ou ajoutant des attracteurs) est appelé paramètre de contrôle. Si le système se trouve dans un état stable, et qu'un paramètre de contrôle est modifié et entraine la disparition de l'attracteur associé, alors le système va spontanément évoluer vers un état associé à un autre attracteur. On parle de transition de phase.

4.1.2 Exemple : coordination bimanuelle

Afin de mieux comprendre l'approche dynamique de la motricité nous allons présenter l'exemple de la coordination bimanuelle étudiée par cette approche dans les travaux suivants [Kelso, Holt, Rubin, et Kugler, 1981; Kelso, 1984].

Ces travaux s'appuient sur une expérience où l'on a demandé aux participants de réaliser des oscillations rythmiques avec les indexes de chaque main. Il a été observé que deux patrons préférentiels de coordination sont alors possibles et sont associés à deux phases relatives correspondantes : 0 ° (mouvement en phase, les doigts bougent de la même façon) et 180 ° (mouvement en antiphase, les doigts ont un mouvement opposé). Puis, lorsque les sujets sont en mode antiphase, on leur demande de progressivement augmenter la vitesse d'oscillation. Les oscillations deviennent de plus en plus instables (avec une forte variabilité

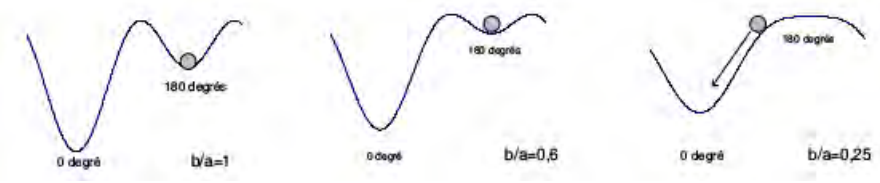


FIGURE 4.2 – Paysage des attracteurs en fonction du rapport $\frac{b}{a}$ pour la fonction potentielle V. Images empruntées à Delignières [2004].

.

de la phase relative observée) puis une transition de phase survient spontanément vers le patron de coordination en phase.

L'interprétation faite par Kelso de ce phénomène est que la vitesse d'oscillation, qui serait une variable de contrôle du système dynamique constitué par les deux doigts oscillants, en augmentant change le paysage des attracteurs en diminuant la force de l'attracteur associé à une coordination à 180 ° jusqu'à sa suppression. Ces phénomènes conforteraient l'approche dynamique de la coordination et montreraient l'existence d'un couplage entre les deux doigts.

Pour asseoir un peu plus ces hypothèses, une formalisation de ces travaux a été donnée dans Haken et al. [1985]. L'évolution de la variable collective est décrite par la fonction potentielle V (qui décrit l'évolution de ϕ, la phase relative) qui est posée comme :

$$V(\phi) = -a\cos(\phi) - b\cos(2\phi) \tag{4.1}$$

La figure 4.2 montre différentes coupes du paysage défini par V en fonction du rapport $\frac{b}{a}$. On remarque que plus le rapport diminue, plus le patron de coordination à 180 ° devient instable. On associe donc l'augmentation de la vitesse du mouvement à une diminution de ce rapport.

4.2 Dynamique de coordination grapho-motrice

Dans cette section nous présentons les trois travaux qui sont à l'origine de notre démarche. Ces travaux ont montré l'existence de patrons de coordination préférentiels dans le tracé d'ellipses. Ils ont par ailleurs étudié la stabilité de ces patrons en regardant l'impact de la vitesse sur ceux-ci.

4.2.1 Travaux d'Athènes et al. 2004

Les travaux présentés par Athènes et al. [2004] ont tenté de mettre en évidence l'existence de patrons de coordination préférentiels lors de la production de traces elliptiques. Pour ce faire, les auteurs ont utilisé la méthode dite du "scanning" [Tuller et Kelso, 1989; Yamanishi, Kawato, et Suzuki, 1980; Zanone et Kelso, 1992] afin de montrer l'existence d'attracteurs.

Il a été demandé à 13 participants droitiers de reproduire 13 formes correspondant à 13 patrons de coordination. Le mouvement est ici vu comme la résultante de deux oscillateurs harmoniques vibrant selon x et y. La différence de phase entre chacun de ces oscillateurs est appelée phase relative (PR). A chaque PR est associé un patron de coordination, les 13 formes correspondant à des PR allant de 0 ° à 180 ° par pas de 15 °.

Ils ont aussi étudié l'amplitude relative (AR) par une série de 13 formes constituées de 6 ellipses orientées verticalement (avec une amplitude horizontale variant de 0 à 1

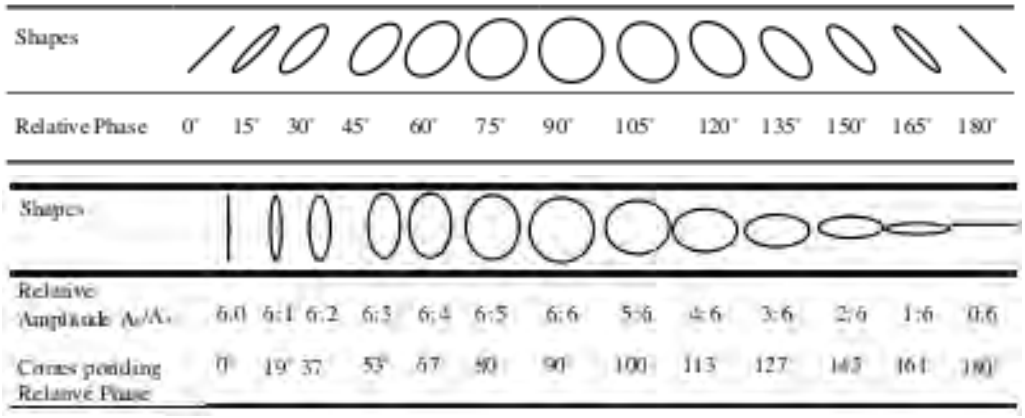


Figure 4.3 – Les deux séries de forme utilisées par Athènes et al. [2004] pour l'expérience s'appuyant sur la méthode du "scanning". Images empruntées à Athènes et al. [2004].

par rapport à l'amplitude verticale), d'un cercle et de 6 ellipses orientées horizontalement (avec une amplitude verticale variant de 1 à 0 par rapport à l'amplitude horizontale). Une rotation des formes ainsi obtenues de 45 ° permet d'avoir une équivalence entre les deux séries et d'associer une PR à chaque AR.

La figure 4.3 montre les deux séries de forme ainsi que leurs phases relatives associées. A chaque essai, une de ces formes était affichée sur la tablette et le participant devait la reproduire par superposition.

L'erreur et la variabilité de la PR produite, ainsi que la fréquence spontanée de production de la trace ont été étudiées. Les résultats obtenus sont montrés en figure 4.4.

Ces profils d'erreurs ont révélé l'existence de quatre patrons préférentiels à 0 °, 45 °, 125 ° et 180 ° dont les formes correspondantes sont reproduites avec une plus grande précision et moins de variabilité. La pente négative observée, sur le profil d'erreur constante, pour les patrons 45 ° et 125 ° montre un pouvoir d'attraction de ceux-ci. En revanche, une pente positive pour 0 ° et 180 ° semble montrer que ces patrons sont répulsifs bien qu'ils soient attractifs sur le plan moteur. Ce phénomène pourrait s'expliquer par une discrimination visuelle des formes associées ayant un impact très négatif sur l'attraction de ces états. En ce qui concerne la fréquence de tracé, elle semble diminuer avec l'augmentation de la phase relative.

4.2.2 Travaux de Sallagoïty et al. 2004

Afin d'aller plus loin dans ce sens et pour tester le degré de stabilité des différents patrons préférentiels de coordination, Sallagoïty et al. [2004] ont étudié l'impact de la vitesse de production de la trace sur ces patrons préférentiels qui, en fonction de leur stabilité, devraient disparaître plus ou moins rapidement selon la vitesse de tracé des formes étudiées. Par ailleurs, l'effet de l'apprentissage a été mesuré en comparant la main dominante à l'autre main. Si les mêmes effets sont observés sur les deux mains, alors ces patrons stables seraient indépendants de l'apprentissage de l'écriture, ou du moins, de l'effecteur utilisé pour cet apprentissage.

Quatre conditions ont donc été étudiées : RHS (Main droite, vitesse spontanée), RHF (Main droite, vitesse rapide), LHS (Main gauche, vitesse spontanée) et LHF (Main gauche, vitesse rapide). La figure 4.5 montre les résultats obtenus. Les travaux de Athènes et al. [2004] avaient mis en évidence l'existence de quatre patrons de coordination stables pour le

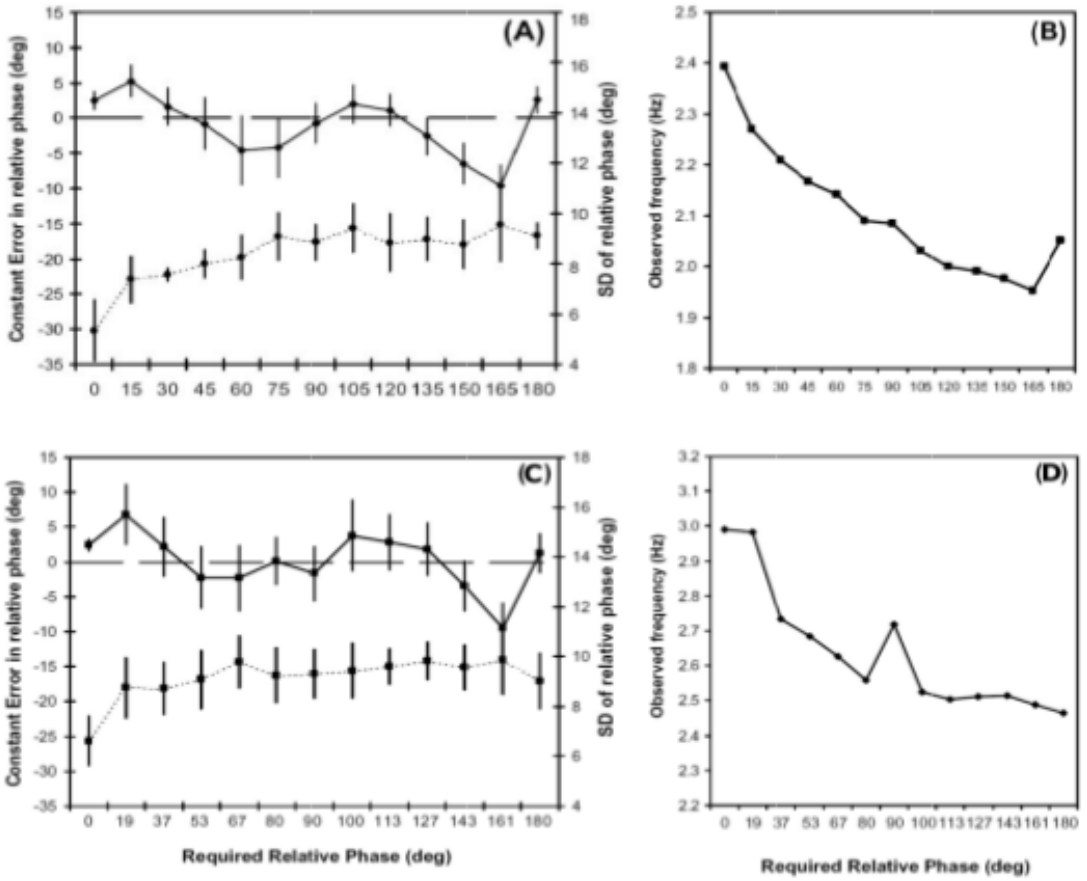


FIGURE 4.4 – Résultat du "scanning" de la PR en haut et de l'AR en bas. L'erreur en PR est représentée en continu et l'écart-type en pointillés. Images empruntées à Athènes et al. [2004].

tracé d'ellipse à vitesse spontanée correspondant, en gros, au PR 0 °, 45 °, 120 ° et 180 ° ; le moins stable étant le patron associé à 120 °. Dans les présents travaux, on remarque que pour la main gauche, ce patron de coordination associé à 120 ° disparaît. Cela laisse supposer qu'il n'y a que trois états stables intrinsèquement liés aux propriétés neuro-biomécaniques du corps et en particulier du bras et de la main. Le quatrième état, le moins stable, peut avoir été acquis par l'exercice et l'apprentissage, pour un effecteur particulier. Cet état, moins naturel, serait donc moins stable s'il est soumis à plus de contraintes. La comparaison entre tracés d'ellipses à vitesse spontanée montre aussi la disparition de ce patron sous l'effet de la contrainte vitesse, et semble donc confirmer ces hypothèses.

4.2.3 Travaux de Danna et al. 2011

Pour mieux comprendre l'évolution de la dynamique de l'écriture, Danna et al. [2011] ont étudié la mise en place de ces patrons préférentiels de coordination au cours de l'enfance. Danna et al. [2011] ont par ailleurs inspecté plus en détails l'influence de l'excentricité et de l'orientation de l'ellipse à reproduire sur la dynamique de coordination ; nous ne développerons pas ce point ici.

La même expérience de reproduction d'ellipses initiée par Athènes et al. [2004], a été effectuée sur des enfants du CP au CM2. Le premier résultat est que les enfants reproduisent trois patrons de manière préférentielle : 0 °, 90 ° et 180 °, contrairement aux adultes qui

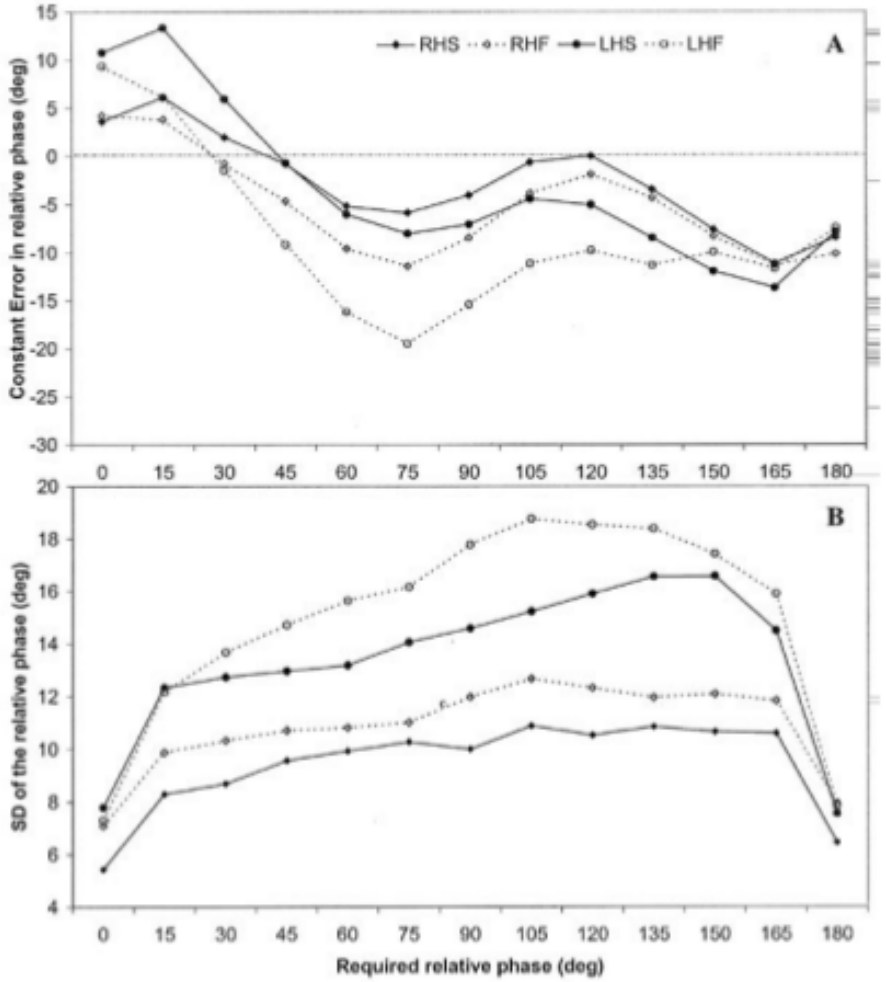


Figure 4.5 – Résultat du "scanning" de la PR. En haut, l'erreur en phase relative. En bas, l'écart-type associé. Images tirées de Sallagoïty et al. [2004].

.

reproduisent quatre patrons préférentiels, comme vu précédemment. Ainsi, au moins jusqu'à 11 ans, les enfants tendent à reproduire plus facilement les formes les plus arrondies quand les adultes sont plus efficaces à reproduire des formes elliptiques associées à une PR de 45 °, 120 ° (voir figure 4.3, pour un rappel des formes associées à chaque PR). Danna et al. [2011] proposent alors une extension de la fonction potentielle V (equation 4.1) permettant de rendre compte de ces différences entre enfants et adultes. Le paysage de V est représenté en figure 4.6 où l'évolution du paysage des patrons préférentiels au cours de l'apprentissage semble déterminant.

Les deux autres observations remarquables de ces travaux sont, premièrement, que les enfants ne sont pas affectés par l'orientation des ellipses à reproduire (contrairement aux adultes [Danna, 2011; Danna, Wamain, Kostrubiec, Tallet, et Zanone, 2010]) et que, par ailleurs, les enfants ont une fréquence de mouvement beaucoup plus faible que les adultes, à savoir : 0.7Hz. contre 2Hz.

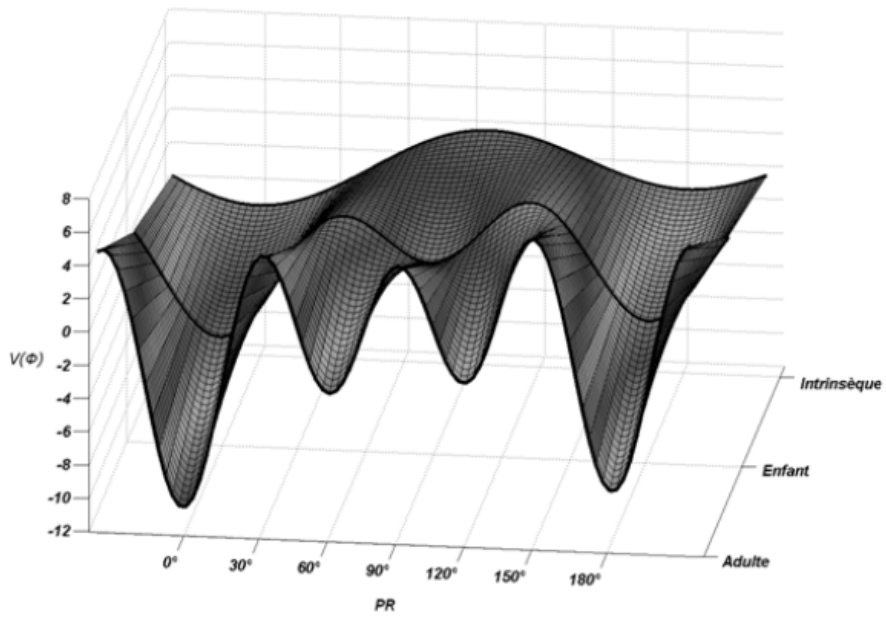


FIGURE 4.6 – Paysage de la fonction potentielle proposée par Danna. Image empruntée à Danna et al. [2011].

.

4.3 Dynamique de coordination de l'écriture

Les travaux présentés en section précédente n'étudient pas la production de l'écriture en tant que telle, mais la génération de formes elliptiques. Notre travail a pour but de retrouver ces résultats sur de l'écriture réelle.

Dans un premier temps, nous passons en revue les méthodes de calcul de la phase relative. Puis, retenant trois d'entre elles, étudiant la dynamique de coordination d'une trace, nous nous intéressons alors à la distribution des phases relatives calculées en chaque point de la trace. Nous montrons enfin au moyen d'une petite expérimentation que l'étude de cette distribution est pertinente en vue d'une expérience plus large présentée en section suivante.

4.3.1 Calcul de la Phase Relative

En modélisant l'écriture au chapitre 3, nous avons cherché à trouver des fonctions $a_{x/y}$, $\omega_{x/y}$ et $\phi_{x/y}$ telles que les profils de vitesse d'une trace soient donnés par :

$$\dot{x}/\dot{y}(t) = a_{x/y}(t)\sin(w_{x/y}(t)t + \phi_{x/y}(t)) \tag{4.2}$$

Ce problème a une infinité de solutions quels que soient les profils de vitesses $\dot{x}$ et $\dot{y}$. Donc pour que ce problème fasse sens il faut ajouter des contraintes fortes sur les fonctions $a_{x/y}$, $\omega_{x/y}$ et $\phi_{x/y}$. C'est ce qui a été fait pour POMH où ces fonctions ont été supposées constantes par morceaux avec des points de changement toutes les demi-périodes.

Nous cherchons ici à estimer la phase relative (PR), donnée par $\phi = \phi_x - \phi_y$. Les contraintes auxquelles sont soumises les autres paramètres de cette équation seront détaillées au gré des différentes méthodes de calcul de la PR.

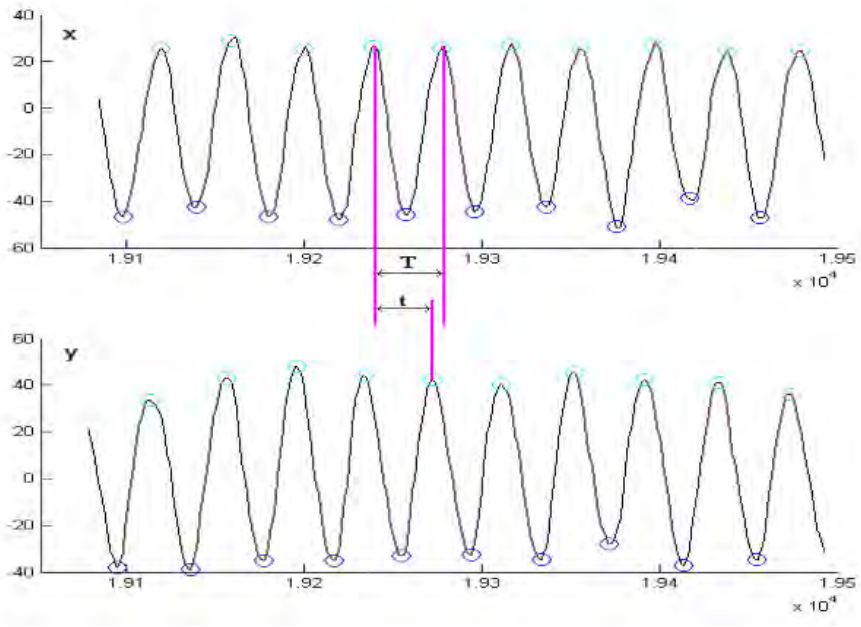


Figure 4.7 – Calcul de la phase relative discrète. Images tirées de Danna et al. [2011].

.

4.3.1.1 Calcul de la phase relative dans la littérature

Plusieurs méthodes documentées permettent d'estimer la phase relative. Trois en particulier méritent d'être abordées car ce sont les plus utilisées, une quatrième élaborée par nos soins sera présentée ensuite. Les calculs de phase présentés ne font sens que si les signaux étudiés sont quasi-harmoniques, ou s'ils ont du moins une bande de fréquence réduite ; dans le cas contraire, les valeurs de phase et phase relative données par ces méthodes n'ont pas grande signification.

La phase relative peut être calculée de manière discrète ou de manière continue (PRD et PRC). Le calcul de la PRD à l'avantage de donner des valeurs plus stables et moins bruitées mais, il ne donne qu'une valeur par demi-période du signal. Le calcul de la PRC donne en revanche autant de valeurs qu'en contient le signal étudié. En revanche, les résultats sont plus bruités et irréguliers.

Calcul de la PRD Le calcul de la phase relative discrète est basé sur l'extraction des moments des maxima des deux signaux pour lesquels on cherche à déterminer la PR. Ces moments permettent de calculer pour chaque demi-période une estimation de la phase (voir Figure 4.7). La PRD (en degrés) est alors calculée par

$$\phi = \frac{t2 - T1}{T2 - T1} \times 360, \tag{4.3}$$

où $T1$ et $T2$ sont les temps de deux maxima (resp. minima) consécutifs de l'un des oscillateurs pris pour référence et $t2$ correspond au maxima correspondant à $T2$ pour l'autre oscillateur. Cette méthode, applicable à n'importe quel type d'oscillateur, est abordée dans de nombreux travaux [Hamill, 2000; Hamill, Bates, et Holt, 1992; McClay et Manal, 1997; Zanone et Kelso, 1992].

Calcul de la PRC Deux méthodes sont essentiellement utilisées pour le calcul de la phase relative continue : la première est basée sur l'arc-tangente et la seconde sur la

transformée de Hilbert.

La méthode basée sur l'arc-tangente, est présentée plus en détail dans la partie suivante dans le cas particulier de nos travaux. Le principe général reste le même. Par ailleurs, certains ajustements peuvent être nécessaires pour obtenir des résultats cohérents (par exemple, le recentrage de l'oscillateur dans l'espace des phases et la normalisation du signal dans ce même espace [Hamill, 2000; Peters, Haddad, Heiderscheit, Van Emmerik, et Hamill, 2003]). Cependant Kurz et Stergiou [2002] montrent que ces ajustements (en particulier la normalisation) ne sont peut-être pas souhaitables est peuvent induire des erreurs.

La méthode de calcul de la PRC basée sur la transformée de Hilbert s'appuie sur les propriétés de cette dernière [Panter, 1965; Rosenblum et Kurths, 1998; Rosenblum, Pikovsky, et Kurths, 2001]. Si on note $\tilde{\cdot}$ l'opérateur de transformation de Hilbert, alors la phase relative entre deux signaux s_1 et s_2 (dans notre cas les profils de vitesse en x et y) est donnée par

$$\phi = \phi_1(t) - \phi_2(t) = \arctan \frac{\tilde{s_1}(t)s_2(t) - s_1(t)\tilde{s_2}(t)}{s_1(t)s_2(t) + \tilde{s_1}(t)\tilde{s_2}(t)}. \tag{4.4}$$

4.3.1.2 Méthode de l'arc-tangente et méthode personnelle de l'arc-cosinus

Dans ces deux méthodes, le calcul de la PRC, se fait par calcul de la phase sur chacun des deux signaux de vitesse. On montre ici le calcul de la phase pour le profil des vitesses horizontales, le calcul est identique pour le profil des vitesses verticales. La phase relative ϕ est ensuite calculée par différence $\phi = \phi_x - \phi_y$.

On veut donc estimer la phase d'un signal S (profil de vitesse) échantillonné dont l'on suppose qu'il suit la formule :

$$\dot{x}(t) = a(t)\sin(\omega_x(t)t + \phi_x) \tag{4.5}$$

On suppose l'écriture périodique, donc w_x est supposé constant et estimé par la moyenne des fréquences de la série paramétrique renvoyée par POMH. L'amplitude du signal a est toujours supposée constante par morceaux sur chaque demi-période et est estimée par le maximum de S entre chaque paire de zéros.

On peut donc estimer la phase par (méthode (i), méthode de l'arc-tangente) :

$$\frac{\dot{x}}{\ddot{x}} = \frac{a(t)\sin(\omega_x t + \phi_x)}{a(t)\omega_x \cos(\omega_x t + \phi_x)} \tag{4.6}$$

$$\frac{\omega_x \dot{x}}{\ddot{x}} = \tan(\omega_x t + \phi_x) \tag{4.7}$$

$$\phi_x = \arctan\left(\frac{\omega_x \dot{x}}{\ddot{x}}\right) - \omega_x t \tag{4.8}$$

Alternativement, on peut utiliser (méthode (ii), méthode de l'arccos) :

$$\phi_x = \tau\frac{\pi}{2} + (-1)^\tau \arcsin\left(\frac{\dot{x}}{a(t)}\right) - \omega_x t \tag{4.9}$$

où

$$\tau = \begin{cases} 0 \text{ si } \arccos\left(\frac{\ddot{x}}{\omega_x a(t)}\right) < \frac{\pi}{2} \\ 1 \text{ sinon} \end{cases} \tag{4.10}$$

La soustraction de $\omega_x t$ pour le calcul de chaque phase peut être omise car ce terme disparaît dans le calcul de la phase relative ϕ pour ces deux méthodes.

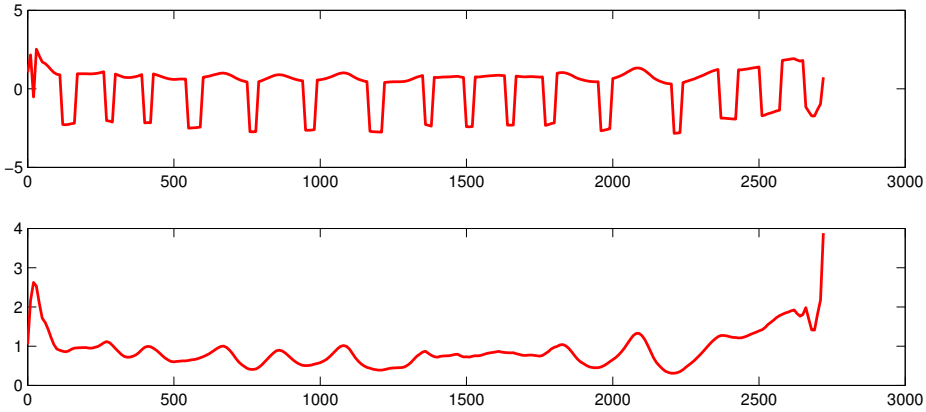


Figure 4.8 – Correction des décrochages dans le calcul de la phase relative. En haut la phase relative calculée par la méthode de l'arc-tangente. En bas la phase relative corrigée. En ordonnées la mesure de la phase relative en radians, en abscisses le temps en ms.

4.3.2 Essais sur l'écriture manuscrite

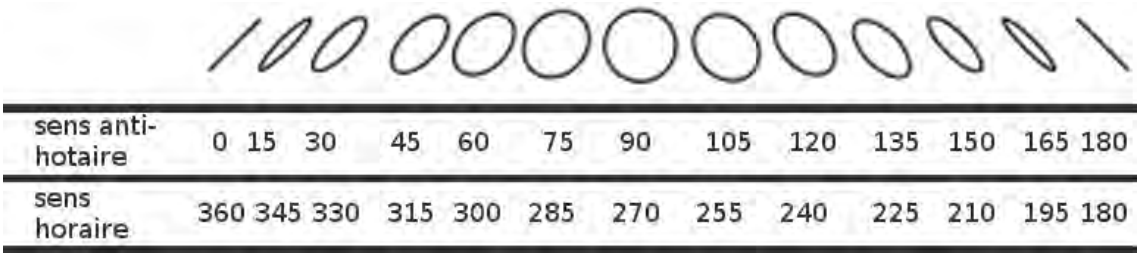
Nous avons calculé la phase relative continue en utilisant les deux méthodes présentées au chapitre précédent, ainsi qu'en utilisant la méthode basée sur la transformation de Hilbert à des fins de comparaison. En effet, le calcul de la phase pour un signal quasi-oscillatoire est sujet à caution.

Lorsque l'on applique ces algorithmes de calcul de phase à des signaux, on observe souvent des décrochages de $k\pi$: ceci est compréhensible étant donnés les domaines de définition de arctan et arccos. Un algorithme simple corrige ces sauts et recentre la phase ainsi calculée autour de zéro. La figure 4.8 montre une phase relative continue calculée par la méthode (i) et sa version corrigée, continue.

4.3.2.1 Expérience préliminaire en vue de valider et comprendre les modes de calcul de la PRC sur l'écriture manuscrite

Le but de cette expérience est de déterminer la distribution de la phase relative continue de l'écriture sur un échantillon restreint (100 mots) de traces écrites par un sujet. Nous espérons que cette distribution ne soit pas uniforme et montre des pics autour des valeurs préférentielles mise en avant par Sallagoïty et al. [2004] et Athènes et al. [2004] à savoir 0°, 45°, 125° et 180° (avec une retenue pour les deux coordinations singulières 0 et 180°, qui correspondent, dans le tracé d'ellipses, à des répulseurs). Si tel est le cas, nous pourrons valider chacune des méthodes de calcul de la PRC.

Avant cela, nous devons faire quelques considérations sur la phase relative. Dans leurs travaux, Athènes et al. [2004] présentent des ellipses correspondant à des phases relatives allant de 0° à 180°, qu'ils font tracer aux participants dans le sens horaire et dans le sens anti-horaire. Or le sens de parcours de l'ellipse change la phase relative réelle du mouvement. Dans notre cas, il est impossible d'imposer aux participants de tracer les lettres dans un sens ou l'autre, cela n'aurait aucune signification : on est obligé de tenir compte de phase relative allant de 0° à 360°. Si l'on reprend la série d'ellipses utilisée par Athènes et al. [2004], la figure 4.9, montre les phases relatives associées, en degrés, au tracé effectué dans le sens anti-horaire et dans le sens horaire.



sens anti-hotaire	0	15	30	45	60	75	90	105	120	135	150	165 180
sens horaire	360	345	330	315	300	285	270	255	240	225	210	195 180

FIGURE 4.9 – Correspondance entre forme des ellipses et phase relative dans le cas d'un tracé en sens anti-horaire et d'un tracé en sens horaire.

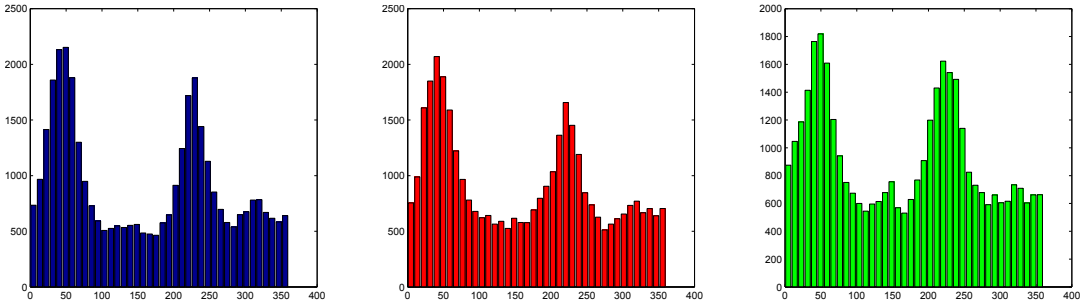


FIGURE 4.10 – Distributions des PR pour 100 mots pour un scripteur pour trois méthodes de calcul de phase continue différentes (en bleu, la méthode (ii) basée sur l'arccos ; en rouge la méthode (i) basé sur l'arctan ; en vert la méthode basée sur la transformée de Hilbert).

.

Procédure Il a été demandé à un scripteur de tracer 100 mots sans lever le crayon, de la manière la plus naturelle possible, sans autre contrainte.

Analyse des données La phase relative continue (PRC) a été calculée sur chacun des mots avec les trois méthodes ((i),(ii) et Hilbert). Pour chacune de ces méthodes, une distribution sur $[0^\circ, 360^\circ]$ est calculée.

Résultats La figure 4.7 montre la distribution de la phase relative continue (calculée avec les trois méthodes présentées précédemment) pour 100 mots. Pour les 3 méthodes, la distribution calculée est similaire.

Chacune des distributions montre un pic de PR autour de 45° et 230°. Ce résultat semble confirmer les travaux de Sallagoïty et al. [2004]; Athènes et al. [2004] où 45° correspond à la coordination non singulière la plus stable. Le pic que l'on observe autour de 230° correspond à une ellipse associée à 145°, mais avec un sens de rotation inverse. Par ailleurs les distributions montrent des pics moindres autour de 130° et 320° qui correspondent aux deux états stables mis en évidence précédemment mais dans le sens inverse.

4.3.2.2 Discussion

Méthodes de calcul de la PRC Ces trois méthodes de calcul de la PRC, donnent des résultats similaires, à l'échelle statistique, compatibles avec les travaux pré-cités [Athènes et al., 2004; Sallagoïty et al., 2004]. On peut donc affirmer, que les méthodes de calcul de la PRC donnent des résultats cohérents et utilisables dans le cadre de l'écriture. Les pics de distribution correspondent aux deux attracteurs observés dans le cas du tracé d'ellipse.

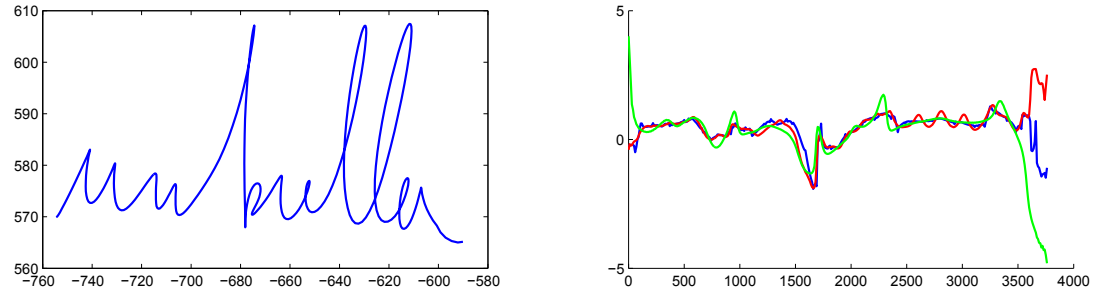


Figure 4.11 – A droite la trace étudiée et à gauche les calculs de PRC effectués sur cette trace pour les trois méthodes étudiées : l'arctan en rouge, l'arccos en bleu et Hilbert en vert.

.

Nous regardons en détail, sur un exemple particulier, les résultats du calcul de la PRC afin de pouvoir comparer qualitativement les trois méthodes (voir la figure 4.11). L'allure générale de la PRC, pour les trois méthodes, est la même. Ceci montre que la définition de la PRC fait sens dans le cas de l'écriture. On remarque qu'il y a des différences significatives en début et fin de trace ; cela est dû à des erreurs de calculs inhérentes à l'une ou l'autre des méthodes en ces points particuliers. Si l'on se concentre sur ce qu'il se passe entre 100 et 3500 ms, on remarque que les courbes données par les méthodes arctan et Hilbert sont plus régulières que celle donnée par la méthode de l'arccos, mais qu'elles oscillent plus. L'arccos est la méthode montrant le moins de variations. La méthode de l'arccos utilise une estimation explicite des amplitudes pour déterminer la PR. Cette estimation semble absorber certaines variations du signal original, variations non absorbées dans les deux autres méthodes.

Plus, généralement, après observation des courbes données pour 12 mots, les trois méthodes donnent de bons résultats, même si elles sont sujettes à certains aléas dus à des problèmes de calcul. La méthode de l'arccos nécessite la connaissance a priori de l'amplitude et de la fréquence, la méthode de l'arctan nécessite la connaissance de la fréquence tandis que la méthode d'Hilbert ne nécessite aucune donnée à priori. La méthode de l'arccos est celle qui montre la courbe de PR avec le moins de variations locales mais manque de régularité (ie. la courbe n'est pas lisse). Les deux autres méthodes, varient plus à certains endroits (pas nécessairement les mêmes dans les deux cas) mais sont plus lisses.

Nous pensons que chacune de ces méthodes est valide afin d'étudier et définir la phase relative continue de l'écriture, l'étude des distributions associées donnant des résultats similaires et pertinents. Dans la suite, nous nous reposerons plus sur la méthode de l'arccos, pour bénéficier de ses variations moindres.

4.4 Expérience - Dynamique de coordination chez l'adulte pour une tâche d'écriture

L'expérience présentée dans cette section a pour but de confirmer les résultats de Athènes et al. [2004] sur de l'écriture réelle constituée de phrases entières et de trouver les corrélations, s'il y en a, entre la vitesse de l'écriture chez l'adulte, la géométrie de l'écriture et la dynamique de coordination (décrite par la PRC).

4.4.1 Méthode

4.4.1.1 Participants

Six adultes droitiers, âgés de 25 à 55 ans, ont accepté de participer bénévolement à cette étude. Ils ne présentaient a priori aucun trouble du langage et aucun trouble moteur.

4.4.1.2 Tâche et procédure

Il a été demandé aux participants d'écrire trois phrases données, chaque phrase devant être réécrite cinq fois. L'écriture devait être le plus naturelle possible. Il a été demandé aux participants de ne pas s'appliquer. La taille de l'écriture était libre tant que chaque phrase était contenue dans une ligne.

4.4.1.3 Matériel

Les phrases était écrites sur un fond blanc affiché sur l'écran d'une tablette graphique Wacom Cintiq 12WX avec un écran LCD de 261.1×163.2 mm d'une résolution de 1280×800 pixels inséré dans un cadre en plastique noir (405×270×17 mm) pouvant être orienté à souhait comme une feuille de papier. Le stylet associé avait approximativement la même taille (175 mm de long et un diamètre de 14 mm en moyenne) et le même poids (17 g) qu'un stylo à bille normal. Les participants étaient assis sur une chaise haute ajustable, face à la tablette posée sur une table. Il leur a été demandé d'adopter une posture d'écriture confortable. Sitôt que le stylet touchait la surface de l'écran, les cordonnées spatiales x et y de la trajectoire tracée étaient échantillonnées à une fréquence de 100Hz avec une résolution spatiale de 0.2 mm. La trace produite était affichée en temps réel sur l'écran dans le but d'offrir aux participants un retour visuel proche de celui qu'ils auraient en écrivant avec un crayon sur une feuille. Les positions du stylet lors du lever de crayon étaient aussi enregistrées mais n'étaient pas affichées sur l'écran. A la fin du tracé de la phrase, l'expérimentateur mettait fin à l'enregistrement en appuyant sur une touche du clavier associé à l'ordinateur sur lequel la tablette était branchée. L'ensemble de l'expérimentation était géré par le programme ExpDO présenté en annexe A.

Même si tout a été fait pour que les participants soient dans les conditions les plus naturelles possibles, certains d'entre eux ont rapporté avoir été gênés par le dispositif qui selon eux ne les met pas dans les mêmes dispositions qu'une tâche d'écriture usuelle. Par ailleurs, l'expérimentateur a remarqué que le glissement de poignet qui intervient dans l'écriture était gêné par le cadre en plastique noir. En fin de ligne, certains participants ont préféré effectuer une rotation de leur poignet plutôt qu'une translation, ce qui a pu avoir un impact sur le résultat.

4.4.1.4 Analyse des données

Chacune des phrases à été découpée en traces, une trace correspondant au mouvement entre un poser et un lever de crayon. Seule les traces ayant une durée supérieure à 700ms ont été étudiées (pour éviter des problèmes liés au calcul). Sur chacune des traces, les trois méthodes de la PRC présentées plus haut ont été appliquées. Pour chaque participant, une distribution des PRC, sur l'ensemble des traces étudiées a été calculée. Les distributions ont été normalisées afin de pouvoir les comparer entre elles. Par ailleurs la vitesse moyenne de tracé d'une phrase pour chaque participant a été calculée.

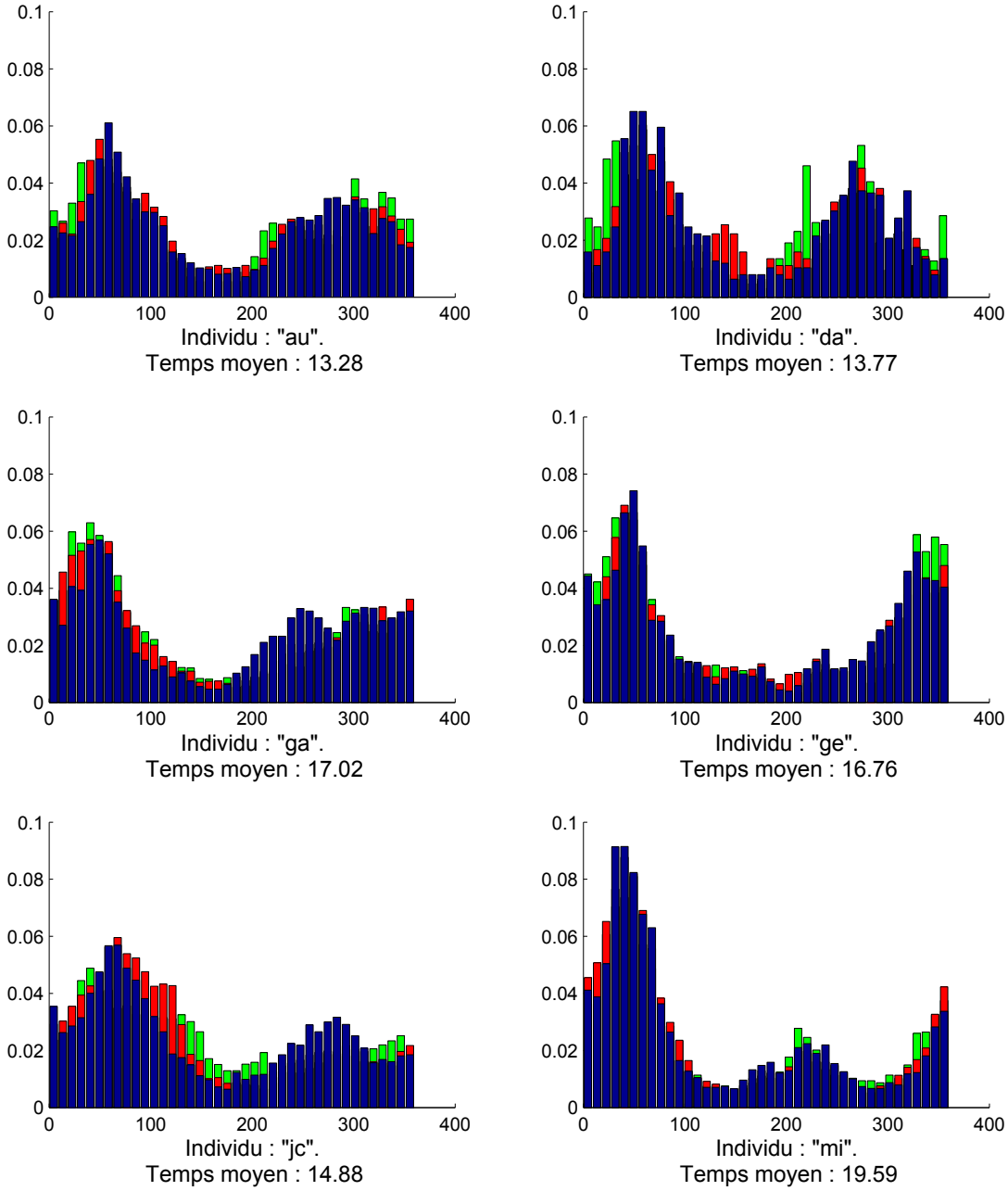


Figure 4.12 – Distributions des PR pour chacun des participants pour les trois méthodes de calcul de phase continue (en bleu, la méthode (ii) basée sur l'arccos ; en rouge la méthode (i) basé sur l'arctan ; en vert la méthode basée sur la transformée de Hilbert). Ces distributions sont obtenues par le calcul de la PRC pour chacun des 5 exemplaires des 3 phrases qui ont été écrites par chaque participant.

.

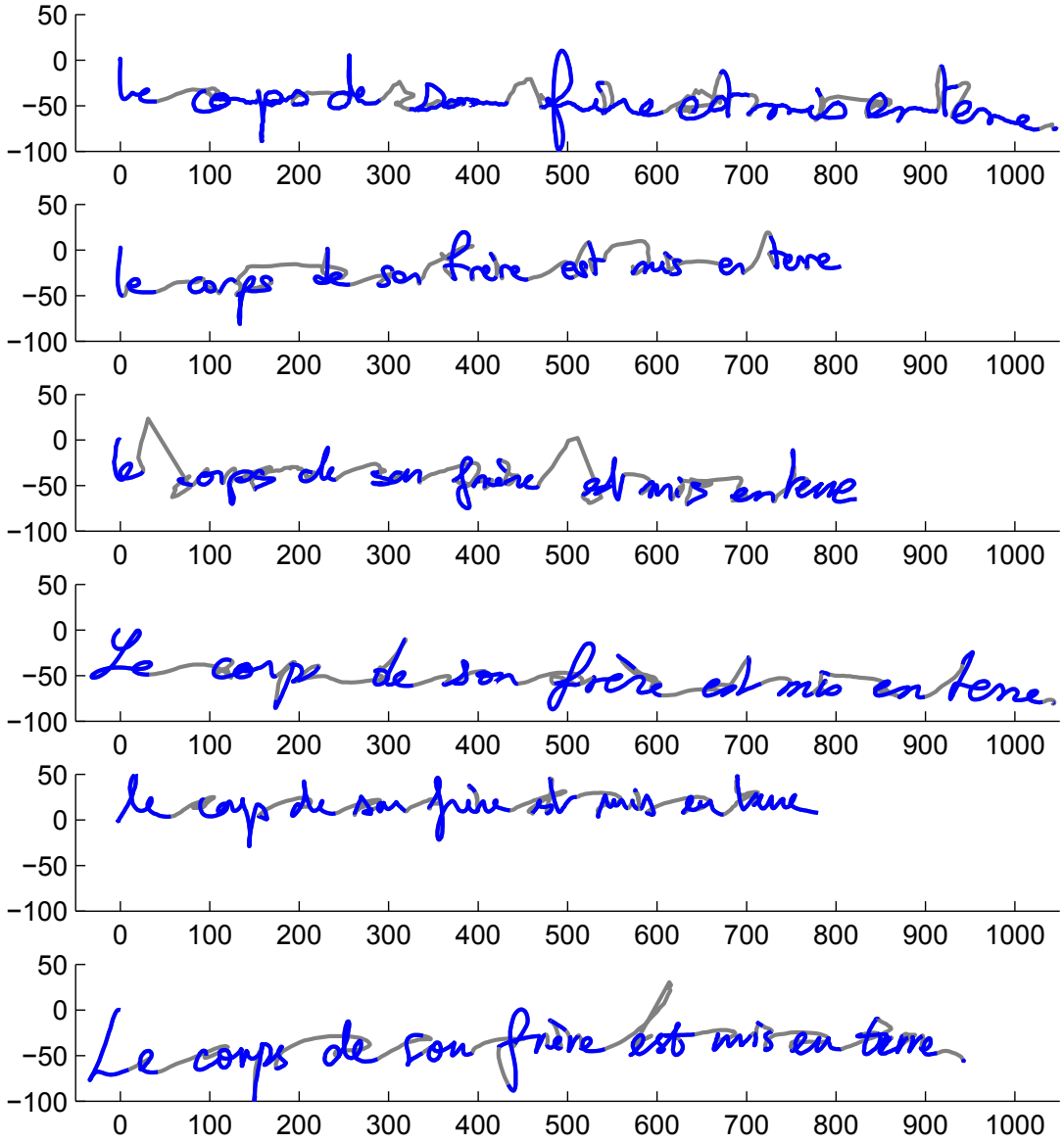


FIGURE 4.13 – Un exemple de phrase écrite par chacun des participants (de haut en bas "au", "da", "ga, "ge", "jc" et "mi").

4.4.2 Résultats

Nous nous contenterons d'une inspection visuelle des résultats, les calculs de moyenne et d'écart-type sur les distributions présentées en figure 4.13 n'ayant pas grand sens. Cette figure montre, pour chaque participant, les distributions des PR calculées en chaque point du signal échantillonné. Les distributions mises en avant en bleu correspondent aux phases relatives calculées avec la méthode de l'arccos ; les distributions rouges sont associées à la méthode de l'arctan et les distributions vertes à la méthode de la transformée de Hilbert. La figure 4.12 montre quant-à-elle un exemple de trace écrite par chacun des participants.

Pour chacun des participants on note la présence d'un pic de répartition autour de 50 °. Par ailleurs un pic moins prononcé et plus large autour de 250-270 ° est présent chez quatre participants. Les deux autres sont "ge" qui présente un deuxième pic prononcé, autour de 330 ° et "mi" qui quand-à-elle présente un tout petit pic autour de 220 °. Ces deux participants semblent produire une plus faible variété de phases relatives que les autres.

Les distributions de la PRC pour chacune des méthodes semblent relativement similaires. Par contre la méthode basée sur la transformée de Hilbert semble donner des distributions plus irrégulières. La méthode de l'arccos donne tout de même des distributions plus stables avec des pics plus resserrés.

En ce qui concerne la vitesse d'écriture, il y a une grande disparité entre les scripteurs, puisque la vitesse moyenne pour "au" est de 13,22s quand elle est de 19,46s pour "mi" (soit environ 150% du temps de "au"). "mi" et "ge" sont parmi les trois participants les plus lents ; comme nous l'avons dit plus haut, ce sont aussi eux qui ont les distributions les plus singulières (avec une plus faible répartition des phases relatives). Ces deux participants sont aussi ceux présentant l'écriture la plus penchée (voir figure 4.12).

4.4.3 Discussion

Les travaux que nous avons présentés plus tôt dans ce chapitre, ont montré l'existence de quatre patrons de coordination préférentiels lors du tracé d'ellipses correspondant à des phases relatives de 0, 45, 120 et 180 ° [Athènes et al., 2004; Sallagoïty et al., 2004]. Notre but était de vérifier si ces résultats se retrouvaient dans la distribution des phases relatives mesurées à partir de traces d'écriture réelles.

L'existence pour chacun des participants d'un pic de distribution autour de 50-60 °, correspond à l'attracteur associé à 45 ° dans les travaux précités. Un point intéressant serait de trouver le lien entre ces deux observations. Le patron de coordination associé à 45 °, apparaissant au cours de l'enfance, conjointement à l'apprentissage de l'écriture, laisse supposer que ce sont les formes de l'écriture à tracer qui conduiraient à l'apparition de cet attracteur. On remarque par ailleurs que les deux patrons intrinsèques de coordination à 0 ° et 180 ° présents aussi bien chez l'enfant que chez l'adulte, ne semblent pas ressortir dans nos distributions. Nous l'expliquons de la même manière qu'Athènes et al. [2004] : une discrimination visuelle de la part du scripteur empêche ces deux attracteurs de s'exprimer. En ce qui concerne le patron préférentiel associé à 120 °, il ressort sous sa forme "horaire", associé à une PR de 220-240 ° chez certains scripteur.

Chez la plupart des scripteurs, le pic, large autour de 270-280 °, pourrait être le pendant des formes associées 50-60 ° tracées dans le sens horaire. Le fait que ce pic soit plus large et proche de 270 ° laisse penser que les scripteurs sont moins à l'aise avec le sens horaire. Ceci semble confirmer les observations d'Athènes et al. [2004], qui ont remarqué que lors de la reproduction de certaines ellipses, l'erreur absolue était plus grande si le tracé avait lieu dans le sens horaire que dans le sens anti-horaire.

Sur ces six sujets, il n'y a aucun lien entre vitesse de l'écriture et taille du pic observé. Par contre, les deux sujets ayant une écriture plus penchée ont un défaut de distribution autour de 90-120 °, ils sont aussi parmi les plus lents. Est-ce à dire que les scripteurs les plus rapides sont ceux capables de reproduire plus facilement le plus grand nombre de patrons de coordination ?

Les deux écritures les plus penchées (celles de "ge" et "mi") sont celles dont le pic de distribution le plus élevé correspond à la phase relative la plus faible, cela est compatible avec l'équation 2.20 présentée par Hollerbach [1981] exprimant la pente de l'écriture en fonction de la phase relative. On notera au passage que, grâce à cette équation, puisque l'on a un accès facile aux amplitudes, si l'on trouve une mesure fiable de ϕ alors on obtiendra une mesure fiable de la pente et vice-versa.

4.5 Dégradation de la géométrie de l'écriture sous l'effet d'une vitesse imposée

A notre connaissance, les effets de la vitesse sur la géométrie de la trace écrite n'ont été que peu étudiés. Pourtant, il nous semble que comprendre ceux-ci nous permettrait d'acquérir une meilleure compréhension de la production de la trace chez l'humain.

Une étude voulant étudier ces effets doit orienter ses recherches selon deux directions. La première, serait d'étudier au cours de l'apprentissage de l'écriture, et en particulier au collège, l'évolution de la forme de l'écriture des sujets face à la nécessité d'écrire toujours plus, toujours plus rapidement. La deuxième serait de tenter d'étudier, pour un sujet adulte la dégradation que subirait un mot si on lui imposait un temps d'écriture minimal. Serait-il capable d'accomplir cette tâche ? A quel prix sur la lisibilité du mot ?

Dans cette section nous proposons des hypothèses pour chacune des directions et pour chacune des hypothèses nous proposons un protocole expérimental. Malheureusement, ces expériences n'ont pas encore été menées ; les résultats ne pourrons donc pas être montrés ici. Les hypothèses qui seront émises ici ne sont pas suffisantes, c'est à dire qu'elles cherchent à rendre compte de certains aspects de la déformation géométrique, mais pas de tous.

4.5.1 Première hypothèse

Danna [2011] a montré qu'au cours de l'apprentissage de l'écriture, le paysage des patrons de coordination préférentiels de l'écriture évolue d'un stade où il y a deux patrons préférentiels de coordination, à 0 et 180 °, vers un stade où il y en a 4, à 0, 45, 120 et 180 °, en passant par un état intermédiaire où il y a trois patrons préférentiels, à 0, 90 et 180 °. Notre hypothèse est qu'au cours de l'apprentissage de l'écriture, du passage de l'état intermédiaire à l'état final, l'attracteur situé à 90 ° se déplace vers 0 ° au fur et à mesure que la vitesse d'écriture du sujet augmente. Finalement, plus cet état stable est proche de 0 ° et plus le sujet aurait une vitesse d'écriture élevée. Géométriquement, cela se traduit par la règle qui voudrait que moins l'écriture d'un sujet est arrondie, plus il écrit, en général, rapidement.

On pourrait croire que, pour vérifier cette hypothèse, il suffirait de comparer la vitesse d'écriture de sujets adultes et d'étudier les pics de phase relative associés. Seulement d'autres paramètres entrent en considération dans la forme de l'écriture, et sa vitesse de trace, comme le montrent les équations 2.20 et 2.19. Par ailleurs comme nous pouvons le voir sur les distributions présentées en section précédente (figure 4.13), notre hypothèse n'est pas vérifiable : deux participants parmi les plus lents ont des pics de distribution correspondant à des phases relatives les plus faibles. Comme expliquer cela ? Si l'on considère

que nous ne sommes pas tous égaux en matière de vitesse de mouvement, on peut supposer que les personnes les plus intrinsèquement lentes ont besoin, au cours de l'apprentissage de l'écriture, de plus décaler leur attracteur vers zéro pour compenser cette lenteur relative.

Il nous faudrait donc faire des comparaisons, pour chaque sujet, à différents stades de leur apprentissage de l'écriture. Le protocole serait alors le suivant :

Protocole Sur N sujet, tous les 6 mois de 7 à 18 ans, on réalise l'expérience de scanning présentée par Athènes et al. [2004], voir section 4.2.1. Puis les sujets doivent ensuite réaliser l'expérience présentée en section 4.4. L'hypothèse serait alors vérifiée si pour chaque sujet, l'apprentissage de l'écriture et, en particulier, l'augmentation de la vitesse de tracé, s'accompagne d'un décalage progressif d'un attracteur de 90° vers 0°. Ce décalage devrait être d'autant plus prononcé pour les sujets ayant eu le plus fort accroissement de vitesse d'écriture par rapport à leur vitesse initiale.

4.5.2 Seconde hypothèse

Pour cette hypothèse nous nous intéressons à la déformation géométrique que subit l'écriture si on "force" un sujet à écrire un mot à une vitesse plus élevée que sa vitesse maximale habituelle. Des observations, effectuées sur trois sujets, laissent penser qu'il est difficile d'augmenter sa vitesse d'écriture. En effet, il a été demandé à ces sujets d'écrire quelques mots à vitesse spontanée, puis d'écrire ces mêmes mots à vitesse maximale (que l'on nomme vitesse spontanée maximale). Les résultats sont sans appel ; les changements géométriques ainsi que la différence de temps de production de ces mots sont minimes. Les mots écrits à vitesse maximale spontanée l'étaient dans une durée rarement inférieure à 70% de la durée associée à la vitesse spontanée.

L'idée est alors de "forcer" les scripteurs à aller encore plus vite. S'ils y arrivent vraiment, alors on suppose qu'il y aura une déformation dramatique de la trace. Dans le cas contraire, on suppose que la trace ne sera simplement pas terminée et aura l'aspect, pour la partie effectuée, de la géométrie associée à la vitesse maximale spontanée.

Dans le cas où il y a déformation majeure de la trace, notre hypothèse est que l'un des aspects de cette déformation est donnée par la disparition du patron préférentiel de coordination autour de 45° et que la phase relative associée au tracé se trouve alors attirée par l'attracteur 0°, comme dans le cas de la coordination bimanuelle (voir section 4.1.2).

Une simulation de notre hypothèse sur une trace "umbrella" écrite à vitesse spontanée est présentée en figure 4.14. Plus la phase relative se contracte, plus la trace perd ses spécificités. En particulier les boucles disparaissent, les "pics" et les "ponts" deviennent semblables. Seule l'amplitude des lettres semble encore permettre de les différencier. Nous ne formulons aucune hypothèse sur cette dernière, mais, il semble clair que la vitesse de l'écriture a aussi un impact sur celle-ci. Il parait aussi naturel que le nombre d'oscillations utilisées pour écrire le mot chute (entraînant la disparition de lettres ou de parties de lettres). Il pourrait être intéressant de tenter de prédire quelles oscillations correspondant à quelles lettres sont impactées.

Protocole Pour N sujet, il s'agit d'étudier l'évolution de la phase avec la vitesse de tracé. On choisit M mots et pour chaque mot, on commence par demander au sujet de le tracer à vitesse maximale spontanée. Puis, on "force" le sujet à écrire le mot de plus en plus vite, une dizaine de fois. A chaque itération le temps autorisé pour écrire ce mot est réduit de 5% par rapport à la vitesse maximale spontanée. Pour "forcer" un sujet à écrire sa trace dans un temps imparti, et pour qu'il ait une idée du temps qu'il lui reste, on peut utiliser un signal sonore dont la variation en fréquence suit toujours le même schéma. On enregistre

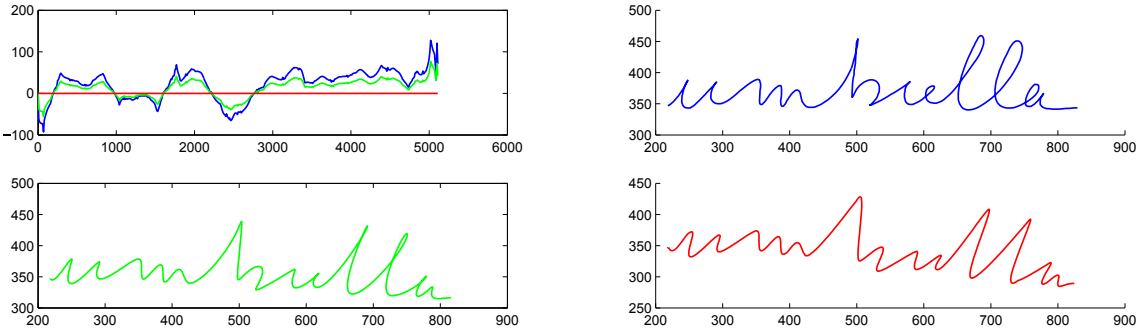


FIGURE 4.14 – Simulation de la déformation d'une trace écrite à vitesse spontanée (en haut à droite). En bas à gauche la trace à été reconstruite par contraction de la phase relative de 30%, en bas à droite, la trace à été reconstruite en utilisant une phase relative nulle. Les profils de phase relative correspondant sont présentés en haut à gauche.

par ailleurs le temps de tracé de chaque trace. Notre hypothèse sera vérifiée si l'on observe une uniformisation topologique de la trace (disparition des boucles, des pics ...) associée, ou non, à la disparition de certaines oscillations. Enfin, il pourrait être intéressant de vérifier si les différentes déformations successives surviennent par paliers (par rapport à la vitesse) plutôt que de manière progressive. C'est d'ailleurs le cas, dans la coordination bimanuelle où la transition du mode en anti-phase vers le mode en phase se fait d'un coup.

Chapitre 5

Reconnaissance off-line de l'écriture manuscrite cursive, un modèle interactiviste

5.1 Introduction

Les deux chapitres précédents ont principalement présenté un modèle de production de l'écriture manuscrite continu et basé sur des processus oscillatoires. Le problème de la construction d'un modèle de perception associé pourrait sembler être d'une toute autre nature, le thème d'un nouveau travail radicalement différent.

Pourtant, nous allons essayer de montrer qu'au contraire, les processus de production d'une activité par un sujet et ceux de perception de cette activité par le sujet sur lui-même ou sur d'autres sont intimement liés, quelles que soient les modalités perceptives ou d'activité, au point que nous ferons l'hypothèse qu'ils sont en grande partie communs. Nous ébaucherons alors une méthode de perception d'une trace écrite basée sur ces principes. En bref, nous défendrons la thèse que lire et écrire, pour l'écriture manuscrite cursive, sont deux facettes d'un même processus.

Nous commencerons d'abord par examiner dans la littérature les preuves de ces rapports étroits entre activité et perception, pour toutes les modalités sensorielles et dans de nombreux types d'activités, en détaillant en particulier la production et la perception de l'écriture manuscrite.

Nous présenterons ensuite les cadres théoriques dans lesquels doivent s'inscrire de tels modèles de production/perception d'une activité. Les cadres computationnels classiques sont en effet inadaptés à ce type de formulation, principalement par le fait qu'ils séparent les étapes de perception et d'action à deux extrémités d'une chaine de traitement sensori-motrice, séparation mise en cause par des auteurs de plus en plus nombreux. Nous présenterons le cadre interactiviste proposé par Bickhard [2008] qui, au contraire, considère l'interaction comme la forme fondamentale de représentation d'un agent, sous la forme de processus proactifs et anticipateurs. Nous montrerons avec Bickhard que ces hypothèses permettent de résoudre ou de dissoudre plusieurs problèmes théoriques anciens liés à la perception et à la représentation.

À l'intérieur de ce cadre interactiviste, nous présenterons alors un modèle de production/perception qui utilise principalement la notion de simulation de l'action, la perception consistant ainsi à utiliser les processus de production disponibles pour recréer de façon interne une trace compatible avec les éléments sensoriels. Nous présenterons enfin deux implémentations préliminaires de ce modèle pour résoudre un problème de reconnaissance

off-line d'une trace écrite. Nous donnerons quelques exemples : l'un illustrera notamment la capacité de ce modèle à séparer la forme du fond (lorsque la trace est bruitée ou mélangée à d'autres traces).

Ce travail sur la perception est un point de départ, une tentative dans une nouvelle direction de recherche. Les premiers résultats semblent prometteurs et montrent les voies à suivre dans des travaux ultérieurs pour le compléter et l'améliorer.

5.2 Les rapports entre activité et perception

5.2.1 Philosophie

Les rapports entre action et perception ont été remarqués depuis longtemps par de nombreux auteurs. Selon Bergson [1939] : « Percevoir consiste à détacher de l'ensemble des objets, l'action possible de mon corps sur eux. La perception n'est alors qu'une sélection. Elle ne crée rien, son rôle est au contraire d'éliminer de l'ensemble des images toutes celles sur lesquelles je n'aurais aucune prise, puis de chacune des images retenues elles-mêmes, tout ce qui n'intéresse pas les besoins de l'image que j'appelle mon corps. Telle est du moins l'explication très simplifiée de ce que nous avons appelé la perception pure ». On a ici l'idée que la perception est principalement au service du corps et de son action. Par contre, nous ne partageons pas l'idée que « la perception ne crée rien ». Tout au contraire, nous chercherons à montrer qu'une perception est bien une création interne de l'agent percevant. Comme le dit Bergson, la construction perceptive va abstraire certains éléments, mais d'autres, relatifs à des parties occultées par exemple, peuvent être ajoutés.

Merleau-Ponty [1942] montre dans le détail comment la perception n'est pas réductible à un ensemble de sensations, et qu'elle est intimement liée à l'activité qu'elle accompagne. Un exemple parlant concerne le joueur de football : "Le terrain de football n'est pas, pour le joueur en action, un 'objet', c'est à dire le terme idéal qui peut donner lieu à une multiplicité indéfinie de vues perspectives et rester équivalent sous ses transformations apparentes. Il est parcouru par des lignes de force (les 'lignes de touche', celles qui limitent la 'surface de réparation'), articulées en secteurs (par exemple les 'trous' entre les adversaires) qui appellent un certain mode d'action, la déclenchent et la portent comme à l'insu du joueur.". Ici encore, la perception est au service de l'action de l'agent et Merleau-Ponty illustre également comment elle dépend de ses intentions. Ainsi à un même endroit du terrain, la perception d'un joueur attaquant n'est pas la même que celle d'un défenseur, car leurs intentions et leurs opportunités d'actions sont différentes. Cette notion est à rapprocher de celle d'affordance de Gibson [1966].

5.2.2 Neurobiologie

Mais les premiers éléments de preuve biologiques et neurologiques des liens entre action et perception proviennent des recherches sur les "neurones miroirs", initiées par Rizzolatti, Fadiga, Gallese, et Fogassi [1996]. Avec son équipe, il a identifié chez le macaque des populations de neurones corticaux qui s'activent à la fois lorsque le macaque réalise une certaine action (telle que : attraper, tenir, arracher), et lorsqu'il perçoit visuellement la main d'un expérimentateur qui réalise cette même action. On a ensuite montré des résultats analogues avec des effecteurs tels que la bouche [Buccino, Binkofski, Fink, Fadiga, Fogassi, Gallese, Seitz, Zilles, Rizzolatti, et Freund, 2001], avec le sens du toucher sur l'ensemble du corps [Keysers, Wicker, Gazzola, Anton, Fogassi, et Gallese, 2004], avec l'ouïe. Par exemple, des neurones miroirs ont été identifiés chez une espèce d'oiseau chanteur [Prather, Peters, Nowicki, et Mooney, 2008] : ils déchargent sélectivement à l'audition d'une séquence

particulière de notes produite par un autre oiseau, ou lorsque l'oiseau-sujet produit lui même cette séquence.

Depuis, on a identifié également des 'neurones canoniques' dans le cortex pré-moteur ventral (aire F5), dont l'activité est déclenchée à la fois lors de l'exécution de mouvements manuels et lors de la simple présentation d'objets dont la taille est compatible avec le type de préhension codés par les neurones [Jeannerod, 1994; Murata, Fadiga, Fogassi, Gallese, Raos, et Rizzolatti, 1997]. Les auteurs ont alors proposé que ces neurones canoniques représentent la description, en termes moteurs, des objets présentés [Murata et al., 1997] : autrement dit, ils coderaient les "affordances" des objets (i.e. propriétés des objets saillantes pour toute action dont ils sont la cible [Gibson, 1966]). Chaque fois qu'un objet serait présenté, ses caractéristiques visuelles seraient automatiquement "transcrites" en actions potentielles, celles-ci reflétant la description "pragmatique" de l'objet (i.e. comment le saisir).

Par ailleurs, on peut ajouter à cette description générale des neurones miroirs et canoniques plusieurs points que nous évoquerons plus loin :

- les neurones canoniques ou miroirs déchargent même lorsque l'action perçue est partiellement occultée. Ainsi, si le macaque sujet observe un autre macaque en train d'attraper un objet, mais que la phase finale de cette action est masquée visuellement, les neurones associés à l'action déchargent tout de même

- lorsque l'action est perçue sur un écran de télévision, aucun neurone canonique ou miroir ne décharge [Jellema, Baker, Wicker, et Perrett, 2000]

- lorsque l'action est simulée (par exemple, l'expérimentateur fait le geste d'attraper un objet dans le vide), aucun neurone canonique ou miroir ne déclenche [Gallese, Fadiga, Fogassi, et Rizzolatti, 1996].

Généralement les travaux de neurobiologie considèrent qu'un encodage est visible dans le cerveau sous forme d'une activité neuronale localisée (voire par l'activité d'un neurone spécifique), ou par un patron d'activité neuronale. L'imagerie cérébrale fonctionnelle est alors utilisée pour démontrer l'existence de ces encodages : on dira par exemple que tel neurone ou groupe de neurones "encode" la vision de tel objet. Cette conceptualisation, qui semble à première vue naturelle, pose en réalité plusieurs problèmes importants :

- si l'activation des neurones qui encodent une perception est un simple phénomène causal, pourquoi et comment se produit-elle encore de façon fiable lorsque la situation perçue est partiellement occultée ?

- comment le cerveau maintient-il (régule-t-il) la qualité de cette correspondance ? Qu'est-ce qui prouve que le déclenchement des neurones 'cheval' pour le sujet ne se modifie pas (suite à la maturation du système nerveux, au vieillissement, ou tout autre phénomène) et ne finit pas par se déclencher aussi en présence d'un âne, sans que le sujet ait conscience de l'erreur commise par son système perceptif ?

- comment le système d'encodage en est-il arrivé à représenter les objets, en les séparant du fond par exemple, ou en faisant abstraction de l'angle sous lequel l'objet est vu, ou sa distance par rapport à l'observateur ?

- pour quelle raison les yeux bougent-ils en permanence, au lieu au contraire de chercher la stabilité, qui améliorerait la qualité de la transduction 'sensations visuelles $\Longrightarrow$ neurones encodeurs'

- comment le cerveau enrichit-il son répertoire de nouveaux neurones encodeurs par rapport à de nouvelles expériences perçues ? Keysers et al. [2004] font l'hypothèse qu'ils sont le résultat de la composition de la reconnaissance "d'actions élémentaires", à l'origine non précisée. Mais alors les 2 premiers problèmes cités s'appliquent aux neurones associés à ces perceptions élémentaires : comment se déclenchent-ils et comment maintiennent-ils la sémantique de leur correspondance ?

- la majorité des auteurs supposent l'activité des neurones miroirs comme donnée et ne questionnent pas le mécanisme qui conduit à leur déclenchement

- la discussion autour des neurones miroirs porte essentiellement sur ceux à congruence stricte. Mais en réalité, beaucoup d'autres neurones voisins ont une congruence plus faible, voire réagissent à des éléments sensoriels et moteurs distincts [Csibra, 2005; Uithol et Haselager, 2008].

Ces problèmes ne sont pas des questions de détails, ils sont au cœur de la problématique de la perception. Ils témoignent de la complexité du problème et de la nécessité de se placer dans un cadre théorique clair.

5.2.3 Psychologie expérimentale

Witt et Brockmole [2012] ont montré par des expériences très parlantes comment les actions qui sont disponibles pour un sujet influencent sa perception d'objets en rapport avec ces actions. Par exemple, le fait de porter une arme modifie de façon significative le taux de reconnaissance de la présence d'une arme sur autrui [Witt et Brockmole, 2012], le fait de porter une charge lourde déforme la perception des dénivelés ou des distances dans le sens d'une majoration [Witt, 2011].

5.2.4 Imagerie mentale

Il existe maintenant de nombreuses preuves expérimentales que les images mentales ont des caractéristiques structurales et fonctionnelles qui sont similaires à celles des objets réels, cette équivalence concernant non seulement la perception visuelle au sens strict, mais aussi tous les processus visuo-moteurs [Decety et Jeannerod, 1996].

L'imagerie motrice peut être définie comme un processus dynamique durant lequel un sujet simule mentalement une action particulière. De nombreux éléments indiquent que l'imagerie motrice fait partie des mêmes processus que ceux qui sont impliqués dans la programmation et la préparation des actions réelles, l'exécution effective étant inhibée au niveau cortico-spinal [Jeannerod, 1994]. Dès 1962, Landauer a comparé le temps mis par des sujets pour réciter l'alphabet ou des suites de nombres, effectivement ou de façon intérieure, et il a montré que le sujet mettait des temps comparables dans les deux cas [Landauer, 1962].

5.2.4.1 Production et perception de l'écriture manuscrite cursive

On a longtemps considéré l'écriture manuscrite cursive comme étant simplement d'une 'fonte' particulière, sans faire de différence qualitative au niveau de sa perception avec des écritures en lettres majuscule ou même en script. Or les travaux de Longcamp et al. [Longcamp et al., 2006, 2011] ont montré sur des expériences avec imagerie cérébrale MEG et fMRI que la reconnaissance de lettres en écriture cursive activait plus fortement les zones du cortex moteur primaire gauche liées à la main que lors de la reconnaissance de lettres imprimées. Les lettres imprimées provoquent malgré tout une activité (plus faible)

de ces zones, mais on peut l'expliquer par le fait qu'une lettre imprimée est elle aussi un objet qui peut s'écrire à la main. On peut se demander si cet effet ne serait pas encore plus marqué si on réalisait des expériences de reconnaissance, non sur des lettres cursives séparées, mais sur des successions continues de lettres cursives, à cause de la coarticulation nécessaire entre les lettres dont on a vu au chapitre 3 qu'elle n'était pas figée et dépendait fortement de la dynamique de sa production oscillatoire. On voit poindre ici l'idée qui va être défendue dans ce chapitre : la lecture d'un texte cursif passe par la simulation intériorisée de sa production motrice.

5.3 Cadres théoriques

Nous allons chercher à démontrer que la perception n'est pas cette simple phase d'acquisition d'information qu'elle est souvent et implicitement supposée être. Cet état de fait s'explique sans doute par l'histoire des théories de la perception, que nous allons d'abord évoquer brièvement.

5.3.1 Le nativisme et la perception

Le nativisme est un mouvement théorique qui naît dans les années 70-80 en réaction au behaviorisme des années 50, et dont les principaux théoriciens sont Noam Chomski et Jerry Fodor. La position nativiste s'est affirmée clairement durant le débat Piaget-Chomski en 1975 [Piattelli-Palmarini, 1980] : "All structure comes from within. Environment reveals this structure ; it does not imprint its own patterns on the system. [...] forms [...] preexist [...] and are forced to materialize under appropriate conditions". Le nativisme reconnaît qu'il existe une interaction complexe entre les facteurs innés et les facteurs environnementaux, mais le point crucial est que les structures qui forment la connaissance et la compétence de l'agent, sont intrinsèques à l'agent lui-même. Contrairement à ce qu'on entend souvent, quand on demande si une connaissance ou une compétence est innée, on ne doit pas montrer qu'elle est indépendante des facteurs environnementaux, mais plutôt montrer qu'elle ne peut pas être apprise.

En ce qui concerne la perception, elle est donc considérée par les nativistes comme une évocation, sous forme de transduction sensorielle, de concepts innés.

5.3.2 L'empirisme et la perception

L'empirisme classique est une doctrine philosophique ancienne, dont les principaux penseurs ont été David Hume et Francis Bacon. À l'inverse du nativisme, elle soutient que toute forme de connaissance prend sa source dans l'expérience du réel, et imprime sa forme dans l'esprit du sujet. Les progrès de la science et de la technologie ont donné des noms plus complexes à ce mécanisme d'empreinte : "transduction énergétique" pour une représentation dans l'instant, "induction" et "abstraction" pour une mémorisation et un apprentissage à plus long terme, mais le mécanisme essentiel reste un transfert direct de la structure du monde perçu vers l'agent.

En résumé, la source de toute forme de connaissance se situe, soit uniquement dans l'agent lui-même pour le nativisme, soit au contraire uniquement dans le monde extérieur pour l'empirisme.

5.3.3 La conception traditionnelle de la perception comme phase d'acquisition de l'information : l'encodage

Étant donné que nativisme et empirisme sont à deux extrémités d'une même échelle en ce qui concerne l'origine des connaissances possédées par un sujet, et en particulier de toute représentation perceptive, on pourrait être tentés de choisir un des camps, ou de prendre une position intermédiaire, dans laquelle certaines connaissances auraient une source innée, tandis que d'autres seraient totalement apprises. Par exemple, le nativisme 'Mad Dog' de Fodor [1975] propose l'existence d'un grand nombre de concepts innés ; le 'nativisme développemental' suppose l'existence d'un petit nombre de concepts fondamentaux innés [Chomsky, 1965; Spelke et Breinlinger, 1992; Gallistel et Gelman, 1992].

Bickhard [2009] montre que, même si nativistes et empiristes semblent totalement opposés, ils partagent en réalité un présupposé commun, généralement non explicité : ils considèrent que la forme fondamentale de connaissance ou de représentation est un encodage, une relation qu'on peut symboliser sous la forme suivante : élément de connaissance $\Longrightarrow$ codage.

L'élément de connaissance est la chose représentée, (par exemple, le concept 'chien', sous forme abstraite ou sous forme perçue) et le codage est situé dans l'agent, par exemple sous forme d'activité d'un groupe de neurones. Bien sûr, des connaissances complexes peuvent être des composés de ces éléments, mais ces derniers ont pour forme fondamentale un encodage.

Ainsi, nativistes et empiristes se disputent sur l'origine interne ou externe de ces représentations, mais ne se demandent pas si leur forme d'encodage capture la nature fondamentale de la représentation. La majorité des auteurs se revendiquent plus ou moins nativistes ou empiristes, mais en acceptant implicitement l'encodage comme la forme fondamentale de représentation des connaissances. Beaucoup d'articles et d'ouvrages utilisent directement les expressions "encode", "constitue une représentation de" etc. sans décrire le cadre théorique sous-jacent. La perception est ainsi vue comme un simple processus d'acquisition de l'information, à sens unique, allant de l'extérieur (la scène) vers l'intérieur, qui construit "dans" le sujet perceptif une "représentation interne" de la situation externe.

Bickhard qualifie de 'fondationnalistes' de tels modèles, et il montre comment les encodages ne peuvent pas constituer la forme fondamentale de connaissance pour un agent, sous peine de multiples contradictions logiques. Une première ligne d'arguments concerne le problème de l'émergence de nouvelles formes de représentation [Bickhard et Terveen, 1995] :

- Dans le cadre fondationnaliste, la perception consiste en la simple évocation (transduction sensorielle) d'encodages associés. Par exemple, la vision d'un chien va déclencher l'activité des neurones associés à la reconnaissance de cet animal.

- L'apprentissage consiste alors à construire des assemblages nouveaux (par association, induction, etc.) d'encodages plus élémentaires. Toute nouvelle représentation, aussi complexe soit-elle, est formée des mêmes éléments de base en termes d'encodages.

- Dans un tel cadre, l'émergence d'éléments de représentation véritablement nouveaux est impossible. Il n'y a pas de possibilité de processus constructif qui permette de faire émerger un élément nouveau dans l'alphabet des encodages élémentaires.

Ce raisonnement est valable au niveau du développement du sujet, mais également au niveau de l'évolution de l'espèce : un système de représentation basé uniquement sur des

encodages ne peut pas émerger, car il ne peut pas créer de nouveaux éléments de représentation. Cela ne signifie pas que les encodages ne sont pas des formes de représentation valables : ils existent clairement, par exemple au travers des systèmes linguistiques. Mais cela signifie qu'une autre forme de représentation, plus fondamentale, est nécessaire.

Une deuxième ligne d'arguments de Bickhard concerne l'épistémologie : il montre comment les encodages ne peuvent pas à eux seuls être les éléments de connaissance du point de vue de l'agent lui-même [Bickhard et Terveen, 1995] :

- Pour un encodage donné, notion $\Longrightarrow$ codage, le sujet a besoin d'un interprète pour s'assurer que la relation est vraie ou fausse à un moment donné. Par exemple, si des observateurs extérieurs utilisant des méthodes d'imagerie cérébrale analysent l'activité du cerveau d'un sujet qui regarde un chien (= la notion), ils vont constater que les neurones associés s'activent (= le codage). Ils utilisent leur propre cognition pour confirmer que la relation entre la présence du chien et l'activité du groupe de neurones est bien valide. Si la situation est perceptivement moins claire (image brouillée, plus petite, etc.), c'est eux qui peuvent dire si la relation reste valide ou non.

- Le sujet, quant à lui, n'a pas d'interprète à sa disposition, autre que lui-même. Mais si on suppose que toutes ses représentations sont des encodages, il n'aura aucun moyen à sa disposition pour s'assurer de la signification de l'activité d'un encodage. Si l'image brouillée d'un renard déclenche l'activité des neurones « chien », comment le sujet va-t-il savoir qu'il ne s'agit pas d'un chien ? Il manque une voie de retour, une façon pour le sujet d'interroger la situation.

Ainsi donc, pour les nativistes comme pour les empiristes, la perception est une étape passive d'acquisition et de traitement d'information, une étape d'entrée, qui n'a rien à voir avec les actions du sujet qui perçoit, considérées comme faisant partie d'une étape de sortie.

Une troisième voie, que nous allons défendre, considère au contraire que c'est au travers de l'interaction entre un sujet et son environnement que celui-ci construit sa connaissance du monde. Ce type d'approche a commencé à être défendu par des philosophes dits 'pragmatistes' tels que William James, mais l'auteur le plus significatif du 20ème siècle est incontestablement le psychologue et épistémologiste suisse Jean Piaget.

5.3.4 Jean Piaget

Jean Piaget propose que la connaissance émerge au travers de l'interaction entre l'agent et son environnement [Piaget, 1954, 1967; Piaget et Duckworth, 1970]. À partir d'un petit ensemble de compétences sensori-motrices innées disponibles à la naissance, Piaget a décrit dans le détail l'apprentissage de nouvelles compétences et connaissances [Piaget, 1954]. Mais le point crucial, la principale différence avec le nativisme et l'empirisme, c'est qu'il a décrit comment peuvent émerger de nouvelles formes de représentation à partir d'états qui ne les contenaient pas, et c'est l'action, l'activité du sujet qui est la force principale de cette construction.

Les connaissances et compétences construites par le sujet émergent de l'interaction des ses structures de comportement avec l'environnement. Par exemple, les structures sensori-motrices permettant la succion du sein par l'enfant à la naissance, après avoir été confrontées plusieurs fois fortuitement à la succion des doigts, vont créer et stabiliser une variante de la succion adaptée aux doigts. Cette variante, devenue distincte de l'original, constitue une forme de représentation active des doigts par la bouche [Piaget et Duckworth, 1970].

Après d'autres constructions analogues, les doigts deviendront le centre d'un réseau d'interactions visuelles, buccales etc. Cette représentation interactive des doigts est nouvelle, autonome et elle a émergé de l'interaction des structures antérieures avec l'environnement ; elle a un sens pour l'enfant (et pas seulement pour un observateur extérieur) puisqu'elle lui permet d'utiliser ses doigts et sa bouche. Dans tout ce processus, c'est l'activité du sujet, au travers du fonctionnement de ses structures sensori-motrices, qui :

- fournit la force d'exploration de nouvelles possibilités,
- permet au sujet d'interroger la situation pour la désambiguïser ou la préciser, ou pour s'ajuster par rapport à elle.

Une forme typique de perception basée sur l'action est celle d'un aveugle utilisant un bâton : s'il reste immobile, il ne peut avoir connaissance que des événements qui vont fortuitement à sa rencontre. Si au contraire il est actif et interroge la scène autour de lui, tout change. En anticipant la présence d'un objet, il va interroger ses contours, le faire résonner, etc. pour confirmer ou infirmer telle ou telle hypothèse.

Nous avons mis en application cette idée de perception par l'action dans un algorithme de conduite automobile. Cet algorithme permet de diriger une voiture de course, dans un environnement virtuel, à une vitesse constante modérée[1]. La mise en place de l'environnement ayant permis de tester cet algorithme est présentée en annexe D. L'idée de base de cet algorithme est très simple : à la manière d'un aveugle, on fait osciller un point perceptif devant la voiture afin d'acquérir de l'information sur la situation. Si ce point oscillant cogne le bord de la route, alors, l'algorithme change la direction de la voiture en conséquence. Cet algorithme, simpliste, donne des résultats étonnement bons. Il est à noter qu'ici le point perceptif bouge de manière automatique. Il pourrait être intéressant de modifier l'algorithme pour que le mouvement de celui-ci soit aussi influencé par le besoin d'informations à certains endroits particuliers, déterminés par l'algorithme.

L'action dans la perception fournit une voie de retour indispensable pour un ajustement fin à la situation perçue, de la même façon qu'une boucle de rétroaction est nécessaire dans un modèle de contrôle pour réguler le degré d'un paramètre par rapport au maintien d'un objectif.

Un tel type de perception active est bien à l'œuvre dans l'utilisation de nos sens habituels. C'est clairement le cas du sens du toucher, plus de choses sont perçues si l'on déplace la main et les doigts sur les objets. La vision a toujours été considérée comme l'analogue d'une caméra, mais les expériences sur la "perceptual blindness" [O'Regan et Noë, 2001] ont montré que la vision était plutôt une forme de toucher à distance. Les mouvements des yeux, bien que perturbant la projection de l'image rétinienne, permettent au sujet d'interroger la scène, de confirmer ou préciser tel ou tel aspect, en bref de se synchroniser sur la situation. L'image rétinienne est en mouvement continuel (3 saccades par seconde en moyenne), avec de nombreuses aberrations chromatiques et elle n'est nette qu'au centre, avec une tache aveugle due au nerf optique. Or nous percevons une image stable, nette, bien colorée : ne serait-elle pas le résultat d'une construction interne ? C'est cette idée que nous allons défendre dans le cadre de la perception visuelle de l'écriture manuscrite.

5.3.5 La théorie de la cognition par simulation de l'action

L'idée principale de cette théorie est que la pensée est une interaction simulée avec l'environnement (pour une revue complète et récente, voir [Hesslow, 2012]). Trois types d'argu-

[1] Une vidéo de cet algorithme en fonctionnement est disponible à cette adresse : http://www.youtube.com/watch?v=gWYaj47UZRg.

ments supportent cette théorie :

5.3.5.1 Simulation de l'action

Les primates (et sans doute d'autres espèces) sont capables d'activer leurs structures motrices, comme lors de la réalisation effective d'une action, mais en supprimant son exécution.

Des expériences déjà anciennes ont par exemple montré que la durée de réalisation mentale simulée d'une action de rotation d'un objet tridimensionnel était identique à celle effectuée réellement, et augmentait linéairement avec l'angle de rotation [Shepard et Metzler, 1971]. Kosslyn, Ball, et Reiser [1978] ont montré que le temps de déplacement du regard dans une tache de vision simulée était proportionnel à la distance à parcourir, de façon identique à celle d'un déplacement réel du regard.

Dans le domaine de l'écriture manuscrite, Decety et Michel [1989] ont montré que le temps d'écriture d'une phrase courte de façon réelle ou mentalement simulé était le même, et diminuait ou augmentait de façon identique en changeant de main (réelle ou simulée). Par ailleurs, cette durée restait constante qu'on demande au sujet d'écrire (réellement ou mentalement) avec des lettres petites ou grosses, c'est à dire que l'écriture simulée préserve le principe d'isochronie. Sur une tâche de déplacement, Decety et Jeannerod [1996] ont montré que la loi de Fitts, qui exprime une relation entre la difficulté d'une tâche en terme de précision et son temps d'exécution, est préservée lorsque celle-ci est réalisée de façon simulée.

Depuis, ces résultats ont été confirmés par l'imagerie cérébrale dans de nombreux domaines. Par exemple chez un pianiste professionnel, jouer un passage ou s'imaginer le jouer active les mêmes aires du cortex frontal et pariétal [Meister, Krings, Foltys, Boroojerdi, Müller, Töpper, et Thron, 2004].

5.3.5.2 Simulation de la perception

Des expériences utilisant l'imagerie cérébrale ont confirmé que le cortex visuel primaire était fortement activé lorsqu'on demande à des sujets d'imaginer des stimuli visuels ou de se rappeler une expérience visuelle [Kosslyn, Alpert, Thompson, Maljkovic, Weise, Chabris, Hamilton, Rauch, et Buonanno, 1993]. De même, le cortex auditif est activé lorsque des sons sont imaginés [Schürmann, Raij, Fujiki, et Hari, 2002].

Par ailleurs, ces perceptions simulées ont les conséquences analogues aux stimuli réels. Par exemple, des sujets qui imaginent effectuer des exercices avec leurs jambes augmentent leur rythme cardiaque [Decety, Jeannerod, Germain, et Pastene, 1991].

5.3.5.3 Anticipation

Lors de l'activation du cortex moteur, soit lors d'une action réelle, soit lors d'une action simulée, le cortex perceptif est activé en fonction des conséquences les plus probables de l'action réalisée. Ainsi, des patients qui viennent d'avoir une attaque hémiplégique et qui ont perdu l'usage d'un bras, ont l'impression de ressentir son mouvement lorsqu'ils essaient de le bouger, au point que certains sujets pensent avoir encore le contrôle de leur membre [Feinberg, Roane, et Ali, 2000]. Également, si on demande à des sujets de mémoriser des objets présents dans une scène, puis de les remémorer ensuite à haute voix devant un écran blanc, leurs yeux tendent à se déplacer comme s'ils suivaient les objets de la scène, alors que rien n'est visible, et ces actions améliorent la performance de la tâche.

Cette théorie de la cognition par simulation de l'action est ontologiquement parcimonieuse ; elle met également l'action du sujet, réelle ou intériorisée, au centre de sa cognition. La simulation permet d'expliquer sans faire référence à des concepts supplémentaires,

l'existence d'une mémoire déclarative ou la conscience d'une vie intérieure imagée [Heslow, 2012].

5.3.6 Mark Bickhard et l'interactivisme

Le modèle interactiviste de Mark Bickhard [Bickhard, 2009; Bickhard et Terveen, 1995] est une approche de la représentation et de la cognition basée sur l'action qui rejoint la théorie de l'intelligence de Jean Piaget sur de nombreux points fondamentaux, avec quelques différences importantes. La nature fondamentale de la représentation selon Bickhard n'est pas un encodage ou une correspondance, mais une anticipation : les anticipations peuvent être fonctionnelles et ne sont pas nécessairement basées sur des formes représentatives déjà disponibles. De la sorte, les anticipations peuvent émerger de façon constructive. Les anticipations, décrivent comment une interaction de l'agent va se dérouler, étant donné le contexte courant, si l'agent choisit d'effectuer cette interaction.

Par exemple, une grenouille discrimine l'environnement à la recherche de petits objets passant dans son champ visuel, lui donnant l'opportunité de projeter sa langue et d'avaler. Cette discrimination est purement fonctionnelle et la grenouille n'a pas besoin pour cela d'avoir une représentation explicite des mouches ou d'autres objets. D'ailleurs, elle lance également sa langue si de petits cailloux sont jetés devant elle.

Ce type de représentation anticipative possède une propriété essentielle : la normativité, c'est à dire la capacité d'avoir une valeur de vérité vraie ou fausse. Par exemple, si de petits cailloux sont jetés devant la grenouille, celle-ci va projeter sa langue, mais la suite de l'interaction ne se déroulera pas comme prévu, les cailloux ne pouvant être mangés normalement. Dans cette situation, la représentation interactiviste associée à l'ensemble de l'interaction, de la détection des objets jusqu'à l'avalement, est fausse, en erreur. Et le plus important, c'est que le sujet lui même peut détecter cette erreur, sans supposer l'existence d'une structure supplémentaire d'évaluation, d'un interprète.

L'interactivisme partage avec Piaget la même conception de ce qu'est un objet pour un agent : un réseau fermé d'interactions (visuelles, tactiles, etc.) qui est construit progressivement durant la petite enfance et qui atteint sa forme mure à environ 9 mois. Avant cette période, les objets pour les enfants n'ont pas encore acquis toute leur permanence, ce qui explique par exemple qu'ils n'iront pas soulever un foulard sous lequel on a caché devant eux un objet qui les attire.

Durant les vingt dernières années, plusieurs auteurs ont cherché à démontrer que les enfants possédaient ces notions bien avant les périodes décrites par Piaget [Baillargeon, Spelke, et Wasserman, 1985; Baillargeon, 1991], mais on a mis en évidence récemment que les méthodes employées étaient sujettes à caution [Schöner et Thelen, 2006]. Pour l'interactivisme, la perception consiste à mettre à jour les possibilités d'interaction offerte par l'environnement, et qui seront exploitées par d'autres processus de plus haut niveau. Trois exemples tirés de [Bickhard, 2009] feront mieux comprendre ce phénomène :

- La perception d'un cube par un enfant consiste à mettre à jour les actions visuelles et tactiles possibles dans la situation courante. Cette représentation est simplement anticipative et ce sont d'autres processus qui mettront en action ou pas ces potentialités.

- La perception d'une ligne ou arête comme étant droite n'est pas le constat statique de l'existence d'une suite de capteurs rétiniens activés. Les capteurs activés auront une forme irrégulière, de part et d'autre de la ligne, mais un scan visuel le long de la ligne va garder invariant ce patron d'activité, et c'est ce processus préservant l'invariant qui constitue la perception de la ligne, d'une extrémité à l'autre.

- De façon analogue, et comme décrit également par [O'Regan et Noë, 2001], percevoir qu'une partie de la scène est rouge n'est pas une détection statique d'une composition de couleur, qui ne fonctionnerait d'ailleurs pas avec des conditions d'éclairage différentes. Comme pour la ligne droite, déplacer le regard au dessus d'une zone rouge génère des invariants dans l'activité des récepteurs rétiniens, et c'est anticiper la présence de ce groupe d'invariants qui constitue la perception du rouge.

Ainsi pour l'interactivisme de Bickhard, la perception est interactive, et anticipative. Elle met à jour les possibilités d'interaction qu'offre la situation, au service des processus de plus haut niveau.

5.4 Un modèle computationnel interactiviste de perception de l'écriture cursive manuscrite par simulation de l'action

La discussion théorique précédente va conduire dans ce paragraphe à la présentation d'un modèle computationnel de perception de l'écriture manuscrite cursive. Il contient en germe les éléments fondamentaux du cadre interactiviste présenté précédemment. Ce qui nous intéresse ici n'est pas de produire un algorithme particulièrement performant de reconnaissance de l'écriture manuscrite, mais plutôt d'aborder ce problème à l'aide d'une méthode basée sur un cadre théorique différent.

5.4.1 Modèle interactiviste de perception de l'écriture manuscrite cursive

La perception visuelle d'un échantillon d'écriture manuscrite cursive est considérée comme un processus intériorisé de simulation de cette écriture, compatible avec les éléments sensoriels présents. Cette compatibilité est évaluée en projetant l'écriture simulée et l'image réelle sur un espace commun, et en évaluant à quel point l'image simulée est incluse dans l'image réelle. La perception de l'écriture est ainsi le résultat d'une assimilation entre une activité intériorisée de production et les éléments visuels présents.

Supposons par exemple que les formes possibles à reconnaître par le sujet soient certaines lettres de l'alphabet, ou des enchaînements de celles-ci. La perception consistera à chercher parmi les formes possibles de cette grammaire celle qui correspond aux traces visibles sur la feuille. Plus précisément :

- recenser les débuts possibles des séquences motrices permises. Ici par exemple, il s'agit des séquences motrices associées à n'importe quelle lettre de notre alphabet. On peut imaginer de les ordonner par un critère probabiliste.

- Pour chaque début de séquence motrice considérée, la tracer dans un espace vierge (une 'image intérieure' à une certaine origine, avec une certaine couleur et épaisseur de trait, et un certain facteur d'échelle,

- utiliser cette image intérieure comme un 'calque' à placer au dessus de la projection de l'image réelle,

- évaluer à quel point cette construction est compatible avec l'image réelle, et correspond aux éléments sensoriels,

- optimiser localement cette évaluation en faisant varier les paramètres de la représentation interne choisie afin de rendre l'image réelle et l'image intérieure plus compatibles.

- si une représentation motrice interne est compatible avec le début d'un trace sur l'image réelle, alors on peut tenter de poursuivre la reconnaissance en enchainant une autre lettre à la représentation interne.

Cette tâche perceptive semble complexe d'un point de vue informatique, avec de nombreux éléments à rechercher dans des espaces de grandes tailles, mais la poursuite des parties viables dans l'arbre des formes possibles va élaguer la plupart des branches, et il sera rare de suivre plus d'une branche à la fois. On peut confondre le début de "lu.." et de "eu.." si le premier « l » ou « e » ont des dimensions voisines, mais l'anticipation de la présence des lettres suivantes va très vite permettre de choisir la branche correcte.

La méthode est interactiviste au sens de Bickhard car les éléments de représentation sont constitués de successions d'anticipations, qui peuvent être confirmées ou infirmées. Elle place l'activité du sujet, par son mécanisme de production, au centre du processus de perception.

5.4.2 Application du modèle à une tâche de reconnaissance off-line d'écriture manuscrite cursive

Nous allons présenter maintenant deux implémentations partielles du modèle présenté ci-dessus, chacune reprenant certains de ses aspects. Le but de ces implémentations est de montrer qu'un tel modèle interactiviste permet de résoudre certains problèmes mal pris en compte par les approches usuelles de la reconnaissance de caractère hors-ligne de l'écriture manuscrite. Il ne s'agit pas ici de fournir des algorithmes de reconnaissance de l'écriture prêts à l'emploi mais plutôt d'apporter des idées qui pourraient être prises en considération lors de l'élaboration d'un système de reconnaissance de l'écriture manuscrite hors-ligne.

Ces deux implémentations reposent sur la même idée de base, à savoir la comparaison d'une représentation interne, s'appuyant sur la génération du mouvement, avec l'image réelle. Ces deux implémentations ont été testées sur les mêmes images, générées à partir du tracé de caractères sur la tablette graphique. Pour chacun de ces tracés, tant l'image résultante que la suite des positions datées sont disponibles.

5.4.3 1re implémentation : représentation interne basée sur POMH

Le but de cette implémentation, au-delà de la reconnaissance de caractères, est de retrouver la dynamique du caractère à reconnaitre : le but est de changer la représentation de ce dernier constituée d'un groupe de pixels sur une image en une suite de positions datées. Si ce but est atteint, il sera alors possible d'appliquer un algorithme de reconnaissance en-ligne, dont les taux de reconnaissance sont meilleurs [Plamondon, 2000]. Par ailleurs ces derniers pourront bénéficier a priori de la nature probable du caractère à reconnaître.

Cette méthode, s'appuie sur une représentation générative des caractères. Puisque nous avons présenté POMH dans cette thèse, c'est évidemment ce modèle que nous utiliserons comme représentation interne des caractères (notez que POMH pourrait-être remplacé par n'importe quel modèle génératif, peut-être même de manière avantageuse). A plusieurs égards, notre algorithme ressemble à un algorithme génétique. D'abord, dans une étape initiale, on construit des représentations internes initiales pour chaque classe de lettre à reconnaître. Puis, de manière itérative, on compare ces représentations à l'image réelle au moyen d'une fonction d'adaptation. Pour chaque classe, les représentations (appelées

individus) les plus adaptées sont conservées, les autres sont, ou remplacées, ou mutées. Par simplification, on suppose que pour chaque caractère, la taille et la position de ceux-ci sont similaires.

5.4.3.1 Présentation de l'algorithme

Étape 1 Pour chaque classe de caractère à reconnaître, la représentation particulière de l'un des éléments de cette classe est donnée, grâce à l'algorithme d'extraction de POMH, à partir d'une trace enregistrée. Puis pour chaque classe, N individus sont générés par mutation. C'est à dire que pour chaque individu, pour chacun de ces paramètres oscillatoires (a_x, a_y, ω_x, ω_y, phi_x et phi_y) il y a une probabilité p de modification aléatoire de celui-ci, selon une loi normale dont la forme est choisie empiriquement.

Étape 2 Chacun des individus de chaque classe est évalué par confrontation avec l'image réelle. Pour ce faire, ils sont d'abord géométriquement reconstruits (voir section 3.1.2.3). A ce moment, chacun est représenté par une suite de positions datées. Ils sont ensuite évalués par rapport au caractère perçu (qui est sous forme d'un tableau de pixels), grâce à une fonction d'adaptation qui sera détaillée par la suite.

Étape 3 Dans chaque classe, les n meilleurs individus sont conservés, puis, la moitié des individus restants sont remplacés par des mutants de ces n meilleurs individus. Les autres individus sont simplement mutés. Puis, on recommence à l'étape 2 tant que la valeur de l'adaptation donnée par le meilleur individu n'a pas atteint un niveau satisfaisant ou que le nombre d'itérations n'a pas atteint une valeur maximale.

La mise en place d'un tel algorithme est simple, mais comme toujours avec ce type de méthode, le dosage est primordial. En effet, les valeurs, n, N et la probabilité de mutation peuvent avoir des conséquences notables sur la qualité des solutions trouvées.

Cependant, le plus important reste le calcul de l'adaptation, qui peut avoir des effets dramatiques et inattendus sur la sélection des individus. Pour ce calcul d'adaptation, il s'agit de comparer une suite de positions datées et un nuage de points. Plusieurs calculs d'adaptation ont été envisagés (et implémentés). Nous donnons ici celui qui nous parait être le plus pertinent d'un point de vue théorique. Pour ce calcul particulier, trois paramètres nouveaux sont ajoutés à chaque individu : x_0 et y_0 qui représentent le point de départ de la trace et *thickness* qui représente l'épaisseur du trait utilisé dans le calcul d'adaptation. Ces trois variables sont elles aussi soumises à la mutation. Le calcul de l'adaptation d'un individu suit les étapes suivante :

- L'individu est tracé en blanc sur une image de fond noir de même taille que l'image réelle avec une épaisseur de trait égale à *thickness*
- L'image réelle est binarisée, inversée, puis multipliée par l'image précédemment générée. L'image résultante ne contient donc que les points communs entre les deux traces de chacune des images multipliées.
- L'adaptation est finalement donnée par la somme des pixels blancs de cette image divisée par *thickness*.

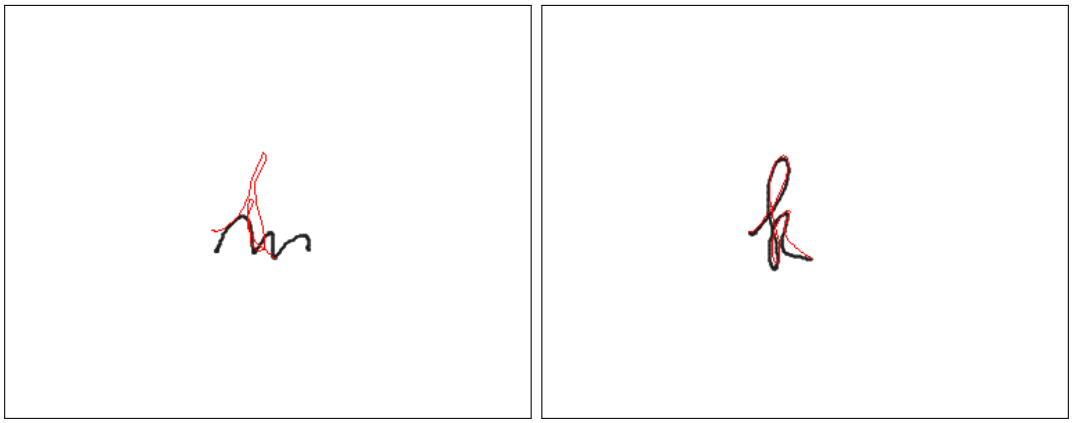


Figure 5.1 – Résultat de cette première implémentation sur deux lettres. La première solution trouvée pour le "m" est catastrophique, celle trouvée pour le "k" est meilleure.

5.4.3.2 Observations

Cette méthode ne donne pas de bons résultats et ne permet que rarement de produire des solutions satisfaisantes. La figure 5.1 montre deux exemples de solutions trouvées pour deux lettres. En général, la meilleur solution n'évolue pas au cours du temps, signifiant que les mutations apportées à chaque itération n'ont pas vraiment d'effets bénéfiques.

Il est à noter que cet algorithme ne peut pas vraiment être considéré comme un algorithme de reconnaissance de caractère, puisque même si les individus de base sont issus de différentes lettres, leurs mutants ne seront pas forcément ressemblants à la lettre de départ et pourront peut être mieux s'adapter à d'autres lettres. Il s'agit avant tout d'essayer de trouver la suite des paramètres dynamiques codant la trace écrite dont seule l'image nous est fournie.

5.4.4 2e implémentation : représentation interne basée sur une image "temporalisé"

5.4.4.1 Présentation de l'algorithme

Pour chaque classe de caractère à reconnaître, on dispose de l'image interne d'un caractère particulier de cette classe ainsi que de sa représentation temporelle (la suite des positions datées ayant produit cette image).

Afin de reconnaître l'image d'un caractère donné, cette dernière ainsi que toutes les images internes se voient superposer une grille. Les positions datées de chaque représentation interne permettent d'établir, pour chacune, une suite de cases empruntées par la trajectoire du tracé. Alors, chronologiquement, pour chaque représentation interne, chaque case est comparée à la case correspondante sur l'image réelle. Une fonction d'adaptation permet pour chaque case de donner un coût qui est ajouté au coût total de chaque représentation interne. A chaque instant, la représentation interne ayant le coût total le moins élevé est celle qui anticipe le mieux le caractère présenté.

La figure 5.2 présente une visualisation du fonctionnement de l'algorithme (cliquez sur celle-ci pour faire défiler les images, si vous lisez une version électronique de cette thèse avec une visionneuse de documents pdf propriétaire). En grand, au centre de l'image, est la lettre à reconnaître. Sur l'exemple on voit qu'elle a été bruitée pour tester la résistance de l'algorithme aux perturbations. Les images du bas représentent certaines des images internes de notre méthode. Sur toutes les images on a superposé une grille de même taille.

Les cases colorées se déplacent temporellement sur la grille pour suivre la dynamique de la trace interne. Par exemple, la case orange va suivre, à chaque étape, le sens du tracé de la lettre o. La case correspondante est aussi tracée sur l'image à reconnaître. La portion d'image encadrée par la case orange sur l'image interne et la portion correspondante sur l'image réelle sont ensuite comparées grâce à une fonction d'adaptation qui va être présentée plus bas. C'est cette fonction qui calcule le coût, pour une case donnée, qui sera ajouté au coût total associé à la représentation interne anticipant la lettre 'o'.

La fonction d'adaptation utilisée pour comparer deux patchs s'effectue en trois étapes. Premièrement, pour le patch associé à l'image interne, on associe un score à n directions du plan uniformément réparties. Pour associer un score à une direction, on sonde l'image en sommant l'ensemble des pixels inclus dans les bandes d'une largeur donnée. On retient comme score pour la direction, le maximum des scores associés à chaque bande. De cette étape on retient trois valeurs : $dirScore$, $dirMax$ et $dirSum$ qui correspondent respectivement à la direction ayant obtenue le meilleur score, à son score, et à la somme totale des scores obtenu pour toutes les directions.

On calcule ensuite le score $dirScoreRelle$ correspondant à la direction $dirMax$ sur la case correspondante de l'image réelle selon la même méthode. On calcule aussi $dirSumRelle$: le score total obtenu par toutes les directions sur la portion d'image réelle. Finalement le coût C correspondant à la comparaison entre ces deux patchs sera donné par :

$$C = |dirScore - dirScoreRelle| \frac{\max(1, dirSum)}{\max(1, dirSumRelle)}. \tag{5.1}$$

Le dernier rapport de cette équation permet d'être moins sensible au bruit et aux croisements. Si il y a un croisement dans l'image originale, et que la mauvaise direction est sélectionnée par $dirMax$ sur le patch de l'image interne, et que le patch de l'image réelle s'avère suivre elle aussi cette mauvaise direction, alors cette image interne sera avantagée à cause d'une erreur, il faut donc moduler cet effet en multipliant par $\max(1, dirSum)$. Dans l'autre cas, si l'image réelle a aussi ce même croisement, il faut ne pas trop pénaliser l'erreur, d'où la division par $\max(1, dirSumRelle)$. Le fait de sélectionner sur l'image interne et de n'évaluer l'image réelle que sur cette direction permet une meilleure résistance au bruit.

5.4.4.2 Observations et améliorations envisageables

La première observation rassurante est que si une image interne de lettre est utilisée comme image réelle à reconnaitre, alors elle se reconnait à coup sûr (ce qui n'était pas forcément le cas avec l'implémentation précédente). Deuxièmement, si cette image est bruitée, il y a aussi de fortes chances pour que la lettre soit aussi correctement reconnue.

De manière évidente, notre algorithme est sensible aux décalages et aux déformations. Pour contrevenir à ce problème, il faudrait faire intervenir un certain nombre de prétraitements utilisés usuellement dans ce genre d'applications. Notons en revanche qu'il n'y aurait pas besoin d'un algorithme de séparation de caractères. En effet dans notre cas, si l'on arrive à la fin d'une image interne et que celle-ci a une bonne adaptation, il est possible d'enchaîner directement la reconnaissance de la lettre suivante en faisant correspondre le dernier patch de cette lettre au premier patch des images internes cherchant à anticiper cette deuxième lettre.

Comme tout algorithme de reconnaissance de caractère, cette méthode pourrait bénéficier de connaissances a priori, telles que la probabilité de présence d'une lettre dans un texte, la probabilité d'enchaînements de paires de lettres, et aussi de l'existence ou non du mot en construction dans un mot. Une connaissance préalable du champ lexical du

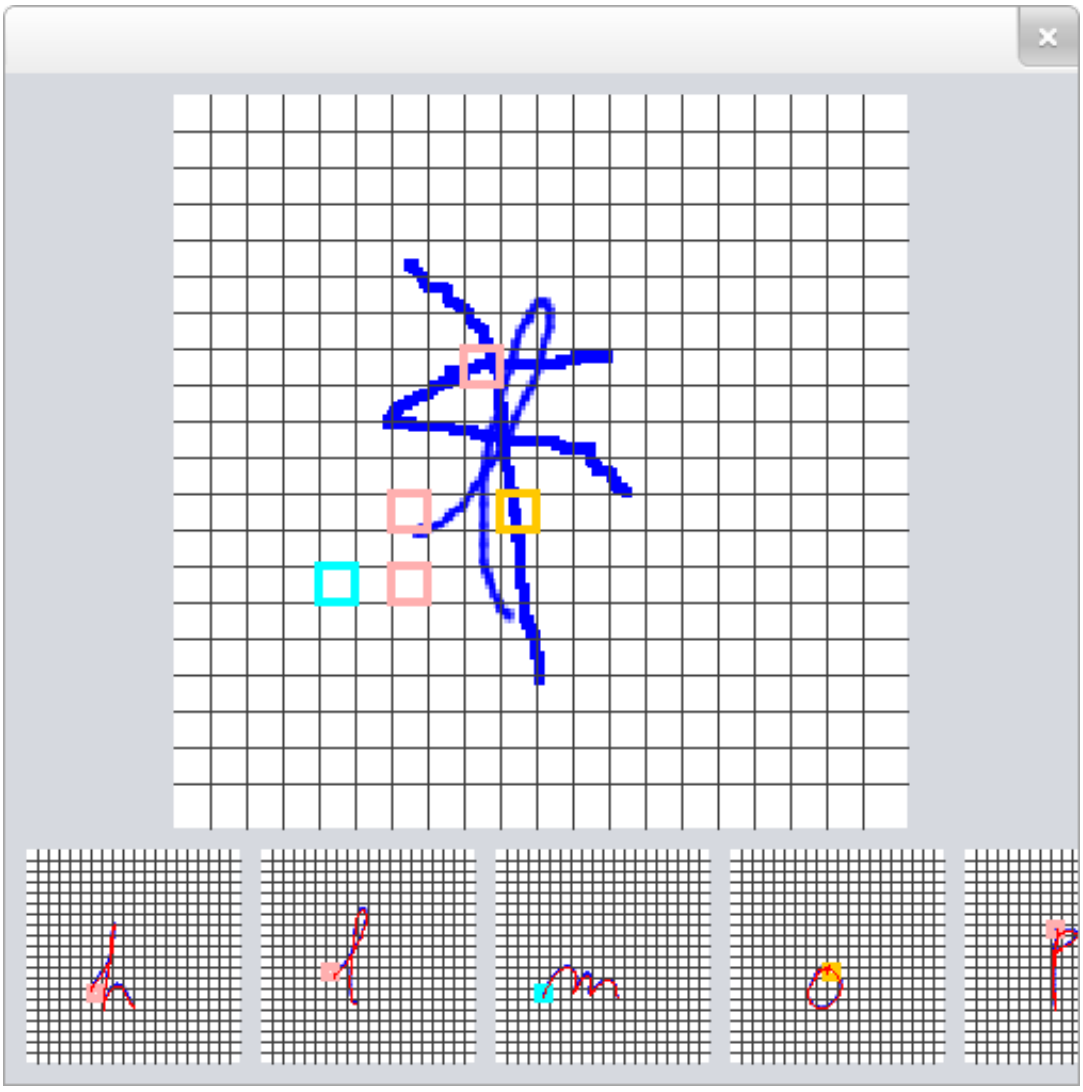


Figure 5.2 – Animation de l'algorithme en fonctionnement. L'image au centre est l'image perçue. Le bandeau d'images en base représente les images internes. Pour chaque lettre interne, une case colorée suit la temporalité de celle-ci sur sa représentation interne et sur l'image réelle. La lettre à reconnaître est bruitée, pourtant l'algorithme arrive à reconnaître le "l".

texte et des constructions grammaticales autorisées permettraient également d'améliorer les résultats.

5.4.5 Discussion

Nous avons proposé deux algorithmes de reconnaissance de caractère dont les résultats semblent mitigés. Le second, qui semble plus prometteur, aurait besoin d'être intégré à une suite de reconnaissance de l'écriture manuscrite pour que l'on puisse statuer sur son utilité applicative. En particulier, une étude quantitative de la qualité de la reconnaissance est nécessaire.

Cependant, on remarque bien qu'il nous apporte d'emblée une certaine résistance au bruit. Cela est dû à l'ancrage actif de la perception ; seules les zones dignes d'intérêt sont prises en compte. Peu importe ce qu'il se passe en dehors de ces zones. Les approches usuelles qui consistent à tenter de comprendre l'ensemble du flux perceptif sont de fait plus sensibles au bruit. Dans notre exemple, seule une petite portion du flux perceptif suivant une représentation dynamique interne de l'objet à reconnaître est appréhendée, ce qui simplifie les choses. La notion d'apprentissage n'est pas présente dans ce second algorithme. Si les caractères à reconnaître sont trop éloignés de l'image interne de celui-ci, alors la reconnaissance n'aura pas lieu.

Théoriquement, la première implémentation est plus intéressante. D'une part, la représentation interne des lettres est purement basée sur un modèle génératif, ce qui en fait une représentation basée sur l'action. D'autre part, la notion d'adaptation est présente : les mécanismes de mutation devraient permettre l'émergence de représentations internes capables d'anticiper n'importe quelle trace (si l'on n'impose aucune contrainte de temps). Hélas, en l'état actuel de l'implémentation, ce résultat n'est pas obtenu en un temps raisonnable. Nous proposons donc des pistes qui pourraient permettre d'obtenir de meilleurs résultats mais qui n'ont pas été testées.

Premièrement, il faudrait utiliser un algorithme permettant de repérer sur une image les points de départ des traces écrites. On pourrait alors faire partir les représentations internes de ces points de départ. Deuxièmement, au lieu de tenter de s'adapter à l'ensemble de la lettre, on pourrait le faire quart de cycle par quart de cycle (selon le modèle POMH), en cherchant à adapter les paramètres oscillatoires dans leur ordre d'apparition. Par ailleurs, la dynamique de l'écriture étant peu variable, on pourrait utiliser les temps des points d'annulation des vitesses associés à l'extraction de tangentes sur l'image à reconnaître. Cela permettrait de réduire grandement l'ensemble des séquences possibles. Bien sûr, de nouvelles erreurs seraient induites car les algorithmes d'extraction de points extrêmes et l'extraction de tangentes ne donnent pas toujours de bons résultats.

Chapitre 6

Conclusion

6.1 Contributions

Nous résumons ici les principales contributions de cette thèse tant au niveau théorique qu'au niveau technique.

Notre première contribution a été de réhabiliter le modèle oscillatoire de l'écriture, d'abord présenté par Hollerbach [1981], de le symétriser, de fournir une heuristique permettant de calculer les paramètres de celui-ci à partir de traces enregistrées, de l'appliquer à des lettres, des mots et des phrases entières. Il a aussi été appliqué avec succès à des traces écrites en langue étrangère, sur des phrases en arabe et sur des kanji chinois. A notre connaissance, c'est la première fois qu'un modèle oscillatoire de l'écriture est appliqué à des traces aussi complexes. Nous avons aussi comparé quantitativement notre modèle à celui d'Edelman et Flash [1987], qui est un des modèles usuels de la production de trace.

Notre seconde contribution a été, en s'appuyant sur les travaux de l'équipe du LAPMA à Toulouse [Athènes et al., 2004; Sallagoïty et al., 2004; Danna et al., 2010], d'étendre l'étude de la coordination par l'approche dynamique lors de la génération de trace, étudiée pour des ellipses, à l'écriture en général grâce à différentes méthodes de calcul de la phase relative continue s'appuyant sur une vision oscillatoire de l'écriture. Par ailleurs, les liens entre phase relative, vitesse d'écriture et formation de la trace ont commencé à être explorés.

Notre troisième et dernière contribution théorique a été de commencer à appliquer la vision générative de l'écriture à la reconnaissance de caractère. En particulier, on a cherché à reconstruite la dynamique d'une trace écrite en utilisant POMH. Ce travail repose sur la philosophie interactiviste développée par Bickhard [2009], reprise dans le domaine de l'intelligence artificielle par Buisson [2004], Buisson et Quinton [2010], Perotto [2010] et Quinton [2008].

En ce qui concerne nos contributions techniques, nous avons d'abord mis en place une suite logicielle d'étude de l'écriture, HollerTools, présentée en annexe A, permettant :

- une gestion facilitée de la saisie de données manuscrites dans le cadre d'une expérience,
- une édition des signaux qui composent l'écriture, permettant d'explorer et de comprendre comment celle-ci se met en place,
- l'application à la volée d'algorithmes d'analyse sur des traces enregistrées,
- une meilleure compréhension de la modélisation oscillatoire de l'écriture.

Nombre de nos algorithmes étant échafaudés avec MATLAB, et la traduction de ce dernier en Java (langage utilisé pour notre suite HollerTools) étant fastidieuse à la main, nous avons élaboré un traducteur automatique MATLAB vers Java, appelé MC, qui pourra être amélioré et mis à disposition de la communauté.

Enfin, nous avons développé une méthode, basée sur le crochetage d'API, permettant la réutilisation de jeux vidéos afin de bénéficier de leur moteurs physiques et graphiques souvent très sophistiqués (voir 6.2 et D). Cet environnement permet aux équipes voulant interagir avec des environnements virtuels, riches et réalistes, de le faire sans dépenser de ressources au préalable à la réalisation d'un simulateur.

6.2 Perspectives

Pour terminer, nous proposons ici des pistes, non encore évoquées, pour ceux qui voudraient poursuivre le travail entamé dans cette thèse.

Nous avons proposé un modèle oscillatoire de l'écriture, dont les paramètres sont constants par morceaux et changent à tour de rôle tous les quarts de cycles. Nous avons déjà suggéré quelques pistes de réflexion, mais d'autres, plus générales, viennent à l'esprit. Est-il possible de réduire le nombre de paramètres de ce modèle, ou leur fréquence de changement, tout en conservant une trace synthétisée lisible ? Dans le chapitre 4, nous avons pu calculer une phase relative en supposant la période de l'écriture constante. Peut-on se servir de ce résultat pour trouver un modèle qui utiliserait seulement comme paramètres une phase relative et des amplitudes ? Cette phase relative peut-elle être simplifiée et supposée constante par morceaux (à première vue ce n'est pas évident) ? Si oui, les points de changement de celle-ci correspondent-ils à des moments remarquables de la trace écrite ou du mouvement associé ? Y-a-t-il un lien temporel entre ces changements et les changements d'amplitudes ?

Un deuxième point qui n'a pas été abordé dans ce travail concerne le problème des levers de crayon. Où sont-ils situés dans le mouvement d'écriture ? Suivent-ils une rythmicité particulière ? Subissent-ils une contrainte temporelle ?

Nous avons cherché à mettre en avant l'existence d'une dynamique de coordination particulière lors de l'écriture manuscrite au chapitre 4. Il a été reproché à Athènes et al. [2004] de seulement supposer l'existence d'une dynamique de coordination sans en donner de modèle mathématique comme ont pu le faire Haken et al. [1985] pour la coordination bimanuelle ou Dubey, Sambaraju, Cautha, Arya, et Chakravarthy [2009] pour le sens de tracé d'un segment en fonction de son orientation. Ce même reproche pourrait nous être fait dans le travail que nous avons fourni.

Toutes ces considérations appliquées au mouvement d'écriture, mouvement plan, sont-elles applicables à d'autre mouvements ? Dans le plan ? Dans l'espace (en utilisant un troisième oscillateur) ? Certaines expérimentations non présentées ici ont semblé montrer que POMH était capable de reproduire des trajectoires ne correspondant pas à de l'écriture. En particulier il semble capable de reproduire les traces utilisées pour créer un dessin. Cela laisse supposer que la réponse à ces questions est affirmative.

Enfin, pour tenter d'ancrer ce modèle dans cadre neuro-musculaire, il pourrait être intéressant de reprendre les travaux de Gangadhar et al. [2007], présentés en section 2.1.2, pour montrer que la génération de trace basée sur deux oscillateurs du plan peut émerger d'interactions à l'intérieur d'un réseau de neurones oscillants.

Annexes

Annexe A

Outils logiciels de saisie et d'analyse de l'écriture

Cette première annexe a pour but de présenter les outils logiciels qui ont été élaborés, au cours de nos travaux, afin de faciliter l'étude de l'écriture manuscrite. Ces logiciels devant pouvoir être utilisés par des non-informaticiens, au nombre de 4, sont réunis sous le nom de HTools. Dans une première section, nous présenterons ces logiciels ainsi que leurs fonctionnalités. Dans une deuxième section nous reviendrons sur certaines de leurs particularités techniques.

A.1 Suite HTools

La suite logicielle HTools est composée de quatre programmes. Le premier, ExpDO, permet de spécifier, gérer et exécuter des expérimentations. Le second, HandWritingEditor permet d'éditer des traces manuscrites. HollerSynth est un petit programme permettant de s'approprier la vision oscillatoire de l'écriture au sens de Hollerbach [1981] (voir 2.2.1). Enfin, le dernier, HollerMap, permet d'appliquer des algorithmes à la volée immédiatement après la saisie d'une trace, et d'afficher les résultats renvoyés par ceux-ci sous forme interactive.

Le but ici n'est pas de donner une notice d'utilisation de ces logiciels. Il s'agit juste de présenter quelles sont les fonctionnalités de la suite HTools et de montrer leur utilité dans un travail de recherche sur l'écriture.

A.1.1 ExpDO

ExpDO permet d'automatiser le processus expérimental afin de faciliter le travail de l'expérimentateur. Les données saisies sont organisées de manière cohérente et systématique, permettant un usage automatique de celles-ci.

D'un point de vue matériel, ExpDO est optimisé pour fonctionner avec une tablette Wacom, munie d'un écran, branchée sur un ordinateur possédant lui aussi un écran. Il est toujours possible d'utiliser ExpDO sans tablette graphique, en utilisant la souris comme périphérique d'acquisition et l'écran de l'ordinateur pour toutes les visualisations, mais l'ergonomie d'utilisation sera beaucoup moins bonne et l'échantillonnage de la trace sera de beaucoup moins bonne qualité.

Dans la configuration idéale, le participant écrit sur la tablette Wacom. L'ensemble des informations dont il a besoin sont affichées sur la tablette, en particulier le retour visuel de la trace en temps réel lors de la saisie, et ce afin de s'approcher le plus possible d'une expérience naturelle de l'écriture. De même, il est possible d'afficher, sur cet écran, les

indications sur ce que doit faire le participant, ainsi que tout élément visuel jugé nécessaire par l'expérimentateur. Ce dernier, utilise l'autre écran, où une fenêtre de contrôle lui permet de suivre le bon déroulement de l'expérience et éventuellement de l'interrompre ou le modifier.

Pour mieux comprendre les fonctionnalités de ExpDO, prenons un exemple. Nous souhaitons faire une expérience (E) où le participant trace un rond (C1 : condition 1), un carré (C2) et un triangle (C3). Pour chaque condition, le participant devra faire 20 échantillons. E peut être dirigée par ExpDo pourvu qu'elle lui soit spécifiée. Cette spécification se fait au travers d'un fichier XML dont l'expérimentateur doit vérifier la cohérence. Lors de l'expérimentation, l'expérimentateur doit s'assurer que le participant respecte les conditions : il est possible de refaire n'importe quelle saisie déjà effectuée au cours d'une expérience.

Les saisies sont enregistrées au fur et a mesure dans une arborescence du type *individu/condition/numero_ echantillon.ex*. Les fichiers *ex* contiennent les suites de points datés de la trajectoire tracée par le participant pour chaque échantillon. Il est par ailleurs possible d'enregistrer au même moment une image de la trace écrite telle que représentée sur l'écran de saisie.

L'accès aux données de bas niveau de la tablette Wacom, comme le fait ExpDO permet d'obtenir des données plus fiables sur la trace saisie (voir section A.2.1). Par ailleurs la gestion automatique des expérimentations permet un gain de temps, et une meilleure organisation de celles-ci.

A.1.2 HandwritingEditor

HandwritingEditor permet d'éditer graphiquement des fichier *ex*, en agissant sur chaque composante de la trace enregistrée. Il est ainsi possible de modifier "à la main" les profils de position et de vitesse d'une trace donnée. Il n'est cependant pas possible de modifier directement la géométrie de la trace.

Les profils éditables sont donc des profils temporels. Il est possible d'en sélectionner tout ou partie afin d'appliquer, sur cette sélection, des opérateurs. Ceux-ci, comprennent entre autres l'opérateur d'homothétie et l'opérateur de translation. L'opérateur homothétique permet d'agrandir une partie du signal (verticalement et/ou horizontalement). En cas d'agrandissement, les valeurs du signal occultées par celui-ci sont supprimées, en cas de rétrécissement le signal est interpolé là où aucune valeur n'est disponible. L'opérateur de translation (vertical et/ou horizontal) fonctionne selon le même principe. On notera aussi la présence d'un opérateur de lissage de la sélection.

Deux autres opérateurs ne nécessitent pas de sélection : l'opérateur de juxtaposition qui permet de juxtaposer une autre trace à la trace en cours d'édition et l'opérateur de dessin à main levée qui permet de dessiner à la souris la forme que l'on veut donner au signal édité.

A.1.3 HollerSynth

HollerSynth permet aux personnes intéressées par les modèles oscillatoires de l'écriture de s'approprier le modèle génératif d'Hollerbach. Il permet de spécifier les six paramètres oscillatoires (amplitudes, fréquences et phases) utilisés par le modèle sur chaque demi-période (entre deux points d'annulation de la vitesse verticale). Afin de mieux comprendre le lien entre les paramètres fournis au modèle et la forme des lettres générées, le diagramme d'Abelson (diagramme des phases), tel que présenté en figure 2.9, peut par ailleurs être visualisé.

A.1.4 HollerMap

Lorsque l'on veut tester différents algorithmes sur un ensemble d'échantillons et que l'on veut avoir un retour visuel instantané du résultat, il peut être avantageux d'avoir une interface graphique permettant de gérer ces besoins.

HollerMap permet de lire des échantillons depuis des fichiers *ex*, tels que générés par ExpDO, ou autorise leur saisie directement via la tablette. A chaque nouvel échantillon importé, une liste d'algorithmes, éditable, lui est appliquée. Il est alors possible de naviguer parmi les différents échantillons et les différents algorithmes pour visionner et interagir avec les résultats de chacun.

Il est bien sûr possible d'ajouter des algorithmes afin d'accroître les usages possibles. Les algorithmes disponibles de base sont l'algorithme FHA et un algorithme qui permet d'afficher la vitesse de la trace en fonction de sa courbure élevée à une puissance donnée (utilisé pour étudier la loi de puissance $\frac{2}{3}$ [Viviani et Terzuolo, 1982; Lacquaniti, Terzuolo, et Viviani, 1983]).

A.2 Détails techniques et implémentation

L'ensemble des logiciels de HTools sont écrits en Java, les interfaces sont basés sur Swing. Ils sont compatibles Linux et Windows et s'exécutent de manière fluide sur du matériel modeste (Intel Cored 2 Duo, 2Go DDR2, Nvidia GT 240).

Deux aspects intéressants de l'élaboration de ces logiciels méritent d'être présentés ici. Le premier concerne la récupération des échantillons des traces écrites, le second concerne l'utilisation d'algorithmes élaborés en MATLAB dans ces applications Java.

A.2.1 Échantillonnage de la trace écrite

La figure A.1 montre l'organisation de la gestion des tablettes graphiques sous Linux comme sous Windows. Le système de gestion des pointeurs (souris, tablettes ...) du système opératoire demande au driver Wacom si des données sont présentes et, si c'est le cas, ce dernier rapatrie les données de la tablette et les lui fournit. Le système opératoire utilise alors ces données pour déclencher des événements de pointeur (déplacements, clics ...). Ces événements peuvent être ensuite utilisés par HTools grâce à Swing qui s'appuie sur ceux-ci.

Cette approche pose deux problèmes. Le premier, c'est que les événements système vont aussi vers les autres applications du système (en particulier vers le bureau), qui réagissent au mouvement du stylet sur la tablette, ce qui peut créer des interférences lors de la saisie d'une trace écrite. Deuxièmement, les gestionnaires d'événement n'envoient pas toutes les données de la tablette vers l'application. De plus les données envoyées le sont de manière irrégulière. Ainsi on peut avoir 30 ms entre deux événements puis 150ms entre les deux suivants. Ces deux problèmes font que la saisie devient perturbée et que les données résultantes sont incomplètes et peu utilisables.

Pour corriger ces problèmes (voir figure A.2), nous avons choisi d'aller directement demander les données au driver Wacom et de désactiver le lien entre la tablette et la gestion d'événements pointeur du système. Nous pouvons ainsi accéder à toutes les données émises par la tablette. De nouvelles données sont disponibles toutes les 10ms. En revanche, comme désiré, les autres applications ne reçoivent plus d'événements système liés à la tablette.

Cette approche fonctionne très bien mais comporte un inconvénient majeur : pour utiliser le driver Wacom du code natif doit être utilisé. Ainsi, ils nous a fallu faire un interface JNI (permettant d'appeler du code natif en Java). Cet interface a dû être implémenté pour chaque système opératoire sur lequel on souhaite que notre application puisse s'exécuter

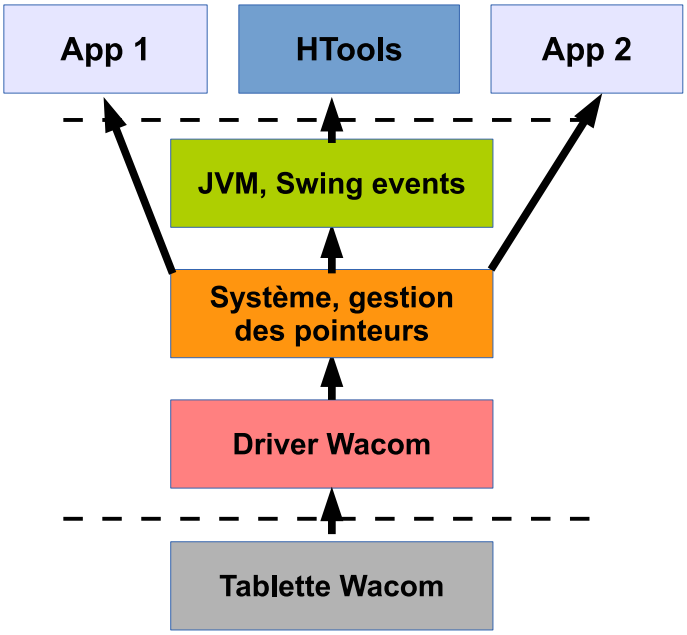


Figure A.1 – Gestion de la tablette sous Linux et Windows. Le système accède aux données du driver puis les utilise dans son système de gestion d'évènement auquel on peut accéder via une libraire de gestion d'interface (Java Swing dans notre cas).

.

(car les drivers Wacom n'ont pas la même API d'un système à l'autre). Nous avons donc dû faire deux implémentations : une pour Linux et une pour Windows.

A.2.2 Utilisation de code MATLAB

L'ensemble des algorithmes liés aux modèles présentés dans cette thèse (chapitres 3 et 4), ont été élaborés avec MATLAB. Pour les intégrer dans la suite HTools, il a fallu les rendre utilisables par Java. Si les premiers ont été traduits à la main, nous avons cherché d'autres méthodes nous évitant cette tâche. Trois options alternatives ont été testées et sont présentées ici plus ou moins en détails en fonction de leur utilité.

A.2.2.1 MATLAB Builder JA

Il s'agit de la méthode officiellement supportée par MATLAB. Le MATLAB Builder JA, accessible depuis MATLAB, fournit une interface utilisateur permettant de choisir les fonctions MATLAB qu'on souhaite exporter. Un fichier jar (librairie Java) est alors généré, qui, une fois inclus dans l'application Java permet d'exécuter ces fonctions. Cependant, il est nécessaire lors du déploiement de cette application d'installer le MCR (MATLAB Compiler Runtime) sur les machines hôtes. Le MCR peut être vu comme une machine virtuelle MATLAB qui permet d'exécuter du code précompilé.

Cette approche a pour avantages de faciliter le déploiement et l'utilisation de fonctions MATLAB depuis une application Java. Deux inconvénients sont néanmoins particulièrement présents. Le premier est la nécessité de régénérer le fichier jar à chaque édition des fichiers MATLAB, ce qui prend beaucoup de temps. Le deuxième est le fait de devoir installer une machine virtuelle MATLAB chez chaque hôte : notre code dépend alors de celle-ci. Par ailleurs, ce MCR et assez imposant en terme de taille.

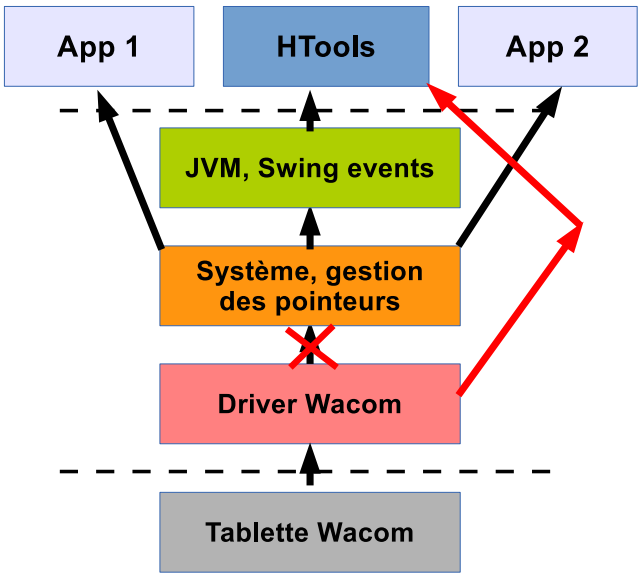


FIGURE A.2 – Pour contrer les problèmes dûs à la gestion de la tablette par le système opératoire (figure A.1), le lien entre le driver et le système de gestion de pointeur de l'OS est désactivé et HTools vient chercher les données directement depuis le drivers

.

A.2.2.2 matlab control

`matlab control` est une librairie pour Java initialement écrite par un enseignant pour automatiser le processus de correction de travaux pratiques MATLAB. Cette librairie permet d'invoquer des fonctions MATLAB depuis Java en utilisant MATLAB lui même pour les exécuter et en utilisant l'API Java de MATLAB pour dialoguer avec ce dernier.

Le premier avantage de cette méthode est de ne pas nécessiter de générateur automatique : il est directement interprété par MATLAB. Le code MATLAB peut même être modifié pendant l'exécution de l'application Java, ce qui en fait un bon outil pour le débugage. L'inconvénient majeur est que MATLAB doit être installé sur toute machine sur laquelle l'application est déployée.

A.2.3 MC : le traducteur MATLAB vers Java

Cette dernière méthode, cherche à résoudre l'un des inconvénients présent dans les deux méthodes précédentes : MATLAB (ou une partie) doit être installé sur toutes les machines devant faire tourner l'application Java faisant appel a du code MATLAB.

Pour contourner ce problème, un traducteur MATLAB vers Java à été écrit. Cela a impliqué deux travaux : écrire un parseur MATLAB qui génère du code Java, et, écrire une librairie Java de calcul matriciel supportant les opérations intrinsèques de MATLAB. Ces deux travaux ont été effectués et sont présentés plus en détails dans l'annexe C. Ici on retrouve la contrainte de compilation à chaque modification des fichiers MATLAB utilisés, mais cette phase de compilation est très rapide (beaucoup plus que dans le cas de MATLAB Builder JA). En revanche, le code source Java généré peut être modifié manuellement si nécessaire.

Annexe B

Oscillateurs, couplage et synchronisation

Nous présentons ici un travail qui a été réalisé en début de notre travail de thèse. L'idée était d'explorer le phénomène de synchronisation sur un ensemble d'oscillateurs couplés afin d'intégrer celui-ci dans une modèle de l'intelligence interactiviste basé sur les principes présentés au chapitre 5. Un tel modèle serait basé sur des processus anticipatifs, dont différentes modélisations ont déjà été réalisées [Drescher, 1991; Buisson, 2004; Quinton, 2008; Perotto, 2010]. Le problème majeur de la plupart de ces modèles est leur absence de prise en compte du temps d'une façon satisfaisante. Ce dernier problème semble pouvoir être résolu par la synchronisation spontanée des processus entre eux et avec le réel, à la manière des populations d'oscillateurs couplés. C'est l'étude préalable de ces derniers qui est présentée ici. Nous avons finalement restreint nos travaux aux processus oscillatoires de l'écriture mais nous pensons néanmoins qu'il est intéressant de présenter ce qui a été fait ici.

En abordant rapidement les systèmes dynamiques au chapitre 4 nous avons dit qu'un système complexe, composé de multiples entités en interaction, pouvait donner lieu à l'émergence de comportements globaux. Le phénomène que nous étudions ici, à savoir la synchronisation éventuelle de populations d'oscillateurs couplés en est un exemple parfait. Plusieurs exemples de ce phénomène sont présents dans la nature, tant au niveau du vivant, du macroscopique ou du microscopique. Il peut s'agir de colonies de lucioles qui la nuit clignotent ensembles, d'oscillations électriques cérébrales qui vibrent plus ou moins en phase, du cycle circadien qui se synchronise à l'alternance du jour et de la nuit, des cycles menstruels chez des femmes vivant en groupe, etc. [Strogatz, 2004; Buzsáki, 2006]. Cette synchronisation est d'autant plus remarquable qu'elle n'est pas dirigée, elle émerge spontanément au sein d'un groupe d'éléments oscillants en interaction.

La première personne ayant rapporté ce phénomène de synchronisation spontanée entre plusieurs systèmes oscillants est Christian Huygens [Korteweg, 1906] qui, en 1665, regardant deux horloges à balancier (inventées par lui même) accrochées au mur de sa chambre, remarqua que celles-ci retrouvaient spontanément un état de synchronisation. Il réussit finalement à les désynchroniser en les séparant (ie. en les accrochant dans des pièces différentes). En réalité, les deux horloges étaient mécaniquement couplées par les vibrations se propageant au travers des murs de la chambre.

De nombreux travaux, parmi lesquels [Arenas, Kurths, et Moreno, 2008; Cuomo, 1993; Jadbabaie, Motee, et Barahona, 2004; Rosenblum et Pikovsky, 2004; Strogatz, 2000, 2001; Tsang, Mirollo, Strogatz, et Wiesenfeld, 1991; Olfati-Saber, 2005], ont étudié ce phénomène de synchronisation aussi bien mathématiquement que par des simulations informatiques

pour tenter de répondre aux questions suivantes : quels facteurs influent sur la synchronisation d'une population d'oscillateurs couplés ? Quelle est l'influence de la taille de la population ? De la latence et de la nature des signaux de couplage émis ? De la topologie du réseau de couplage ? De la forme des oscillateurs ?

Nous avons exploré certaines de ces questions grâce à des simulations. En particulier nous abordons les problèmes de délais et de topologie du réseau. L'une de nos simulations à été implémentée directement dans un FPGA. Hélas, le manque de place dans celui-ci nous a empêché d'obtenir des résultats significatifs.

Dans la suite de cette annexe, nous commençons par donner des éléments théoriques permettant de modéliser une population d'oscillateurs couplés et leur adaptation à la simulation informatique. Nous présentons ensuite les résultats de nos simulations effectuées sur des populations d'oscillateurs sous certaines conditions. Finalement, nous présentons nos expérimentations réalisées avec le FPGA.

B.1 Éléments théoriques et simulation

B.1.1 Considérations mathématiques

On définit $\mathcal{O}$ un oscillateur comme un couple (x, f) où f décrit intrinsèquement l'évolution de la variable de x au cours du temps :

$$\frac{dx}{dt}(t) = f(t, x). \tag{B.1}$$

On définit une population $\mathcal{P}$ comme un ensemble d'oscillateurs :

$$\mathcal{P} = \{\mathcal{O}_i, 1 \leqslant i \leqslant N, N \in \mathbb{N}^*\} \tag{B.2}$$

$\mathcal{P}$ est dite homogène si

$$\forall \mathcal{O} \in \mathcal{P}, \forall O \in \mathcal{P}, O.f = \mathcal{O}.f. \tag{B.3}$$

Pour une population $\mathcal{P}$ donnée on peut alors définir un oscillateur couplé $\mathscr{O}$ comme un quintuplet $(x, f, \mathcal{P}, \mathcal{E}, \mathcal{S})$ où $\mathcal{E}$ et $\mathcal{S}$ décrivent les interactions entrantes et sortantes avec les autres oscillateurs de $\mathcal{P}$ de sorte que

$$\frac{dx}{dt}(t) = f(t, x) + \mathcal{E}(t, x) \tag{B.4}$$

et le message M émit à l'instant t par l'oscillateur est donné par :

$$M = \mathcal{S}(t) \tag{B.5}$$

Bien entendu, $\mathcal{S}$ peut dépendre de f tout comme $\mathcal{E}$. $\mathcal{E}$ dépend aussi des messages émis par les autres oscillateurs. Une population d'oscillateurs couplés est dite dense si :

$$\forall \mathcal{O} \in \mathcal{P}, \forall O \in \mathcal{P}, \mathcal{O} \neq O \Leftrightarrow O.\mathcal{E} \not\perp \mathcal{O}.\mathcal{S} \tag{B.6}$$

B.1.2 Cas des pendules

L'expérience des pendules de Huygens a servi de modèle de base pour une expérience réalisée par Jean-Christophe Buisson qui a élaboré une simulation basée sur des pendules. Nous allons ici présenter cette expérience en décrivant la simulation associée.

Un pendule est ici décrit par l'oscillateur $\mathcal{P}_i = (x_i, f_i, \mathfrak{P}, \mathcal{E}_i, \mathcal{S}_i)$ où $x \in \mathbb{R}^+$ et $\mathfrak{P}$ la population d'oscillateurs constituée de N pendules. f_i décrivant l'évolution intrinsèque

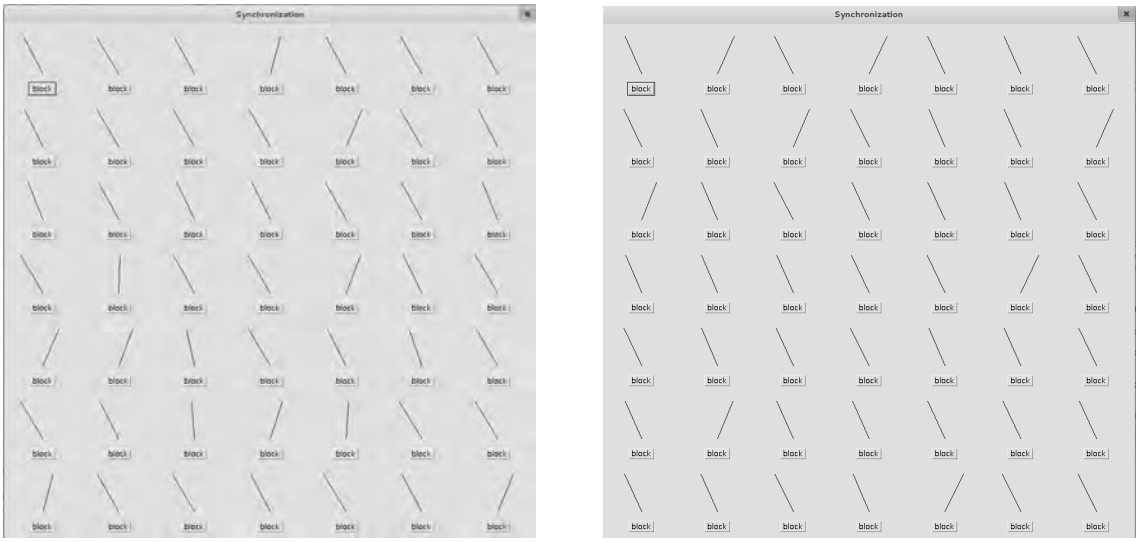


FIGURE B.1 – Captures d'écran du programme de simulation original de synchronisation des pendules. A l'état initial, les pendules oscillent à différente phases (à gauche), elles finissent par se synchroniser pour que seules deux phases coexistent (à droite)

de x, qui est ici la phase de l'oscillation du pendule au cours du temps, est définie par $f : x \mapsto \tau$ ou $\tau > 0$, Autrement dit, f fait osciller le pendule de manière constante. $\mathcal{E}_i$ peut s'exprimer par :

$$\mathcal{E}_i = \sum_{\substack{k=1 \\ k \neq i}}^{k=N} e_{i,k} \tag{B.7}$$

où les $e_{i,k}$ sont définis par

$$e_{i,k} = \begin{cases} -\tau \text{ si } t \in \bigcup_l [t_l + \Delta t_k, t_l + \Delta t_k + \cos(x(t_l + \Delta t_k)] \\ 0 \text{ sinon} \end{cases} \tag{B.8}$$

où t_l sont les temps pour lesquels $\mathcal{S}_k$ n'est pas nul. De manière plus intuitive, chaque message qui arrive à l'oscillateur a pour effet de bloquer celui-ci pour un temps $\cos(x)$, c'est à dire que plus le pendule, dont la phase est représentée par x et le mouvement par $\sin(x)$, sera loin d'une extrémité quand il recevra ce message, plus il sera affecté. Enfin

$$\mathcal{S}_i = \mathbb{1}_{\{t, |\sin(x(t))|=1\}} \tag{B.9}$$

Une telle population d'oscillateurs couplés dans le cas où $\forall k, \Delta t_k = 0$ atteint un état de synchronisation mixte dans lequel les pendules oscillent en phase ou en antiphase. La figure B.1 montre une capture d'écran du programme de simulation original.

B.1.3 Cas des accumulateurs

Les oscillateurs basés sur des accumulateurs sont très présents en biologie : on les retrouve par exemple sous la forme de cellules cardiaques chargées de réguler les contractions du cœur ou sous la forme de neurones qui "déchargent" vers d'autres les charges électriques accumulées.

Notons la population d'accumulateurs $\mathfrak{A}$ composées de N oscillateurs $(\mathcal{A}_i)_{0<i\leqslant N}$. Chaque $\mathcal{A}_i$ est défini par $\mathcal{A}_i = (x_i, f_i, \mathfrak{A}, \mathcal{E}_i, \mathcal{S}_i)$ où x varie dans $[0, 1]$ de manière monotone et croissante. Pour un ϵ donné lorsque $x > 1 - \epsilon$ alors x retombe à 0.

f_i décrivant l'évolution intrinsèque de x au cours du temps est définie par $f : x \mapsto (1-x)/\tau$ ou $\tau > 1$. $\mathcal{E}_i$ est définie par :

$$\forall t \in \mathbb{R}, \mathcal{E}_i(t) = \sum_{\substack{k=1 \\ k \neq i}}^{k=N} \omega_k \mathcal{S}_k(t - \Delta t_k) \tag{B.10}$$

où les ω_k déterminent l'influence de chaque oscillateur de $\mathfrak{A}$ sur $\mathcal{A}_i$ et les Δt_k représentant le délai de propagation de chaque signal de sortie $\mathcal{S}_k$ vers $\mathcal{A}_i$. Enfin

$$\mathcal{S}_i = \begin{cases} 1 \text{ si } x = 1 - \epsilon \\ 0 \text{ sinon} \end{cases} \tag{B.11}$$

Mirollo et Strogatz [1990] ont étudié mathématiquement de telles populations d'oscillateurs couplés. Les preuves de synchronisation de telles populations, présentées dans ces travaux, font l'hypothèse que l'intégrale de f est concave, ce qui est le cas ici.

B.1.4 Simulation du cas des accumulateurs

L'implémentation de la simulation repose sur le paradigme de programmation objet. Une classe `CondoSim` a pour but de gérer la population d'oscillateurs ainsi que de déclencher chaque nouveau pas de la simulation. Chaque oscillateur est représenté par une classe `Condo`, gérant la variable collective x qui représente l'état de l'oscillateur.

A chaque pas de la simulation, la classe `CondoSim` appelle la méthode `next()` de tous les oscillateurs. Cette méthode, associée à chaque oscillateur, commence par vérifier si $x > 1 - \epsilon$. Si tel est le cas, alors x est mise à zéro et un signal est envoyé à tous les oscillateurs couplés avec celui-ci pour qu'ils incrémentent brutalement leur phase. Ensuite x est mis à jour avec la formule : $x = x + (1-x)\tau$ (on suppose dans notre simulation $dt = 1$). Afin d'éviter les appels récursifs de fonction, la variable x n'est pas testée lorsqu'elle reçoit un message de la part d'un autre oscillateur (lors d'un incrément brutal). Le test de $1 - \epsilon$ ne se fera alors qu'au prochain appel de la fonction `next()`.

La valeur x de chacun des oscillateurs est présentée à l'écran par un carré allant du vert ($x = 0$) au blanc ($x \approx 1$). Les oscillateurs étant présentés sous forme de grille, s'ils sont tous synchronisés alors à chaque instant la couleur de toute la population sera identique, sinon, l'on discernera les différents oscillateurs. La figure B.2 montre un exemple d'exécution de la simulation pour une population de 256 oscillateurs couplés densément. Les trois images successives montrent trois états de la population dans un ordre chronologique. Sur la dernière image, la population est synchronisée.

B.2 Expériences autour de la synchronisation de populations d'oscillateurs

Nous nous attachons donc ici à tenter d'inspecter, grâce à une simulation, le comportement d'un groupe d'oscillateurs couplés sous plusieurs conditions : population couplée densément et avec liaisons sans délai, population couplée densément avec liaisons à délai et population couplée en topologie petits-mondes avec liaisons sans délai.

Les populations étudiées dans cette partie sont constituées de 256 oscillateurs tels que présentés dans la section B.1.3. On utilise $\tau = 32$ et on suppose que les ω_k sont uniformes pour toutes les liaisons. Le nombre d'oscillateurs a une influence sur la vitesse de convergence. Le choix du nombre 256 a été déterminé empiriquement.

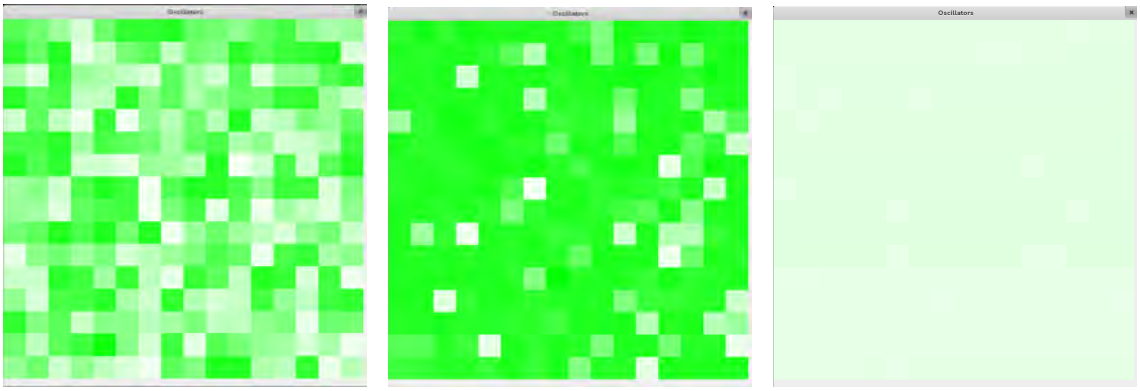


FIGURE B.2 – Exemple de visualisation de la synchronisation d'une population de 256 oscillateurs densément couplés.

Afin d'estimer la vitesse et le degré de synchronisation de la population nous avons mis un place un indicateur que nous avons nommé l'*asynchronicité*. Pour une population $\mathcal{P}$ de taille N on définit son asynchronicité $\mathfrak{S}$ par :

$$\mathfrak{S} = \frac{1}{N} \sum_{i=1}^{i=N-1} \mathbb{1}_{\|x_{\sigma(i+1)} - x_{\sigma(i)}\| > \tau}, \tag{B.12}$$

où σ est une permutation sur $[\![1, N]\!]$ telle que $\forall i \in [\![1, N]\!], x_{\sigma(i+1)} \geqslant x_{\sigma(i)}$ et τ un seuil de tolérance sur la synchronie de deux oscillateurs. Dans notre simulation nous utilisons 0.05 comme valeur de τ. Plus le taux d'asynchronicité est élevé et plus le degré de synchronisation de la population est faible.

B.2.1 Population densément couplée, liaisons sans délai

Pour ce cas le plus simple, on cherche à découvrir l'influence de ω_k sur la vitesse de synchronisation. La figure B.3 montre le résultat de la simulation lancée pour des ω_k de trois ordres de grandeur différents 0.001, 0.0001,et 0.00001. Sur une échelle logarithmique, on a représenté l'asynchronicité en fonction du nombre de cycles de la simulation (on va jusqu'à 100 000). On remarque que dans le premier cas il y a une convergence très rapide en quelques centaines d'itérations, dans le second cas il y a une convergence plus lente en quelques milliers d'itérations, enfin, dans le dernier cas il n'y a pas convergence.

Il semble donc, comme on aurait pu le prédire intuitivement, que le poids du message reçu par chacun des oscillateurs a un impact positif sur la convergence. On peut aussi remarquer que l'asynchronicité oscille jusqu'à la convergence. Dans le cas où il n'y a pas convergence, cette mesure oscille régulièrement et continûment. Par ailleurs même s'il y convergence on peut remarquer une petite oscillation résiduelle entre 0 et 0.05 dans le cas où $\omega_k = 0.0001$. Est-ce un effet de la simulation où une observation généralisable ? Cela vient-il de notre choix de la mesure du degré de synchronisation ? Nous n'avons pas la réponse. Dans les expériences suivantes, nous utiliserons pour ω_k la valeur 0.001 qui semble favorable à une synchronisation rapide de la population.

B.2.2 Population densément couplée, liaisons avec délai

On suppose maintenant que les délais de couplage sont constants (et non-nuls), c'est à dire qu'une impulsion allant d'un oscillateur à un autre, dans une population, aura un temps constant de propagation quels que soient les oscillateurs impliqués. Afin de mieux

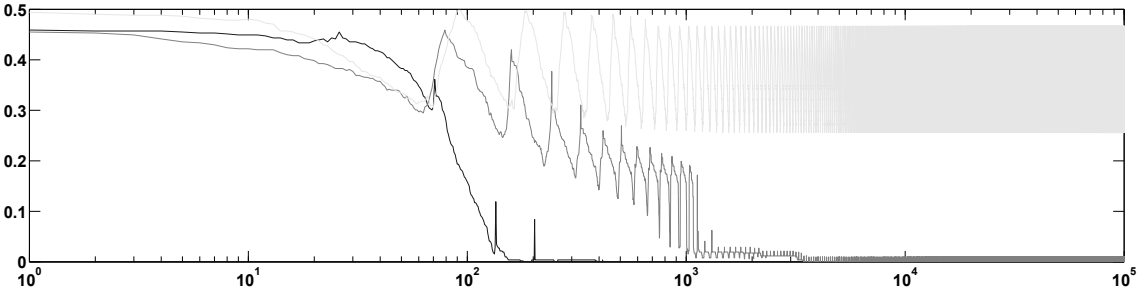


Figure B.3 – Comparaison de la convergence de la population vers un état de synchronie pour ω_k valant 0.001 (en noir),0.0001 (en gris foncé) et 0.00001 (en gris clair).

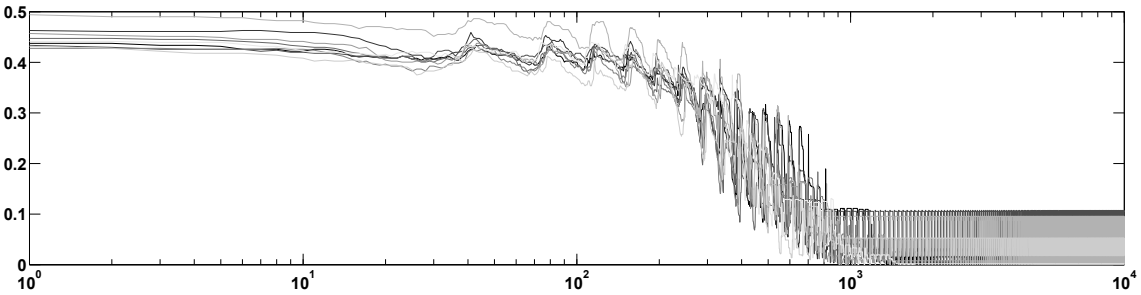


Figure B.4 – Visualisation de l'évolution de l'asynchronocité pour 10 populations de 256 individu densément couplés avec des liaisons à délai fixe, avec des états initaux différents.

comprendre l'impact de l'ajout de ce délai sur la synchronisation d'une population nous avons effectué deux simulations.

Premièrement nous avons cherché à comprendre l'impact du délai sur la synchronisation de populations (toujours de 256 individus) mais dont l'état initial est différent. La figure B.4 montre l'évolution des 10 populations soumises aux même conditions de couplage. Jusqu'au millième cycle, elles montrent un comportement semblable, avec une forte diminution de l'asynchronocité, diminution qui se fait par oscillations. Ensuite, les populations arrivent dans un état oscillant stable où l'asynchronicité oscille entre 0 et une valeur plus élevée de celle-ci. L'amplitude de ces oscillations est forte pour certaines des populations (jusqu'à 0.1) tandis que pour d'autres elles sont très faibles (moins de 0.05 où l'on peut parler de synchronisation).

Nous regardons maintenant l'évolution de l'asynchronicité en fonction de plusieurs délais (allant de 0 à 90), sur la même population, initialisée de la même façon. La figure B.5 présente les profils de synchronisation de cette population pour 9 délais différents (allant de 0 à 90ms, du noir au gris clair). La synchronicité n'est pas améliorée pour les cas où le délai est supérieur à 50ms. Dans les autres cas, il y a une augmentation de la synchronicité vers un état stable oscillant comme déjà vu dans le cas précédent. Encore une fois l'amplitude des oscillations varie, de 0 à 0.1. Le problème est que l'amplitude de ces oscillations ne semble pas directement lié à la magnitude du délai. Difficile encore une fois de savoir si ces oscillations stables, proches d'un état de convergence, sont liées à la simulation ou au modèle.

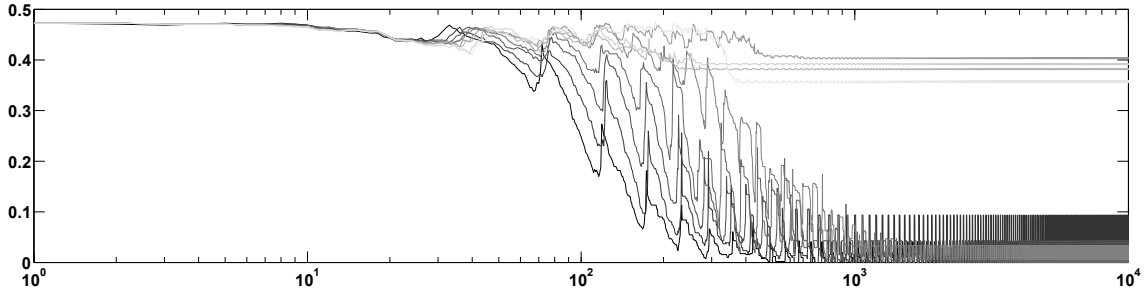


FIGURE B.5 – Influence du délai sur la synchronisation d'une population d'oscillateurs couplés. Les courbes plus foncées représentent les simulations dans lesquelles le délai des messages entre les oscillateurs sont les plus courts.

B.2.3 Population couplée avec une topologie en petits-mondes, liaisons sans délai

Le modèle de la population dense, permet d'atteindre la synchronisation de manière certaine si le délai de propagation du signal est suffisamment court. Dans une population dense, tout oscillateur envoie un message vers tous les autres, ce qui pour une population de N fait $N(N-1)$ connexions. Une population de taille $N = 256$ a donc 65 280 connexions. Peut-on réduire ce nombre ?

Regardons ce qu'il se passe si nous connectons chaque oscillateur de la population à ses quatre plus proches voisins. La courbe rouge figure B.6 montre l'évolution de l'asynchronicité pour une population reliée de cette façon. Si il y a diminution de celle-ci, correspondant aux synchronisations dûes aux influences locales, une synchronisation globale n'est pas possible.

Pour contrevenir à ce problème, il est possible de créer des connexions de manière aléatoire. On définit une mesure de densité allant de 0 à 1 exprimée comme le nombre de connexions aléatoires divisé par $N(N-1)$. Sur la figure B.6, la courbe claire représente le profil d'asynchronicité pour une densité de connexions aléatoire de 0.1, et la courbe foncée est associée à une densité de 0.001. Dans le premier cas, la convergence est meilleure mais pas parfaite alors qu'elle l'est dans le second cas.

Si l'on fait la moyenne de l'asynchronocité entre la 2000^{e} et la dernière itération sur les profils de vitesse correspondant à plusieurs valeurs de densité de connexions aléatoires (voir figure B.7, on observe que l'asynchronicité moyenne atteinte diminue très vite avec la densité. Dès 0.01 la population se synchronise presque parfaitement (asynchronocité $<$ 0.02). Dans ce cas on passe de 65 280 liaisons à 1677 soit une économie de plus de 97% de liaisons pour un résultat similaire. On remarque par ailleurs, pour une densité avoisinant 0.07, une petite augmentation de l'asynchronicité : est-ce dû à la simulation ?

Cette topologie, où chaque nœud est relié à ses voisins sous forme de cluster et où chaque cluster est relié par des liaisons longue distance de manière aléatoire est appelée topologie en petits-mondes. On retrouve cette topologie dans de nombreux systèmes réels. On peut citer comme exemple les liens entre les sites internet, les relations sociales ou le système nerveux. Si l'on considère qu'établir une liaison est coûteux, ce type de topologie offre le meilleur compromis entre accessibilité (nombre de liaisons empruntés pour atteindre n'importe quel nœud du système) et coût. Par ailleurs ces rares liaisons longue distante sont parfois plus performantes afin de faire en sorte que les latences soient du même ordre que celle observées pour les liaisons courtes. Par exemple dans le cerveau, les plus long axones ont la plus grosse section.

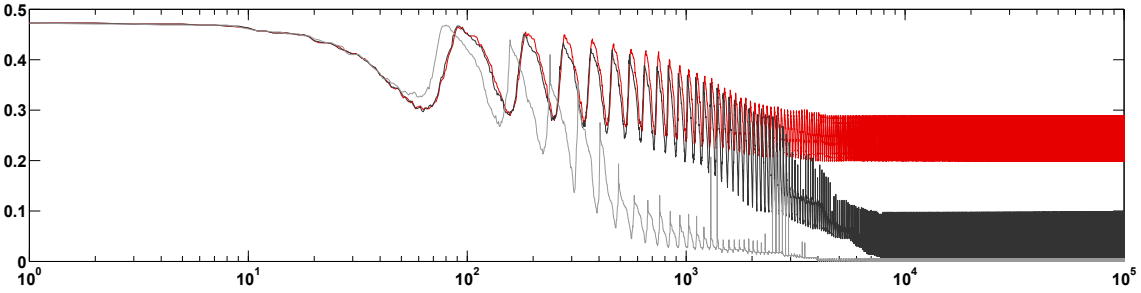


Figure B.6 – Convergence de population couplée en topologie petits-mondes. Les trois courbes sont associées à trois valeurs de densité différentes. Voir le texte pour plus de détails.

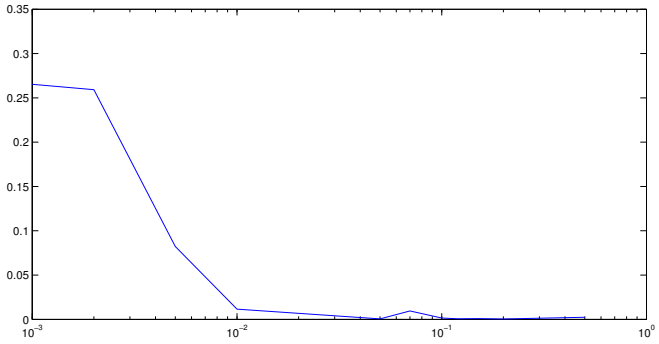


Figure B.7 – Asyncronicité moyenne atteinte par la population entre le 2000^e et le dernier cycle de la population en fonction de la densité des connexions aléatoires.

B.3 Grille d'oscillateurs couplés implémentée dans un FPGA

Pour aller plus loin dans notre exploration de la synchronisation de groupes d'oscillateurs, nous avons configuré un FPGA afin qu'il simule une population d'oscillateurs reliés par une topologie en petits-mondes.

B.3.1 Outils et matériel

Un FPGA (pour Field-Programmable Gate Array) est un circuit intégré dont le comportement peut être modifié par l'utilisateur, contrairement à un circuit intégré usuel ayant un comportement fixé. Un FPGA est constitué d'une matrice de blocs logiques, chaque bloc pouvant être configuré pour effectuer une fonction combinatoire complexe. Par ailleurs chaque bloc contient un ou plusieurs éléments de mémoire. Les blocs sont reliés entre eux au travers d'une grille de connexions configurables.

En pratique, le comportement que l'on veut donner au circuit peut être décrit à l'aide d'une description dans un langage dédié. Il en existe plusieurs tels que le VHDL ou encore le Verilog. Cette description est interprétée par les outils fournis par le fabricant de FPGA qui permettent la configuration automatique de ce dernier. A des fins de pédagogie dans le cadre d'un module d'enseignement, Jean-Christophe Buisson a mis en place, grâce aux travaux précédents de Jean Conter, une suite logicielle permettant de programmer et de monitorer le comportement d'un FPGA. Au cœur de ces outils est le langage SHDL qui permet de décrire tout circuit logique de manière structurelle en utilisant signaux et

opérations combinatoires et séquentielles. Notre simulation de population d'oscillateurs a été écrite avec ce langage. L'un des concepts clé de ce langage est la notion de module qui permet de hiérarchiser la description. Un module peut alors être vu comme une boite noire ayant des signaux d'entrées et de sorties.

Le FPGA utilisé pour la simulation a été le Spartran 3 de Xillinx, monté sur la carte Nexys 2 fabriquée par Digilent. Pour plus de détails techniques sur cette carte et sur ce FPGA le lecteur est invité à ce rendre sur la page de spécifications disponible sur le site du constructeur [1].

B.3.2 Implémentation

Ici un oscillateur est toujours vu comme un accumulateur et fonctionne sur le même principe que dans la simulation précédente. La population est couplée selon les principes de la topologie en petits-mondes présentée précédemment. Plus exactement, pour chaque oscillateur, cinq connexions sont prévues. Quatre sont établies vers les quatre voisins, et la dernière, pouvant être active où non, pointe vers un oscillateur choisi aléatoirement. La densité est alors définie ici comme le taux d'utilisation de cette cinquième connexion.

Dans notre implémentation, un oscillateur est décrit comme suit :

```
module oscillateur(clk,msg_rcv[4..0],ini,inival[15..0]:msg_snd)
```

où `clk` sera le signal d'horloge permettant à la simulation d'avancer à l'étape suivante, `msg_rcv` correspond aux cinq signaux reçus de la part d'autres oscillateurs. `ini` et `inival` permettent, au début de la simulation, de forcer les oscillateurs dans un état donné choisi aléatoirement. `msg_snd` est le signal émis par l'oscillateur lorsqu'il produit une décharge avant de retourner à l'état bas.

Le module principal de notre simulation est le module représentant la population qui connecte entre eux les oscillateurs et leur attribue une valeur initiale. Ce module est généré par un script Python afin de pouvoir créer des populations de n'importe quelle taille et avec des états initiaux différents. La densité des liens longue distance de la topologie petits-mondes est par ailleurs paramétrable.

Afin d'avoir un retour visuel de l'état de synchronisation de la population, les signaux de sortie, `msg_snd`, de certains oscillateurs sont reliées aux diodes disponibles sur la carte FPGA utilisée. Afin de pouvoir repérer visuellement le clignotement des diodes, le fonctionnement de la simulation est ralentie de 2^{19} fois au moyen d'un compteur.

B.3.3 Résultats

Le nombre de blocs disponible sur le Spartran 3 utilisé ne nous permet pas de traiter une population de plus de 64 oscillateurs en utilisant l'implémentation présentée précédemment. C'est décevant car il n'est pas possible de bénéficier de la vitesse du FPGA pour d'importantes populations.

Néanmoins, sur l'ensemble des populations testées, une synchronisation parfaite à été observée en quelque secondes pour les huit oscillateurs reliés aux LEDs de la cartes FPGA.

[1] Digilent : Nexys™2 Spartan-3E FPGA Board specifications, http://www.digilentinc.com/Products/Detail.cfm?Prod=NEXYS2

Annexe C

Compilateur MATLAB vers Java

C.1 Avant propos

L'ensemble des algorithmes présentés aux chapitres 3 et 4 ont été ébauchés avec le langage MATLAB [MATLAB, 2010]. La plateforme d'analyse de l'écriture présentée en annexe A est écrite en Java. Cette plateforme doit embarquer les algorithmes en question. Il est fastidieux de traduire à la main du code MATLAB en Java. Par ailleurs, si des modifications algorithmiques sont apportées à l'une des deux implémentations, il faut propager ces modifications sur l'autre implémentation. Il a fallu trouver un moyen d'automatiser le processus de traduction. Trois solutions ont été envisagées en section A.2.2. La dernière solution s'appuie sur un traducteur MATLAB vers Java, dont des éléments de conception sont donnés en suivant.

C.2 Introduction

MC (pour MATLAB Compiler) a pour but de traduire du code écrit en MATLAB vers plusieurs langages cibles. L'implémentation courante de MC propose un seul langage cible : Java. Par ailleurs, seul un sous-ensemble du langage MATLAB est pour le moment supporté.

MATLAB est une suite logicielle programmable (avec un langage du même nom) de calcul matriciel haute performance [Hanselman et Littlefield, 1997]. Le langage MATLAB permet de manière concise de spécifier des enchaînements de calculs en tout genre. En particulier, les opérations matricielles sont accessibles de manière directe. Par ailleurs MATLAB vient avec une impressionnante librairie de fonctions et d'outils divers allant du domaine des statistiques jusqu'au domaine de l'apprentissage en passant par le traitement du signal. Cette profusion de fonctionnalités est l'un des enjeux majeurs dans la réalisation de MC puisque pour pouvoir traduire et faire fonctionner tout code MATLAB, il devra supporter ces outils et ces fonctionnalités de la manière la plus exhaustive possible pour être utile au plus grand nombre.

En l'état actuel, MC est composé de deux parties, une partie traducteur qui traduit le code MATLAB en Java et une partie librairie, MCJavaCore écrite en Java, qui implémente les fonctions intrinsèques de MATLAB. Les deux sections suivantes s'attardent sur chacun de ces deux aspects. Nous présenterons ensuite le fonctionnement général de MC et nous conclurons par une discussion.

C.3 Le traducteur MC

C.3.1 Fondements

Les deux principaux outils utilisés pour réaliser ce traducteur sont Flex et Bison.

Flex (pour "Fast Lexical Analyzer"), réécriture de Lex [Lesk et Schmidt, 1975], permet de générer des analyseurs lexicaux à partir d'un fichier de description fourni par l'utilisateur. Un analyseur lexical permet de traduire un programme source en unités lexicales (tokens en anglais), constituées de suites de caractères représentées par des expressions régulières. Par exemple, la suite de caractères ('i','f') dans le fichier source fera émettre à l'analyseur lexical le token IF.

Bison, basé sur yacc [Johnson, 1975], permet de générer un analyseur syntaxique à partir d'un fichier de description de grammaire non contextuelle LALR(1). Nous n'entrerons pas dans les détails de l'utilisation de ce programme ni dans l'explication de ce qu'est une grammaire LALR : le lecteur est invité à lire Aho, Lam, Sethi, et Ullman [2007]. La seule chose à savoir est qu'un langage de programmation peut s'écrire sous forme de règles de production qui permettent de le décrire (ie. décrire l'ensemble des programmes valides pour ce langage) : c'est ce qu'on appelle une grammaire. Par exemple, un programme S peut être une suite d'instructions. On peut donc écrire que $S \longrightarrow I\ IS$ où I est une instruction et IS une suite d'instructions. Ensuite I peut être par exemple une expression E donc on aura la règle de production $I \longrightarrow E$. E peut être un chiffre donc $E \longrightarrow chiffre$. $chiffre$ est un token reconnu par l'analyseur lexical avec cette règle (sous forme d'expression régulière) : '[0-9]'. L'analyseur syntaxique permet de reconnaitre la suite des règles (tirées de la grammaire) permettant de décrire le programme analysé, et pour chaque règle reconnue, d'effectuer une action spécifiée par l'utilisateur (de Bison). Cette action peut être de n'importe quel type. Dans notre cas, il s'agit de faire les opérations nécessaires à la génération du code Java.

Le langage MATLAB a quelques particularités tant au niveau lexical qu'au niveau syntaxique qui font qu'il peut être difficile à spécifier de manière simple pour les outils précités. Afin de nous aider dans cette tâche nous nous sommes inspirés des travaux de Abhay, Joisha, et Kanhere [1999]. La documentation MATLAB est aussi un passage obligé pour rendre compte exactement de la sémantique du langage, en particulier la page définissant les opérateurs et leurs priorités a été d'une grande aide [Hanselman et Littlefield, 1997].

C.3.2 Analyseur lexical pour le langage MATLAB

L'analyse lexicale d'un programme MATLAB peut poser plusieurs problèmes. Leur résolution peut avoir pour conséquence que le traducteur MC et MATLAB n'ont pas exactement le même comportement dans certains cas limites, mais, cela n'a pas trop de conséquences sur l'utilisateur. Nous décortiquons ici deux problèmes nous paraissant intéressants.

C.3.2.1 Le caractère '

En MATLAB le caractère ' peut avoir deux significations : il peut déterminer le début et la fin d'une chaîne de caractères ou bien définir l'opération de transposition :

```
B = 'A' %ici la variable B contiendra la chaine de caracteres 'A'
B = A'  %ici B contiendra la transpose de la matrice A
```

La règle qui permet de différentier ces 2 formes est la suivante : ' sera un opérateur de transposition si, et seulement si, ce caractère suit immédiatement un entier, un flottant,

un imaginaire, un identifiant, un opérateur de transposition, une parenthèse fermante ou un crochet fermant.

Pour pouvoir repérer ce genre de situations dans Flex, une variable d'état S est disponible. Il est possible d'associer à une règle permettant de reconnaître un élément lexical, un pré-requis, basé sur l'état de S : la règle ne peut alors être utilisée dans l'analyse lexicale qu'aux moments où S est dans un état attendu. Ainsi il est possible d'avoir le comportement voulu, si chaque fois que l'on reconnaît un entier, un flottant, un imaginaire, un identifiant, un opérateur de transposition, une parenthèse fermante ou un crochet fermant, l'on met la variable S dans un état particulier qui sera nécessaire à l'application de la règle reconnaissant l'opérateur de transposition. Si S n'est pas dans le bon état, cela signifie que le ' rencontré commence une chaîne de caractère.

C.3.2.2 Les matrices

La syntaxe MATLAB permet de définir les matrices par une suite de valeurs entre crochets où les lignes sont séparées par un ';' ou par un retour à la ligne, et les éléments de chaque ligne par une ',' ou un espace. Or, en dehors des matrices, l'espace est en général ignoré. Il faut donc, si l'on est dans une matrice émettre un token associé au caractère ' ' alors que ce n'est pas nécessaire dans les autres cas. Pour ce faire, on utilise l'analyseur syntaxique qui, si l'on entre dans une matrice, active un drapeau qui est utilisé par l'analyseur lexical pour ignorer les espaces ou au contraire émettre un token de séparation de colonne.

Une deuxième ambiguïté en ce qui concerne la syntaxe des matrices est celle-ci :

```
A = [1,2,3]'
[A,B,C] = func(in_args)
```

Dans le premier cas une matrice est définie et affectée à A, dans le deuxième cas trois valeurs de sortie d'une fonction $func$ sont affectées chacune à une variable. Dans le premier cas les crochets délimitent une matrice et dans le second cas une liste d'arguments de sortie dont la sémantique est totalement différente.

Pour résoudre ce problème, on demande à l'analyseur lexical d'émettre des tokens différents pour les crochets ouvrants et fermants dans chacun des deux cas. Dans Flex, il existe une fonctionnalité permettant de gérer ces cas. En effet, pour chaque token, Flex permet non seulement de donner une expression régulière de celui-ci pour la reconnaissance, mais aussi de donner une expression régulière de ce qui suit. Ainsi, on peut différencier les deux cas en spécifiant ce qui doit suivre. Dans le second cas, le crochet ouvrant sera toujours suivi de n'importe quoi (à quelques exceptions près), d'un crochet fermant, d'un nombre quelconque d'espaces et d'un signe égal. Quant au crochet fermant, il sera suivi d'un nombre quelconque d'espaces et d'un signe égal. Cela permet de résoudre cette ambiguïté dans une majorité de cas.

C.3.3 Analyseur syntaxique

La syntaxe MATLAB étant très permissive, l'analyse syntaxique n'en est que plus compliquée. Par ailleurs, la sémantique associée peut être elle aussi complexe à mettre en œuvre à cause de cette permissivité. Nous montrons deux cas particuliers que sont l'opération ' :' et l'utilisation du mot clé 'end'.

C.3.3.1 Exemple d'ambiguïté

Les expressions basées sur ' :' ('colon expressions' en anglais) permettent de définir de manière synthétique des vecteurs lignes dont les éléments suivent une progression arith-

métique. Par exemple, $A = 1 : 4$ est équivalent à $A = [1, 2, 3, 4]$. De même, $A = 1 : 2 : 4$ produit $A = [1, 3]$. On remarque donc qu'il y a deux types d'expressions ' :' : $debut : fin$ et $debut : pas : fin$. Si $debut > fin$ un vecteur vide est produit (de taille 1×0 !).

Étrangement, il est aussi possible d'empiler l'opérateur ' :', au prix de la lisibilité. On peut par exemple écrire $2 : 5 : 10 : 10$ et $2 : 5 : 10 : 3 : 10$: ' :' étant associatif à gauche, ces expressions reviennent à $(2 : 5 : 10) : 10$ et $(2 : 5 : 10) : 3 : 10$. Par ailleurs, si un vecteur est utilisé comme argument de cette opérateur, alors son premier élément est pris en compte. Ainsi, dans le premier cas on obtient $[2, 3, 4, 5, 6, 7, 8, 9, 10]$ et dans le second $[2, 5, 8]$.

On peut obtenir ce comportement en utilisant ces règles de production :

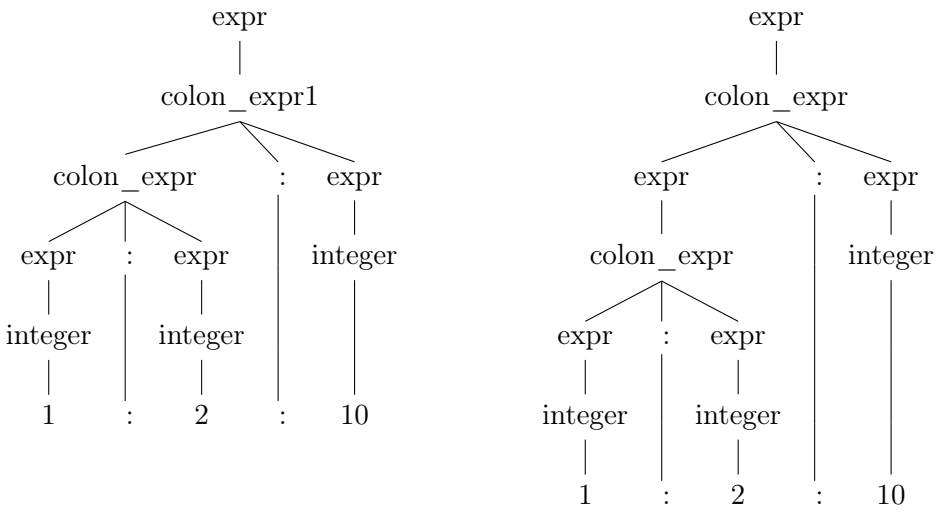
```
expr : colon_expr {
$$.source = "colon("+$1.start+",matrixFromDouble(1),"+$1.stop+")";
}
     | colon_expr1 {
$$.source = "colon("+$1.start+","+$1.stride+","+$1.stop+")";
}
;

colon_expr :   expr ':' expr {
  $$.start = $1.source;
  $$.stride = $3.source;
  $$.stop = $3.source;
}
;

colon_expr1 : colon_expr ':' expr {
$$.start = $1.start;
$$.stride = $1.stride;
$$.stop = $3.source;
}
;
```

Dans ce code, $$ désigne l'élément à gauche dans les règles de production ($\longrightarrow$ y étant représenté par un ' :'). $n désigne le n^e élément à droite de ' :'. Ces symboles commençant par $ désignent des variables pouvant contenir des champs que l'on définit comme l'on veut. Le champ *source* contient le code généré en Java. le champ *start* contient la valeur de départ de l'expression colonne, *stride* son pas et *stop* sa fin. Le code Java généré fait appel à une fonction *colon* définie dans MCJavaCore qui génère le vecteur désiré.

Ces règles sont ambigües : l'analyseur syntaxique a deux façons d'interpréter $1 : 2 : 10$ en tenant compte de l'associativité à gauche (sinon il y en aurait plus) :



On remarque qu'une fois le premier *colon_expr* réduit, on a le choix entre décaler en utilisant la règle de production de *colon_expr*1 (arbre de gauche) et réduire *colon_expr* en *expr* en utilisant la règle de production $expr \longrightarrow colon\_expr$ (arbre de droite). C'est ce qu'on appelle un conflit décalage/réduction. Pour résoudre ce genre de conflit, Bison favorise le décalage par défaut : ce qui dans notre cas, et dans la plupart des cas, est la bonne solution.

C.3.3.2 Cas particulier du *end*

Le *end* est un élément lexical utilisé dans deux cas. Il est d'abord utilisé pour signifier la fin d'un bloc (bloc conditionnel, boucle, fonction). Il peut aussi désigner, lorsqu'il est utilisé dans une référence, le dernier élément de la variable à laquelle on fait référence. Par exemple, on peut écrire $A(5 : end-1)$ qui signifie que l'on veut accéder à une sous matrice de A de son 5^e à son avant-dernier élément. Pire, on peut aussi écrire $A(1 : end, 4 : end)$. Dans ce cas, le premier *end* correspond au nombre de lignes de A et le deuxième *end* correspond au nombre de colonnes. Différencier ces deux types *end* en utilisant l'analyseur lexical est assez facile, il suffit de chercher s'il est entouré de parenthèses. Par contre comment rendre la signification du deuxième *end* en Java ?

Pour gérer correctement ces expressions, *end* est d'abord placé dans le code généré tel quel. Puis au moment de la réduction d'une règle correspondant à une référence, une fonction est appliquée au code source généré de cette référence, qui a pour but de remplacer ces *end* par leur véritable valeur :

```
void replaceEnds(const string &var_name,int out_ref,string &source){
  if(out_ref == 2){
    //remplacer end par A.lentgth et A[0].length
    int pos = source.find("end");
    if(pos !=-1){
      source.erase(pos,3);
      string rep = "matrixFromDouble("+var_name+".length)";
      source.insert(pos,rep);
    }
    pos = source.find("end");
    if(pos !=-1){
      source.erase(pos,3);
      string rep = "matrixFromDouble("+var_name+"[0].length)";
```

```
      source.insert(pos,rep);
    }
  }else if(out_ref == 1){
    int pos =1;
    while(pos!=-1){
      pos = source.find("end");
      if(pos !=-1){
        source.erase(pos,3);
        string rep = "numel("+var_name+")";
        source.insert(pos,rep);
      }
    }
  }
}
```

Cette fonction cherche d'abord combien d'arguments sont passés à la référence. S'il n'y en a qu'un seul, *end* est remplacé par $numel(A)$ (qui renvoie le nombre d'éléments de A). Si deux arguments sont passés à la référence, *end* est remplacé par $A.length$ (qui renvoie le nombre de lignes de A) dans le premier argument, et $A[0].length$ (le nombre colonne de A) dans le deuxième argument (les matrices dans MCJavaCore sont représentées par des $double[][]$).

C.3.4 Conclusion

Nous voyons donc que le langage MATLAB voulant apporter des facilités à l'utilisateur, et voulant à tout prix minimiser le nombre d'erreurs de syntaxe, nous force à gérer des cas complexes. Pourquoi autoriser plusieurs séparateurs lors de la définition des matrices ? Pourquoi autoriser des expression contenant plus de 2 ' :' à la suite ?

C.4 MCJavaCore

MCJavaCore est la librairie Java fournissant les fonctions matricielles de base sur lesquelles repose le code traduit par MC.

C.4.1 Type

Lorsqu'il utilise MATLAB, l'utilisateur n'a guère à se soucier des types de données qu'il utilise. Le typage est dynamique et transparent. Au contraire le language Java est fortement typé et l'utilisateur doit se soucier des types des objets qu'il manipule. La traduction de code MATLAB vers Java doit donc prendre en compte cet aspect.

Dans notre première implémentation de MCJavaCore, nous avons supposé que tout ce qui était manipulé était matrice et avait un type unique : $double[][]$. Les avantages de ce choix sont que les fonctions de MCJavaCore n'ont pas à être implémentées plusieurs fois pour supporter plusieurs types. Ce choix à aussi plusieurs inconvénients. Premièrement, il alourdit certains calculs : un vecteur ou un scalaire seront toujours représentés par un $double[][]$. Ensuite, il ne permet pas de représenter des matrices à 0 éléments dont le nombre de colonnes est non nul (une étrangeté matlab qui permet d'avoir des matrices vides de taille non nulle). Les structures plus compliquées de MATLAB nécessitent aussi des structures plus complexes dans Java (et donc d'autre types, vouloir se cantonner à un seul type est dès lors caduque) ; on peut citer les *cellarrays* (représentées alors par des

double[][][]). Enfin les matrices dans MATLAB peuvent avoir plus de deux dimensions ce que ne permet pas le type choisi *double*[][]. Avec le recul ce choix semble peu judicieux.

C.4.2 Fonctions de base

Le code généré par le traducteur MC transforme les opérateurs MATLAB en appels de fonctions de base qui sont implémentées dans MCJavaCore. Le fonctionnement semble être identique dans MATLAB puisque chaque opérateur a une fonction qui lui est associée et que l'on peut appeler telle quelle dans le code.

Les opérations de base dans MATLAB peuvent être divisées en plusieurs groupes. D'abord, il y a les opérateurs arithmétiques (addition, soustraction ...) qui sont présents dans nombre de langages de programmation. Ici, ces opérateurs sont un peu plus compliqués puisqu'ils s'appliquent non seulement à des scalaires, mais aussi à des vecteurs et des matrices. Lorsque des opérations sont sémantiquement ambiguës, comme par exemple la multiplication, qui peut signifier un produit matriciel ou un produit élément par élément, deux opérateurs sont fournis. Ils sont alors invocables avec une syntaxe différente (par exemple pour la multiplication '*' et '.*').

Viennent ensuite les opérateurs relationnels (>, >=, ...), qui comparent les matrices et les vecteurs élément par élément et renvoient une matrice de 0 et de 1. Les opérateurs logiques (&&, &, ||, |, ...) sont eux aussi de deux types. Soit ils désignent une opération élément par élément, soit ils désignent une opération applicable seulement à des scalaires logiques (0 et 1). Le groupe des opérations bit à bit est aussi présent même si aucuns symboles ne leur sont associés : il faut les appeler par leur nom (*bitand*, *bitxor*, ...), comme une fonction. On note aussi la présence des opérations sur les ensembles (*union*, *unique*, ...). Ces deux groupes ne sont, pour le moment, pas supportés par MC.

Le dernier groupe d'opérations est associé à des expressions particulières. Il peut s'agir, par exemple, des opérateurs de référence ou d'affectation : $A(1,4)$ utilise la fonction de base $subsref$, et, $A(1,4) = 8$ appelle la fonction $subsasgn$.

C.5 Fonctionnement général

C.5.1 Gestion des symboles et des fonctions

En MATLAB les identifiants peuvent désigner des noms de variable ou des noms de fonction. Par ailleurs, à chaque instant un symbole peut être redéfini et changer de nature. Lors de la compilation il faut tenter de conserver à chaque instant la nature d'un symbole. En effet imaginons que l'on doive traduire :

```
B = 2 + A(2,4)
```

Si a est une variable alors $A(2,4)$ sera traduit en utilisant la fonction $subsref$ qui permet d'accéder à des sous-éléments de la variable A. Mais si A est une fonction, alors il sera traduit par un appel à la fonction A qui sera elle aussi traduite.

Si l'on a pas accès à la nature de ce que désigne l'identifiant (fonction ou variable) on suppose par défaut qu'il s'agit d'une fonction, car s'il s'agissait d'une variable il est probable que nous en aurions eu une déclaration préalable dans le programme source.

C.5.2 Organisation du code généré

Lorsque l'on compile un fichier .m, il s'agit soit d'un script, soit d'une fonction (en général). Dans tous les cas on génère une classe qui contiendra l'ensemble des fonctions qui seront

traduites. Le code d'un fichier .m non inclus dans une fonction sera mis dans une fonction *main*, la classe générée sera alors exécutable.

MC gère une liste de fichiers à traduire. Lorsqu'il lit le fichier qu'on lui a fourni en argument et qu'il doit interpréter une syntaxe de type référence $ID(list\_arg)$, si ID n'est pas une variable alors il regarde si c'est une fonction connue (implémenté dans MCJJava-Core). Si ce n'est pas le cas, il va rechercher directement dans les fichiers fournis par les boîtes à outils de MATLAB (la liste de ces fichier est générée grâce à un petit utilitaire fourni avec MC). Ce fichier est alors ajouté à la liste des fichiers à traduire. Lorsque l'on arrive à la traduction de ce fichier, trois cas peuvent se présenter :

- le fichier est écrit en MATLAB et il est traduit sans erreurs, dans ce cas pas de problème la traduction peut continuer,
- le fichier est en MATLAB mais utilise des éléments non encore supportés par MC, dans ce cas il faut soit réécrire la fonction qu'il contient dans MCJavaCore, soit faire en sorte que MC supporte ces éléments,
- le fichier ne contient que des commentaires, ce qui signifie que la fonction n'est pas écrite en MATLAB mais qu'elle est présente sous forme de code natif compilé dans MATLAB, dans ce cas il faut l'écrire dans MCJavaCore.

A l'avenir, le premier cas devrait devenir majoritaire.

Notons que certaines fonctions natives de MATLAB (non écrites en MATLAB) peuvent être avantageusement réécrites en MATLAB plutôt qu'en Java : cela permet de réduire le nombre de fonctions dans MCJavaCore, ce qui impliquera moins de travail lors de l'ajout du support d'autres langages à MC.

C.5.3 Mise en place d'une suite de tests

Afin de tester les calculs effectués par le code généré par MC ainsi que les fonctions incluses dans MCJavaCore, une suite de tests a été mise en place. Les tests sont des fichiers MATLAB, particuliers, où l'on impose que la première ligne soit de la forme :

```
%A,B
```

Cette ligne spécifie quelles variables présentes dans la suite du fichier seront testées (ici A et B).

Le fonctionnement automatique de la suite de test est géré par un script écrit en bash. Ce script lance d'abord MATLAB et lui fait lire les commandes sur un pipe qui a été préalablement créé.

```
mkfifo tomatlab #pipe creation
matlab -nodesktop < tomatlab > matlaboutput &
```

Pour chaque fichier on lance le programme `lmsjr` qui envoie le code à MATLAB au travers de ce pipe :

```
./lmsrj $i >tomatlab
```

Ce programme lit le fichier de test (écrit en MATLAB, donc) ligne par ligne et envoie chacune des lignes à MATLAB qui les exécute normalement. Ensuite ce programme va lire l'entête du fichier MATLAB pour avoir accès aux variables qui vont être comparées par le test. Il fait alors appeler par MATLAB une fonction qui exporte ces variables en Java. Voici un exemple de fichier généré par cette fonction MATLAB appelée *exportVar* :

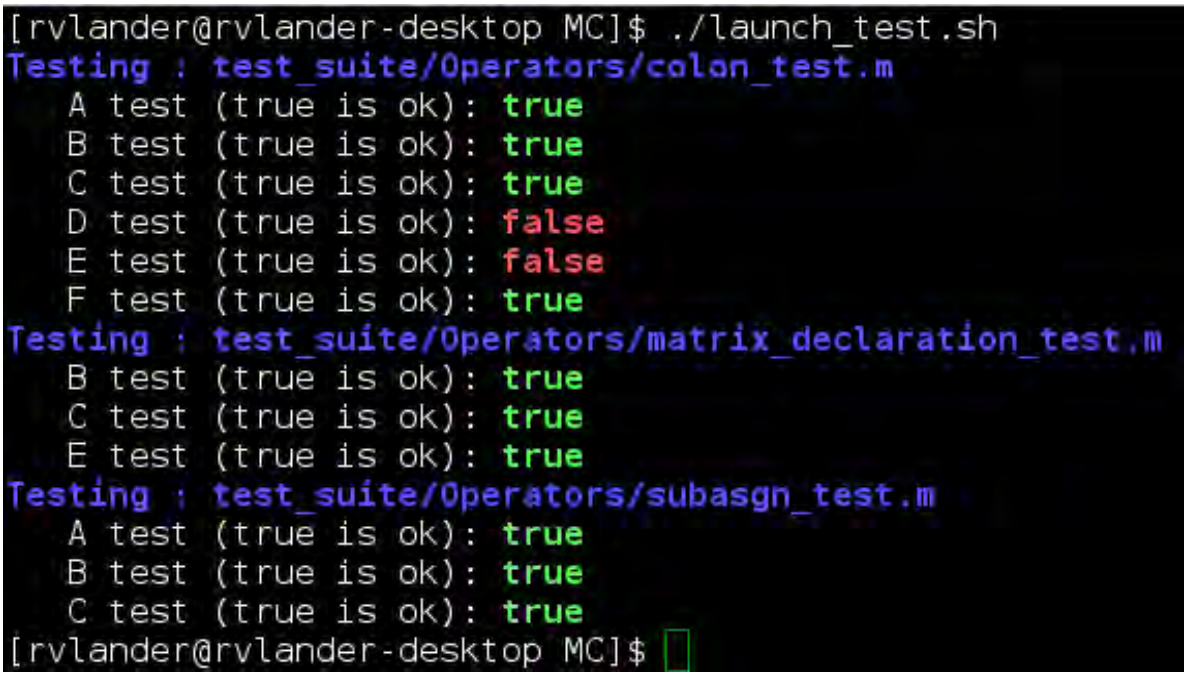


FIGURE C.1 – Retour visuel de l'exécution d'une série de tests pour MC.

.

```
double[][] A_matlab_result = {
{1.000000,2.000000,3.000000,4.000000}
};
double[][] B_matlab_result = {
{1.000000,3.000000,5.000000,7.000000,9.000000}
};
```

Ensuite, MC est exécuté sur le fichier test en mode test, ce qu'il lui fera inclure le fichier créé par *exportVar* dans le code généré. Dans la méthode `main` générée par MC, les résultats produits par MATLAB sont alors comparés aux valeurs générées par la traduction MATLAB. La figure C.1 montre le retour visuel d'une suite de test.

C.6 Discussion

Le langage MATLAB est un langage très efficace pour échafauder des algorithmes basés sur des calculs numériques compliqués. De plus, nombre de boîtes à outils, apportant des fonctions avancées, sont fournies. La syntaxe permettant une mise en place concise des calculs matriciels augmente la lisibilité d'un code dont le fond peut déjà être compliqué. Cependant, dépendre de MATLAB pour déployer des applications n'est pas concevable, c'est en ce sens qu'il a été décidé de créer un traducteur MATLAB vers d'autres langages de programmation. Nous avons présentés certains aspects de la réalisation de MC. Il aurait été trop long, et ce n'est pas le rôle de cette thèse de présenter tous ces travaux en détail.

En ce qui concerne les améliorations à apporter, elles concernent bien entendu un meilleur support du langage MATLAB ; en particulier de ses structures les plus complexes (telles que les cell_arrays, les enregistrements et les poignées sur les fonctions). Il faut ajouter de nouvelles fonctions dans MCJavaCore et aussi améliorer celles déjà présentes. Enfin, ajouter le support d'un langage tel que Python semble aussi une piste envisageable ; d'autant que Python grâce à Scipy [Jones, Oliphant, et Peterson], numpy et matplotlib

[Hunter, 2007], implémente déjà nombre de fonctionnalités fournies par MATLAB. Ce langage semble d'ailleurs de plus en plus utilisé dans le monde du calcul hautes-performances [Chudoba, Sadílek, Rypl, et Vořechovský, 2013].

Pour finir sur MC, nous aimerions dire quelques mots sur la performance. Il semblerait que le code traduit s'exécute plus rapidement que le code MATLAB, mais des tests sont à faire. Par contre, le code traduit par MC en Java est une dizaine de fois moins efficace que celui traduit manuellement. Cela s'explique par le fait que la traduction manuelle permet de faire beaucoup d'optimisations : entre autres, le type *double*[][] n'est pas utilisé dans la traduction manuelle pour tous les éléments manipulés, les accès aux éléments d'une matrice se font de manière plus directe sans passer par la fonction *subsref* qui n'est pas très optimisée. Bien sûr le temps passé à la traduction manuelle est lui beaucoup plus conséquent que d'utiliser MC.

Annexe D

Simulateur d'environnement virtuel par crochetage d'API

Dans le chapitre 5, nous avons présenté un algorithme très simple de conduite automatique fonctionnant selon le principe du bâton des aveugles. Pour pouvoir le tester, nous avons dû mettre en place un simulateur de conduite. Notre but était de pouvoir diriger une automobile le long d'une route, en ayant comme source d'information uniquement une vision frontale de l'environnement, et comme pouvoir d'action, les commandes d'accélération, de freinage et de direction (gauche et droite). Dans la suite, nous présentons les aspects techniques de la mise en place de notre simulateur. Nous commençons par présenter l'architecture de notre système.

L'architecture adoptée pour notre système de simulation (figure D.1), faisant communiquer la partie intelligence et la partie simulateur par réseau, permet de remplacer chacun des deux modules pourvu qu'il respecte le protocole réseau établi (qui est très simple). Le module "intelligence" peut donc rapidement être testé sur plusieurs modules "simulateur". Par exemple, au cours de nos travaux, nous avons été amenés à remplacer la partie simulateur par un "module simulateur" réel, c'est à dire une voiture électrique, munie d'un caméra USB, guidée par Wifi.

Nous développons ici la solution adoptée, permettant d'obtenir un simulateur de conduite réaliste, avec un environnement détaillé, s'approchant du réel. L'idée principale est d'utiliser les jeux de simulation de course du marché comme briques centrales de ce simulateur, et de les adapter à nos besoins grâce à des méthodes qui vont être présentées au cours de cette annexe.

L'intérêt d'utiliser des jeux vidéos pour notre simulateur est double : le coût faible et le réalisme de la simulation est élevé.

D.1 Partie réseau

Nous présentons ici en détail la partie réseau où siègent les échanges d'images et de commandes entre les deux modules présentés précédemment. Les échanges se font au travers de deux connexions TCP (en mode connecté), l'une étant dédiée à la réception des image (connexion IM) l'autre à l'envoi des commandes (connexion CO).

Les deux sockets serveur sont mis en place dans des threads indépendantes, côté Simulateur, mis en regard, côté Intelligence, de deux sockets clients. La connexion IM fonctionne comme suit : chaque fois que le côté Intelligence à besoin d'une image il envoie un message GET sur la liaison et attend de recevoir l'image (il ne peut envoyer un deuxième GET avant réception de l'image précédemment demandée). Sur la liaison CO, il envoie à in-

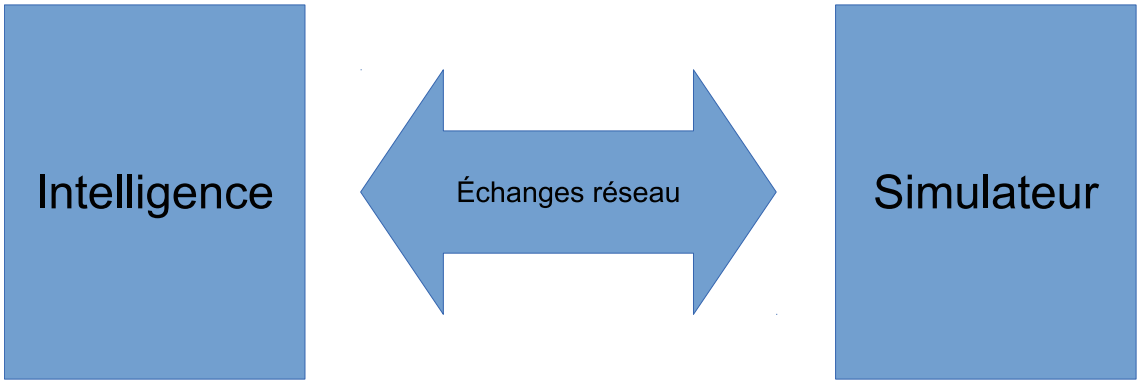


Figure D.1 – Architecture simplifiée du système de simulation. Deux modules constituent ce système : un module "Intelligence" et un module "Simulateur" qui communiquent selon un protocole réseau simple.

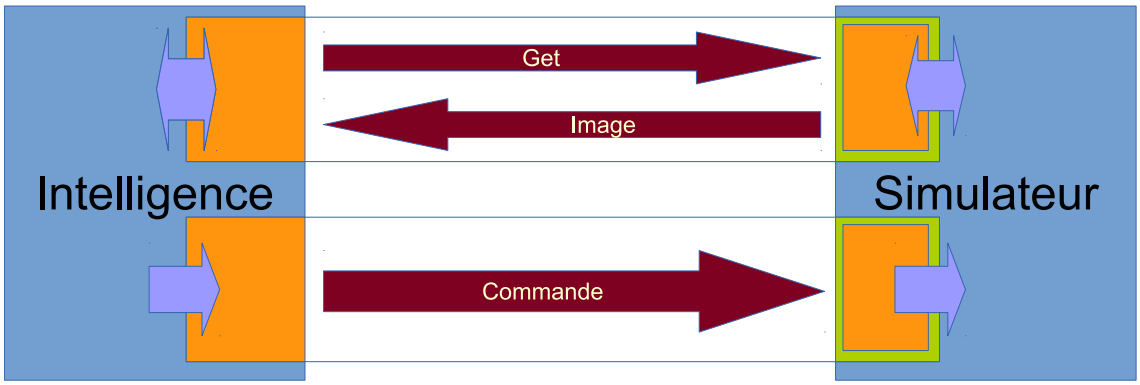


Figure D.2 – Schéma représentant les échanges réseau entre les deux parties applicatives. En haut, la connexion IM, en bas la connexion CO. Les rectangles oranges représentent les sockets, les rectangles verts les threads dans lesquels ils sont encapsulés (quand c'est le cas).

tervalle régulier les commandes sous forme de deux octets représentant chacun un axe de commande : l'axe de direction et l'axe d'accélération.

D.2 Transformer un jeu vidéo en simulateur réaliste

Un jeu vidéo utilise une simulation réaliste d'un environnement et permet à un utilisateur d'y naviguer à l'aide d'un contrôleur. Afin de faciliter la tâche de l'utilisateur, un retour visuel de la situation est affiché à l'écran. Notre but était de modifier le comportement du jeu afin qu'il puisse implémenter le protocole réseau présenté dans la section précédente, qu'il permette la capture de l'image affichée à l'écran et l'envoie au module Intelligence, qu'il soit capable d'interpréter et d'utiliser les commandes envoyées par le réseau par ce dernier. Tout ceci est effectué sans accès au code du jeu, comme c'est le cas de la plupart des applications non libres du marché.

L'idée va être de faire exécuter à ce programme du code qui n'est pas le sien. Ceci sera présenté en partie D.2.3. Dans un premier temps on se propose de présenter en détail les fonctions qui seront ajoutées.

D.2.1 Partie commande

Afin de pouvoir commander le jeu à distance, on utilise l'application PPJoy, qui permet de définir des manettes de jeu virtuelles. Ces manettes virtuelles peuvent ensuite être commandées de manière automatique. L'utilisation de ces manettes à l'intérieur du jeu est simple puisqu'il suffit de configurer les commandes du jeu pour les utiliser comme pour n'importe quelle autre manette.

PPJoy permet l'ajout de ces périphériques virtuels via un interface graphique. L'état de ces périphériques est ensuite mis à jour dans l'application grâce à une API fournie par PPJoy. Une fonction de l'API permet de se connecter au joytick virtuel, puis de l'utiliser de manière cyclique, une autre fonction permet de mettre à jour l'état de celui-ci grâce à un enregistrement représentant l'état de ce joystick.

Ce code est intégré dans le thread serveur de la connexion CO, qui traduit les deux octets de commande reçus en commande pour le périphérique virtuel PPJoy. La figure D.3 montre les interactions entre PPJoy, le jeu et notre code. Insistons sur le fait que nous faisons exécuter la commande par le jeu. Cependant, on pourrait très bien utiliser une application tierce pour cela, puisqu'il suffit, pour que la commande soit effective, de changer l'état du périphérique virtuel, ce qui ne nécessite en rien d'avoir accès à une quelconque donnée du jeu.

D.2.2 Partie vidéo

Ici, le socket serveur IM écoute dans l'attente d'une commande GET qui ordonnera la capture de l'image dans le jeu et l'enverra sur la connexion IM.

On suppose que le jeu de simulation utilise DirectX. DirectX est une librairie graphique fournie par Microsoft permettant entre autre la gestion et l'affichage d'environnements 3D. Cette librairie est utilisée dans la plupart des jeux vidéos PC du marché.

Pour gérer l'affichage des éléments dans la scène, le thread principal du jeu fait appel à des fonctions fournies par l'API DirectX. Nombre de ces fonctions modifient le FrameBuffer, un buffer contenant l'image en cours de création. A chaque fin de boucle le FrameBuffer est envoyé à l'écran, puis est vidé pour que la scène soit de nouveau rendue à la boucle suivante.

Notre but est de capturer le FrameBuffer, si la commande GET nous a été envoyée, juste avant qu'il ne soit envoyé vers l'écran. Ceci est effectué dans DirectX par la fonction `EndScene()`. Donc avant l'appel à `EndScene()` il nous faut faire en sorte que le FrameBuffer soit copié vers un autre tampon dont les données seront ensuite envoyées sur la liaison réseau.

La figure D.4 résume les différentes interactions entre les différents éléments du simulateur permettant d'obtenir une image depuis le réseau. On voit que du code est ajouté dans les threads d'affichage du jeu. Or, comme nous l'avons dit en introduction, nous ne pouvons pas ajouter du code dans le jeu. Il faut donc trouver un moyen détourné pour obtenir ce comportement. C'est ce qui va être expliqué dans la section suivante.

D.2.3 Injection de dll et détournement d'appels

D.2.3.1 Faire exécuter du code par une application non modifiable

Un moyen pour changer le comportement d'un exécutable est de lui faire charger dynamiquement une librairie à laquelle il n'est pas lié à priori : on parle d'injection de dll (sous Windows). L'un des moyens de faire cela et d'utiliser la librarie Mircosoft Detours.

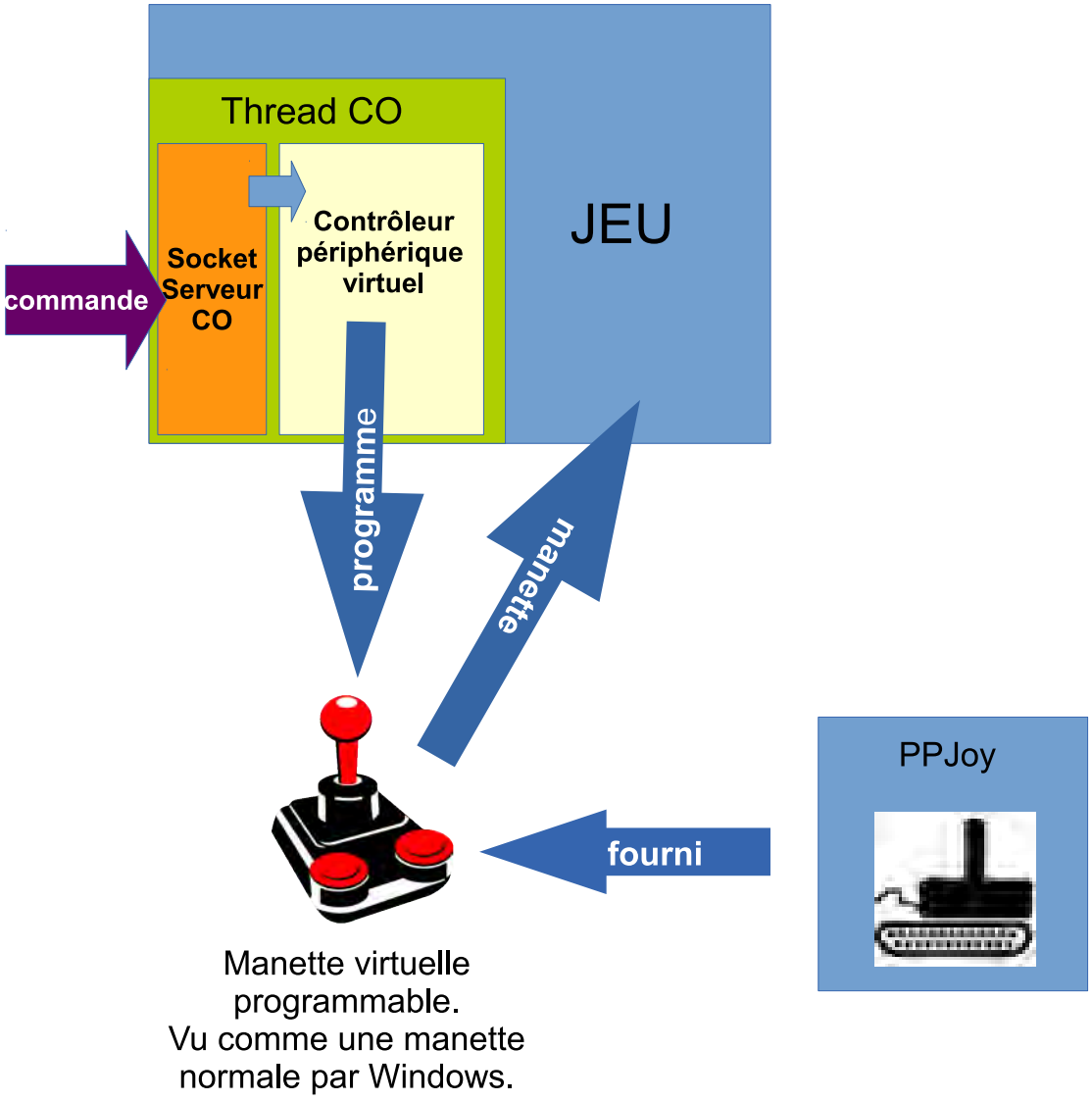


Figure D.3 – Schéma explicatif de la commande dans le simulateur. PPJoy fournit un périphérique virtuel programmable. Ce dernier est programmé par le thread CO, de commande, exécuté par le jeu. Ce thread lit la commande sur le réseau. Il est à noter que ce thread pourrait très bien ne pas être exécuté par le jeu mais comme un programme à part : cela n'aurait aucune incidence sur le fonctionnement global.

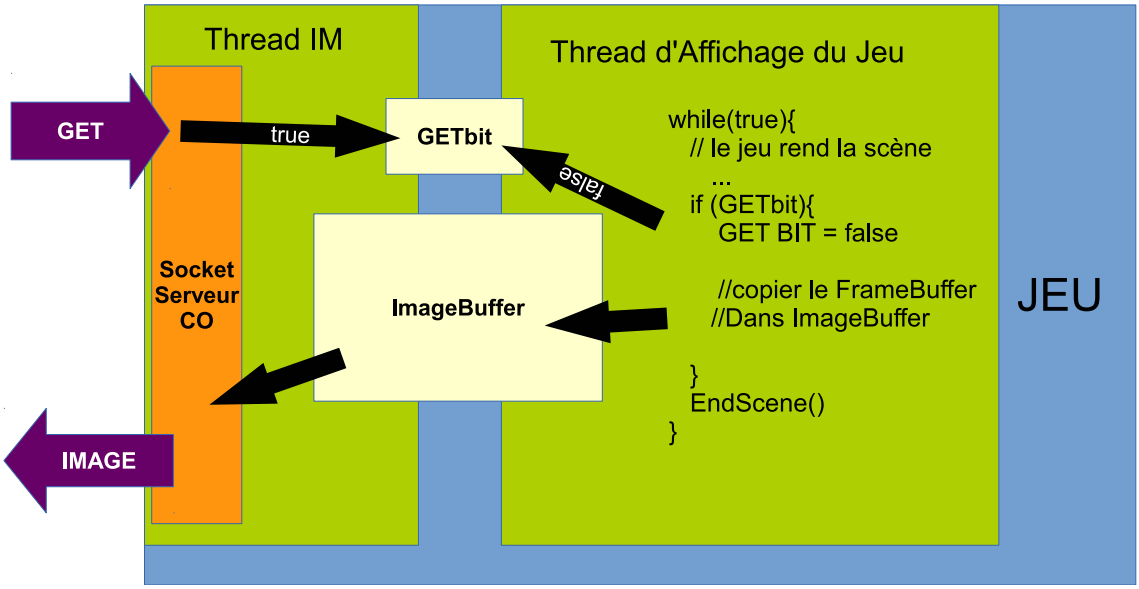


FIGURE D.4 – Schéma explicatif du rapatriement d'une image du jeu vers le réseau. Le socket serveur IM reçoit la commande GET qu'il répercute dans le bit partagé avec le thread d'affichage du jeu, qui, au prochain affichage agira : le FrameBuffer sera alors copié dans ImageBuffer qui est partagé avec le Thread IM, qui pourra alors envoyer l'image rendue sur la connexion IM

.

La fonction `DetourCreateProcessWithDll()` de l'API de Detours nous permet de créer un processus avec une dll particulière (liée dynamiquement).

Au chargement d'une dll par un programme, la méthode `DllMain()` de cette librarie est exécutée : c'est à cet endroit que l'on peut créer et démarrer les Thread IM et CO.

Il nous reste à expliquer comment l'on intercepte l'appel du jeu à la fonction `EndScene()` de DirectX afin de pouvoir récupérer l'image du FrameBuffer juste avant l'affichage.

D.2.3.2 Le crochetage d'API

L'API Detours permet aussi de remplacer dynamiquement l'appel d'une fonction par un appel vers une autre fonction (grâce à `DetourFunctionWithTrampoline()` fournie par l'API Detours). Pour comprendre le principe de la méthode utilisée ici, nous proposons de le montrer sur un cas simplifié. Le passage à DirectX est direct : la fonction `EndScene()` étant la méthode d'une classe DirectX qui subira le même traitement que la classe `Personne` de l'exemple suivant.

Supposons que nous crochetions la fonction suivante :

```
Personne * createPersonne(){
        return new Personne();
}
```

Supposons la classe `Personne` définie comme suit. On ne peut en modifier le code (on suppose que l'on n'a que sa version compilée) :

```
class Personne{
  public :
    virtual string getName(){
                return "Sophie" ;
```

```
    }

    virtual int getAge(){
                return 21;
    }

}
```

On peut alors créer notre propre fonction qui remplacera la fonction `createPersonne()` grâce au crochetage. Cette fonction a pour but d'insérer un objet proxy entre l'objet retourné par la fonction crochetée et l'application qui l'utilise (dans notre cas le jeu). L'objet proxy est défini comme suit :

```
class MyPersonne:public Personne{
  public :
    MyPersonne(Personne * pp):p(pp){}

    string getName(){
        //Ici vous pouvez faire ce que vous voulez
        return p->getName();
    }

    int getAge(){
        //Ici vous pouvez faire ce que vous voulez
        return p->getAge();
    }
  private:
        Personne *p;
}
```

et votre fonction :

```
Personne * my_createPersonne(){
        Personne * p = new Personne();
        return new MyPersonne(p);
}
```

On crochète la fonction `createPersonne()` pour la remplacer, grâce à Detours, par un appel à `my_createPersonne()` :

```
DetourFunctionWithTrampoline((PByte)createPersonne(),
                             (PByte)my_createPersonne());
```

De la sorte, lorsque l'application hôte (le jeu) appellera la fonction `createPersonne()`, elle appellera en réalité la fonction `my_createPersonne()` et recevra un objet `MyPersonne` (qui sert de proxy) vers la classe `Personne`, ce qui nous permet d'intercepter les appels aux méthodes de cette dernière.

Bibliographie

H Abelson, A di Sessa, et L Rudolph. Velocity space and the geometry of planetary orbits. *American Journal of Physics*, December 1975.

PJ Abhay, PG Joisha, et A Kanhere. The Design and Implementation of a Parser and Scanner for the MATLAB Language in the MATCH Compiler. Technical Report September, Center for Parallel and Distributed Computing, 1999.

AV Aho, M Lam, R Sethi, et J Ullman. *Compilateurs : principes, techniques et outils*. Pearson edition, 2007.

G André, V Kostrubiec, JC Buisson, JM Albaret, et PG Zanone. A Parsimonious Oscillatory Model of Handwriting and other 2D trajectories production. *Biological Cybernetics, en révision*.

P Arena. The Central Pattern Generator : a paradigm for artificial locomotion. *Soft Computing*, 4(4) :251–266, December 2000.

A Arenas, J Kurths, et Y Moreno. Synchronization in complex networks. *Physics Reports*, 469(3) :93–153, 2008.

T Artières, S Marukatat, et P Gallinari. Online handwritten shape recognition using segmental hidden markov models. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 29(2) :205–217, 2007.

S Athènes, I Sallagoïty, PG Zanone, et JM Albaret. Evaluating the coordination dynamics of handwriting. *Human movement science*, 23(5) :621–41, November 2004.

R Baillargeon. Reasoning about the height and location of a hidden object in 4.5- and 6.5-month-old infants. *Cognition*, 38(1) :13–42, January 1991.

R Baillargeon, ES Spelke, et S Wasserman. Object permanence in five-month-old infants. *Cognition*, 20(3) :191–208, August 1985.

PJ Beek. *Juggling dynamics*. Free University Press, 1989.

H Bergson. *Matière et mémoire*. Presses Universitaires de France, Paris, 1939.

N Bernstein. The problem of the interrelation of co-ordination and localization. *Archives of Biological Science*, 38 :15–59, 1935.

MH Bickhard. The interactivist model. *Synthese*, 166(3) :547–591, July 2008.

MH Bickhard. Interactivism : A manifesto. *New Ideas in Psychology*, 27(1) :85–95, April 2009.

MH Bickhard et L Terveen. *Foundational Issues in Artificial Intelligence and Cognitive Science*. Elsevier Science Publishers, 1995.

AE Bryson et YC Ho. *Applied optimal control : optimization, estimation, and control*. Taylor & Francis, 1975.

G Buccino, F Binkofski, GR Fink, L Fadiga, L Fogassi, V Gallese, RJ Seitz, K Zilles, G Rizzolatti, et HJ Freund. Action observation activates premotor and parietal areas in a somatotopic manner : an fMRI study. *European Journal of Neuroscience*, 13 :400–404, 2001.

JJ Buchanan, JH Park, et CH Shea. Target width scaling in a repetitive aiming task : switching between cyclical and discrete units of action. *Experimental Brain Research*, 175(4) :710–725, 2006.

JC Buisson. A rhythm recognition computer program to advocate interactivist perception. *Cognitive Science*, 28(1) :75–88, January 2004.

JC Buisson et JC Quinton. Internalized activities. *New Ideas in Psychology*, 28(3) :312–323, December 2010.

KP Burnham et DR Anderson. *Model selection and multi-model inference : a practical information-theoretic approach*. Springer-Verlag, New York, 2002.

G. Buzsáki. *Rhythms of the brain*. Oxford University Press US, 2006.

WD Byblow, RG Carson, et D Goodman. Expressions of asymmetries and anchoring in bimanual coordination. *Human Movement Science*, 13(1) :3–28, 1994.

H Chen, OE Agazzi, et CY Suen. Piecewise linear modulation model of handwriting. *Analysis and Recognition, IEEE Proceedings.*, 1(908) :363–367, 1997.

N Chomsky. *Aspects of the Theory of Syntax*. The MIT press, Cambridge, Massachusetts, 1965.

R. Chudoba, V. Sadílek, R. Rypl, et M. Vořechovský. Using Python for scientific computing : Efficient and flexible evaluation of the statistical characteristics of functions with multivariate random inputs. *Computer Physics Communications*, 184(2) :414–427, February 2013.

TF Coleman et Y Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6 :418–445, 1993.

TF Coleman et Y Li. On the convergence of interior-reflective Newton methods for nonlinear minimization subject to bounds. *Mathematical programming*, 67(1) :189–224, October 1994.

A Crespi et AJ Ijspeert. AmphiBot II : An amphibious snake robot that crawls and swims using a central pattern generator. *Proceedings of the 9th International Conference on Climbing and Walking Robots*, pages 19–27, 2006.

G Csibra. Mirror neurons and action observation. Is simulation involved ? *http ://www.interdisciplines.org/mirror*, pages 1–5, 2005.

KM Cuomo. Circuit implementation of synchronized chaos with applications to communications. *Physical Review Letters*, 1993.

J Danna. *Dynamique de coordination dans la formation de la trace écrite chez l'adulte et l'enfant.* PhD thesis, 2011.

J Danna, Y Wamain, V Kostrubiec, J Tallet, et PG Zanone. Vers une prise en compte de la contrainte liée à l'effecteur dans la dynamique de coordination graphomotrice. *Psychologie Française*, 55(2) :171–180, June 2010.

J Danna, S Athènes, et PG Zanone. Coordination dynamics of elliptic shape drawing : effects of orientation and eccentricity. *Human Movement Science*, 30(4) :698–710, August 2011.

J Decety et M Jeannerod. Mentally simulated movements in virtual reality : does Fitt's law hold in motor imagery ? *Behavioural Brain Research*, 72 :127–134, 1996.

J Decety et F Michel. Comparative analysis of actual and mental movement times in two graphic tasks. *Brain and Cognition*, 11(1) :87–97, 1989.

J Decety, M Jeannerod, M Germain, et J Pastene. Vegetative response during imagined movement is proportional to mental effort. *Behavioural Brain Research*, 42(1) :1–5, January 1991.

F Delcomyn. Neural basis of rhythmic behavior in animals. *Science*, 210(4469) :492–498, 1980.

D Delignières. *L'approche dynamique du comportement moteur*, volume 1. Paris, revues eps edition, 2004.

AP Dempster, NM Laird, et DB Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1977.

JJ Denier et JP Thuring. The guiding of human writing movements. *Biological Cybernetics*, 2(4) :145–148, 1965.

EH Dooijes. Analysis of handwriting movements. *Acta Psychologica*, 54 :99–114, 1983.

GL Drescher. *Made-up minds : a constructivist approach to artificial intelligence.* The MIT Press, 1991.

S Dubey, S Sambaraju, SC Cautha, V Arya, et VS Chakravarthy. A phase dynamic model of systematic error in simple copying tasks. *Biological Cybernetics*, 101(3) :201–13, September 2009.

S Edelman et T Flash. A model of handwriting. *Biological Cybernetics*, 36 :25–36, 1987.

M Eden. Handwriting and pattern recognition. *IEEE Transactions on Information Theory*, 8(2) :160–166, February 1962.

TE Feinberg, DM Roane, et J Ali. Illusory limb movements in anosognosia for hemiplegia. *Journal of Neurology, Neurosurgery, and Psychiatry*, 68(4) :511–3, April 2000.

PW Fink, VK Jirsa, P Foo, et JAS Kelso. Local and global stabilization of coordination by sensory information. *Experimental Brain Research*, 134(1) :9–20, August 2000.

T Flash. *Organizing principles underlying the formation of hand trajectories.* The MIT Press, Cambridge, Massachusetts, 1983.

R Fletcher et MJD Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, (1) :163–168, 1963.

JA Fodor. *The language and thought*. Harvard University Press, 1975.

V Gallese, L Fadiga, L Fogassi, et G Rizzolatti. Action recognition in the premotor cortex. *Brain : a Journal of Neurology*, 119 (Pt 2 :593–609, April 1996.

CR Gallistel et R Gelman. Preverbal and verbal counting and computation. *Cognition*, (44) :43–73, 1992.

G Gangadhar, D Joseph, et VS Chakravarthy. An oscillatory neuromotor model of handwriting generation. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(2) :69–84, June 2007.

JJ Gibson. *The senses considered as perceptual sytems*. Houghton Mifflin, 1966.

E Gilet. *Modélisation Bayésienne d'une boucle perception-action : application à la lecture et l'écriture*. PhD thesis, 2009.

S Grossberg et RW Paine. A neural model of cortico-cerebellar interactions during attentive imitation and predictive learning of sequential handwriting movements. *Neural networks : the official journal of the International Neural Network Society*, 13(8-9) : 999–1046, 2000.

W Guerfali et R Plamondon. A new method for the analysis of simple and complex planar rapid movements. *Journal of Neuroscience Methods*, 82(1) :35–45, 1998.

Y Guiard. On Fitts's and Hooke's laws : Simple harmonic movement in upper-limb cyclical aiming. *Acta Psychologica*, 82(1-3) :139–159, March 1993.

Y Guiard. Fitts' law in the discrete vs. cyclical paradigm. *Human Movement Science*, 16 : 97–131, 1997.

H Haken. *Synergetics. An Introduction. Nonequilibrium Phase Trasitions and Self-organization in Physics, Chemistry, and Biology*. Springer-Verlag, Berlin and New Yortk, 1983.

H Haken, JAS Kelso, et H Bunz. A Theoretical Model of Phase Transitions in Human Hand Movements. *Biological Cybernetics*, 51 :347–356, 1985.

J Hamill. Issues in quantifying variability from a dynamical systems perspective. *Journal of Applied Biomechanics*, pages 407–418, 2000.

J Hamill, BT Bates, et KG Holt. Timing of lower extremity joint actions during treadmill running. *Medicine and science in sports and exercise*, 24(7) :807–13, July 1992.

D Hanselman et B Littlefield. Matlab, the language of technical computing. 1997.

SS Haykin. *Neural networks : a comprehensive foundation*. Prentice Hall PTR, Upper Saddle River, NJ, 2007.

G Hesslow. The current status of the simulation theory of cognition. *Brain research*, 1428 (71-79), 2012.

T Hoinville, P Henaff, et S Delaplace. Etude sur l'intérêt des modèles biologiques de réseaux de neurones pour la synthèse de rythmes locomoteurs adaptatifs. *Journal Européen des Systèmes Automatisés*, 41 :413–435, 2007.

JM Hollerbach. An Oscillation Theory of Handwriting. *Biological Cybernetics*, 156 :139–156, 1981.

J Hu et MK Brown. HMM Based On-Line Handwriting Recognition. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 18(10) :1039–1045, 1996.

JD Hunter. Matplotlib : A 2D graphics environment. *Computing In Science & Engineering*, 9(3) :90–95, 2007.

R Huys, BE Studenka, NL Rheaume, HN Zelaznik, et VK Jirsa. Distinct timing mechanisms produce discrete and continuous movements. *PLoS Computational Biology*, 4(4) : e1000061, 2008.

AJ Ijspeert. Central pattern generators for locomotion control in animals and robots : a review. *Neural networks : the official journal of the International Neural Network Society*, 21(4) :642–53, May 2008.

A Jadbabaie, N Motee, et M Barahona. On the stability of the Kuramoto model of coupled nonlinear oscillators. In *American Control Conference, 2004. Proceedings of the 2004*, volume 5, pages 4296–4301. IEEE, 2004.

M Jeannerod. The representing brain : Neural correlates of motor intention and imagery. *Behavioral and Brain sciences*, 17(2) :187–201, 1994.

M Jeannerod. Neural simulation of action : a unifying mechanism for motor cognition. *Neuroimage*, 14(1) :S103–9, July 2001.

T Jellema, CI Baker, B Wicker, et DI Perrett. Neural representation for the perception of the intentionality of actions. *Brain and cognition*, 44(2) :280–302, November 2000.

AJ Jerri. The Shannon sampling theorem—Its various extensions and applications : A tutorial review. *Proceedings of the IEEE*, 65(11), 1977.

VK Jirsa et JAS Kelso. The excitator as a minimal model for the coordination dynamics of discrete and rhythmic movement generation. *Journal of Motor Behavior*, 37(1) :35–51, January 2005.

SC Johnson. Yacc : Yet another compiler-compiler. Technical report, Bell Laboratories, 1975.

E Jones, T Oliphant, et P Peterson. {SciPy} : Open source scientific tools for {Python}.

KT Kalveram. A neural oscillator model learning given trajectories, or how an" alloimitation algorithm "can be implemented into a motor controller. *Motor Control and Human Skill : A Multi-disciplinary Perspective*, pages 127–140, 1998.

M Kawato, Y Maeda, Y Uno, et R Suzuki. Trajectory formation of arm movement by cascade neural network model based on minimum torque-change criterion. *Biological Cybernetics*, 288 :275–288, 1990.

JAS Kelso. Phase transitions and critical behavior in human bimanual coordination. *Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, (15), 1984.

JAS Kelso, KG Holt, P Rubin, et PN Kugler. Patterns of human interlimb coordination emerge from the properties of non-linear, limit cycle oscillatory processes : theory and data. *Journal of Motor Behavior*, 13(4) :226–261, 1981.

C Keysers, B Wicker, V Gazzola, J Anton, L Fogassi, et Vittorio Gallese. A touching sight : SII/PV activation during the observation and experience of touch. *Neuron*, 42 (2) :335–46, April 2004.

M Kherallah, L Haddad, AM Alimi, et A Mitiche. On-line handwritten digit recognition based on trajectory and velocity modeling. *Pattern Recognition Letters*, 29(5) :580–594, April 2008.

G Knoblich et E Seigerschmidt. Authorship effects in the prediction of handwriting strokes : Evidence for action simulation during action perception. *The Quarterly Journal of Experimental Psychology*, (3) :1027–1046, 2002.

DJ Korteweg. Les horloges sympathiques de Huygens. *Archives Neerlandaises, Serie II. Tome XI*, pages 273–295, 1906.

SM Kosslyn, TM Ball, et BJ Reiser. Visual images preserve metric spatial information : evidence from studies of image scanning. *Journal of Experimental Psychology.*, 4(1) : 47–60, February 1978.

SM Kosslyn, NM Alpert, WL Thompson, V Maljkovic, SB Weise, Christopher F. Chabris, Sania E. Hamilton, Scott L. Rauch, et Ferdinando S. Buonanno. Visual Mental Imagery Activates Topographically Organized Visual Cortex : PET Investigations. *Journal of Cognitive Neuroscience*, 5(3) :263–287, July 1993.

V Kostrubiec, R Soppelsa, JM Albaret, et PG Zanone. Facilitation of non-preferred coordination patterns during the transition from discrete to continuous movements. *Motor Control*, 15(4) :456–80, October 2011.

E Kunesch, F Binkofski, et HJ Freund. Invariant temporal characteristics of manipulative hand movements. *Experimental Brain Research*, 78(3) :539–546, 1989.

MJ Kurz et N Stergiou. Effect of normalization and phase angle calculations on continuous relative phase. *Journal of Biomechanics*, 2002.

F Lacquaniti, C Terzuolo, et P Viviani. The law relating the kinematic and figural aspects of drawing movements. *Acta Psychologica*, 54(1-3) :115–130, October 1983.

TK Landauer. Rate of implicit speech. *Perceptual and Motor Skills*, page 646, 1962.

KS Lashley. The accuracy of movement in the absence of excitation from the moving organ. *American Journal of Physiology*, 43 :169–194, 1917.

KS Lashley. An examination of the "continuity theory" as applied to discriminative learning. *The Journal of General Psychology*, 26(2) :241–265, 1942.

ME Lesk et E Schmidt. Lex : A lexical analyzer generator. Technical report, Bell Laboratories, 1975.

Z Lin et L Wan. Style-preserving English handwriting synthesis. *Pattern Recognition*, 40 (7) :2097–2109, July 2007.

M Longcamp, T Tanskanen, et R Hari. The imprint of action : motor cortex involvement in visual perception of handwritten letters. *NeuroImage*, 33(2) :681–8, November 2006.

M Longcamp, Y Hlushchuk, et R Hari. What differs in visual recognition of handwritten vs. printed letters ? An fMRI study. *Human brain mapping*, 32(8) :1250–9, August 2011.

DG Luenberger et Y Ye. *Linear and Nonlinear Programming*, volume 9. Springer, 2008.

JS MacDonald. *Experimental studies of handwriting signals.* Massachusetts Institute of Technology, Research Laboratory of Electronics, 1966.

MATLAB. *version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts, 2010.

I McClay et K Manal. Coupling parameters in runners with normal and excessive pronation. *Journal of Applied Biomechanics*, 13(1) :109–124, 1997.

I G Meister, T Krings, H Foltys, B Boroojerdi, M Müller, R Töpper, et A Thron. Playing piano in the mind–an fMRI study on music imagery and performance in pianists. *Brain research. Cognitive brain research*, 19(3) :219–28, May 2004.

M Merleau-Ponty. *La structure du comportement.* Presses Universitaires de France, Paris, 1942.

P Mermelstein et M Eden. Experiments on computer recognition of connected handwritten words. *Information and Control*, 7(2) :255–270, June 1964.

RE Mirollo et SH Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM Journal on Applied Mathematics*, 50(6) :1645–1662, 1990.

P Morasso et FAM Ivaldi. Trajectory formation and handwriting : a computational model. *Biological cybernetics*, 45(2) :131–142, 1982.

A Murata, L Fadiga, L Fogassi, V Gallese, V Raos, et G Rizzolatti. Object representation in the ventral premotor cortex (area F5) of the monkey. *Journal of Neurophysiology*, 78 (4) :2226–30, October 1997.

E Nakano, H Imamizu, R Osu, Y Uno, H Gomi, T Yoshioka, et M Kawato. Quantitative examinations of internal representations for arm trajectory planning : minimum commanded torque change model. *Journal of Neurophysiology*, 81(5) :2140–55, May 1999.

E Nel, JA du Preez, et BM Herbst. Estimating the pen trajectories of multi-path static scripts using hidden Markov models. *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, 1 :41–45, 2005.

R Olfati-Saber. Ultrafast consensus in small-world networks. *Proceedings of the 2005, American Control Conference,*, pages 2371–2378, 2005.

JK O'Regan et A Noë. A sensorimotor account of vision and visual consciousness. *The Behavioral and Brain Sciences*, 24(5) :939–73, October 2001.

RW Paine, S Grossberg, et AWA Van Gemmert. A quantitative evaluation of the AVITEWRITE model of handwriting learning. *Human movement science*, 23(6) :837–60, December 2004.

PF Panter. *Modulation, noise, and spectral analysis.* McGraw-Hil, New-York, 1965.

D Perdikis, R Huys, et VK Jirsa. Complex processes from dynamical architectures with time-scale hierarchy. *PloS one*, 6(2) :e16589, 2011.

FS Perotto. *Un mécanisme constructiviste d'apprentissage automatique, d'anticipations pour des agents artificiels situés*. PhD thesis, 2010.

BT Peters, JM Haddad, BC Heiderscheit, REA Van Emmerik, et J Hamill. Limitations in the use and interpretation of continuous relative phase. *Journal of Biomechanics*, 36 (2) :271–4, February 2003.

J Piaget. *Construction of reality in the child*. Psychology Press, 1954.

J Piaget. *Biology and knowledge : An essay on the relations between organic regulations and cognitive processes.* U. Chicago Press, 1967.

J Piaget et E Duckworth. Genetic epistemology. *American Behavioral Scientist*, 13(3) : 459–480, 1970.

M Piattelli-Palmarini. *Language and learning : the debate between Jean Piaget and Noam Chomsky*. Cambridge Univ Press, 1980.

MA Pitt et IJ Myung. When a good fit can be bad. *Trends in Cognitive Sciences*, 6(10) : 421–425, 2002.

R Plamondon. The generation of rapid human movements. Part II : Quadratic and power laws. *Laboratoire Scribens, Ecole Polytechnique edition*, 1993.

R Plamondon. A kinematic theory of rapid human movements : Part III. Kinetic outcomes. *Biological Cybernetics*, 78(2) :133–145, 1998.

R Plamondon. On-Line and Off-Line Handwriting Recognition : A Comprehensive Survey. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 22(1), 2000.

R Plamondon et W Guerfali. The generation of rapid human movements. Part I, A delta-lognormal law. *Biological Cybernetics*, 78(93/94) :119–132, 1993.

R Plamondon et W Guerfali. The generation of handwriting with delta-lognormal synergies. *Biological Cybernetics*, 132 :119–132, 1998.

R Plamondon et F Lamarche. Modelization of handwriting : A system approach. *Graphonomics : Contemporary Research in Handwriting*, pages 169–183, 1986.

R Plamondon, a M Alimi, P Yergeau, et F Leclerc. Modelling velocity profiles of rapid movements : a comparative study. *Biological Cybernetics*, 69(2) :119–28, January 1993.

R Plamondon, C Feng, et A Woch. A kinematic theory of rapid human movement. Part IV : a formal mathematical proof and new insights. *Biological Cybernetics*, 89(2) :126–138, 2003.

T Plötz et GA Fink. Markov models for offline handwriting recognition : a survey. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(4) :269–298, October 2009.

MJD Powell. An iterative method for finding stationary values of a function of several variables. *The Computer Journal*, 5(2) :147–151, 1962.

JF Prather, S Peters, S Nowicki, et R Mooney. Precise auditory-vocal mirroring in neurons for learned vocal communication. *Nature*, 451(7176) :305–10, January 2008.

J Proust. Pour une théorie 'motrice' de la simulation. *Psychologie Française*, pages 295–306, 2000.

JC Quinton. *Coordination implicite d'interactions sensorimotrices comme fondement de la cognition.* PhD thesis, Université de Toulouse, 2008.

G Rizzolatti, L Fadiga, V Gallese, et L Fogassi. Premotor cortex and the recognition of motor actions. *Brain research. Cognitive brain research*, 3(2) :131–41, March 1996.

M Rosenblum et J Kurths. *Nonlinear analysis of physiological data.* Number 1996. Springer, 1998.

M Rosenblum et A Pikovsky. Controlling Synchronization in an Ensemble of Globally Coupled Oscillators. *Physical Review Letters*, 92(11) :1–4, March 2004.

M Rosenblum, A Pikovsky, et J Kurths. Phase synchronization : from theory to data analysis. *Handbook of Biological Physics*, 4(August 2001) :279–321, 2001.

S Russell et P Norvig. Artificial Intelligence, 2nd Ed. *Artificial Intelligence*, 82, 2006.

I Sallagoïty, S Athènes, PG Zanone, et JM Albaret. Stability of coordination patterns in handwriting : effects of speed and hand. *Motor Control*, 8(4) :405–21, October 2004.

R Sassoon. *The art and science of handwriting.* Intellect Books, 1993.

S Schaal, D Sternad, R Osu, et M Kawato. Rhythmic arm movement is not discrete. *Nature neuroscience*, 7(10) :1137–1145, 2004.

LRB Schomaker. *Simulation and recognition of handwriting movements : a vertical approach to modeling human motor behavior.* PhD thesis, Nijmengen University, Netherlands, 1991.

G Schöner. A dynamic theory of coordination of discrete movement. *Biological Cybernetics*, 63(4) :257–270, 1990.

G Schöner et E Thelen. Using dynamic field theory to rethink infant habituation. *Psychological Review*, 113(2) :273–99, April 2006.

M Schürmann, T Raij, N Fujiki, et R Hari. Mind's ear in a musician : where and when in the brain. *NeuroImage*, 16(2) :434–40, June 2002.

RN Shepard et J Metzler. Mental rotation of three-dimensional objects. *Science*, 171 (3972) :701–703, 1971.

Y Singer et N Tishby. Dynamical encoding of cursive handwriting. *Biological Cybernetics*, 71(3) :227–37, January 1994.

ES Spelke et K Breinlinger. Origins of knowledge. *Psychological Review*, 99(4) :605–632, 1992.

RMC Spencer, HN Zelaznik, J Diedrichsen, et RB Ivry. Disrupted timing of discontinuous but not continuous movements by cerebellar lesions. *Science*, 300(5624) :1437–9, May 2003.

O Stettiner et D Chazan. A statistical parametric model for recognition and synthesis of handwriting. In *Proceedings of the 12th IAPR International.*, volume 2, pages 34–38. IEEE, 1994.

SH Strogatz. From Kuramoto to Crawford : exploring the onset of synchronization in populations of coupled oscillators. *Physica D*, 143 :1–20, 2000.

SH Strogatz. Exploring complex networks. *Nature*, 410(6825) :268–76, March 2001.

SH Strogatz. *Sync : The emerging science of spontaneous order*. Penguin sc edition, 2004.

HL Teulings. Handwriting movement control. *Handbook of Perception and Action*, 2 : 561–613, 1996.

HL Teulings et FJ Maarse. Digital recording and processing of handwriting movements. *Human Movement Science*, 3(1) :193–217, 1984.

HL Teulings et AJWM Thomassen. Computer-Aided Analysis of Handwriting Movements. *Visible Language*, 13(3) :218–231, 1979.

AJWM Thomassen et LRB Schomaker. Between-letter context effects in handwriting trajectories. *Graphonomics : Contemporary Research in Handwriting*, pages 253–272, 1986.

K Tsang, RE Mirollo, SH Strogatz, et K Wiesenfeld. Dynamics of a globally coupled oscillator array. *Physica D*, 48(1) :102–112, February 1991.

B Tuller et JAS Kelso. Environmentally-specified patterns of movement coordination in normal and split-brain subjects. *Experimental Brain Research*, pages 306–316, 1989.

S Uithol et WFG Haselager. When do we stop calling them mirror neurons. *Proceedings of the 30th annual meeting of the Cognitive Science Society.*, 2008.

Y Uno, M Kawato, et R Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biological cybernetics*, 101 :89–101, 1989.

J van der Gon, JJ Denier, JP Thuring, et J Strackee. A handwriting simulator. *Physics in medicine and Biology*, 6(3) :407, 1962.

AM van Mourik et PJ Beek. Discrete and cyclical movements : unified dynamics or separate control ? *Acta psychologica*, 117(2) :121–38, October 2004.

N Vincent, A Seropian, et G Stamon. Synthesis for handwriting analysis. *Pattern Recognition Letters*, 26(3) :267–275, February 2005.

A Vinciarelli. A survey on off-line cursive word recognition. *Pattern Recognition*, 35 : 1433–1446, 2002.

P Viviani et C Terzuolo. Trajectory determines movement dynamics. *Neuroscience*, 7(2) : 431–437, 1982.

Y Wada et M Kawato. A theory for cursive handwriting based on the minimization principle. *Biological Cybernetics*, 73(1) :3–13, June 1995.

A Wald. The fitting of straight lines if both variables are subject to error. *The Annals of Mathematical Statistics*, 11(3) :284–300, 1940.

Jue Wang, C Wu, YQ Xu, HY Shum, et L Ji. Learning-based cursive handwriting synthesis. *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 157–162, 2002.

JK Witt. Action's Effect on Perception. *Current Directions in Psychological Science*, 20 (3) :201–206, May 2011.

JK Witt et JR Brockmole. Action alters object identification : wielding a gun increases the bias to see guns. *Journal of Experimental Psychology : Human Perception and Performance*, 38(5) :1159–67, October 2012.

JI Yamanishi, M Kawato, et R Suzuki. Two coupled oscillators as a model for the coordinated finger tapping by both hands. *Biological Cybernetics*, 225 :219–225, 1980.

M Yasuhara. Experimental studies of handwriting process. *Japanese journal of medical electronics and biological engineering*, 12(2) :69–80, April 1974.

PG Zanone et JAS Kelso. Evolution of behavioral attractors with learning : Nonequilibrium phase-transitions. *Journal of Experimental Psychology : Human Perception and Performance*, 18(2) :403–421, 1992.

Abstract

In this thesis, we propose an oscillatory model of handwritten production, called POMH, where handwriting is considered as the combination of movement of two oscillators. It derives from the classical Hollerbach model which has been simplified and symmetrized, allowing for a fast method of extraction of its parameters from recorded strokes. We compare this algorithm with a Newton-based optimization method in order to prove its efficiency. We show that POMH is as good as the widespread Edelman-Flash model, in terms of accuracy at reproducing simple shapes. Then, it is successfully applied to more complicated handwritten strokes such as entire sentences or Chinese characters.

In a second time, we aim at explaining the correlations between these two oscillators. In that purpose, we study their relative phase, based on the dynamical approach of motor coordination. We try to answer these two questions : (i) Is there, like it is the case in the drawing of ellipses, preferred coordinations in the handwriting movement ? (ii) What are the links between handwriting speed and the geometry of the produced trace ? In order to answer the first question, two experiments have been conducted. The first aims at validating the continuous relative phase computation on real handwriting; the second studies the possible existence of preferred coordination patterns in the handwriting task. To begin to answer the second question, we propose two hypotheses and two experiments that would allow to validate them.

In a last effort, based on the interactivist theory, we propose a simplified model of off-line handwriting recognition, using the idea that reading and handwriting share a lot of their mechanisms. We first begin by a search, in the literature, for evidence of the links between perception and action for all modalities, particularly regarding handwriting. We then propose two algorithms implementing handwriting perception as an inner simulation. The first uses POMH as an internal representation of characters; the second is based on the concept of "temporalized images" and gives better results.

Keywords: handwritting, graphomotricity, oscillators, motor coordination, interactivism, off-line recognition

Résumé

Dans ce travail de thèse, nous proposons un modèle oscillatoire de la production de traces écrites, appelé POMH, où l'écriture est vue comme la composition du mouvement de deux oscillateurs orthogonaux du plan. Il s'agit d'une extension du modèle classique d'Hollerbach, qui a été simplifié et rendu symétrique, permettant ainsi une méthode très rapide d'extraction de ses paramètres à partir de traces réelles enregistrées. Nous comparons cet algorithme avec une méthode d'optimisation usuelle afin de montrer son efficacité. Nous montrons que POMH permet de rivaliser avec un modèle répandu de la génération de trace, en terme de précision de reproduction de formes simples. Il est ensuite confronté, avec succès, à des traces plus complexes telles que des phrases entières ou des caractères chinois.

Dans un second temps, nous cherchons à interroger les liens entre ces deux oscillateurs. Pour cela nous étudions leur phase relative, nous appuyant sur une approche dynamique de la coordination motrice. Nous tentons d'apporter des éléments de réponses à deux questions : (i) Y-a-t-il, comme pour le tracé d'ellipses, des coordinations préférentielles lorsque l'on écrit ? (ii) Quels sont les liens entre vitesse d'écriture et géométrie de la trace écrite ? Pour tenter de répondre à la première question, deux expérimentations sont faites. La première cherche à valider le calcul de la phase relative continue sur de l'écriture réelle. La seconde étudie l'existence éventuelle de patrons de coordinations (représentés par la phase relative) préférentiels pour la tâche d'écriture chez l'adulte. Pour tenter de répondre à la deuxième question, nous formulons deux hypothèses concernant l'évolution du paysage des patrons de coordination préférentiels avec la vitesse d'écriture. Nous proposons alors deux protocoles expérimentaux (non réalisés) qui permettraient de les valider.

Dans un dernier temps, nous inscrivant dans un cadre interactiviste, nous proposons un modèle simplifié de reconnaissance de caractères, basé sur l'idée que lecture et écriture partagent un grand nombre de leurs mécanismes. Nous commençons par une recherche dans la littérature des preuves de ces liens entre perception et action pour toutes les modalités, et en particulier pour l'écriture cursive. Nous proposons ensuite deux algorithmes implémentant la perception de l'écriture comme une simulation intériorisée. Le premier utilise POMH comme représentation interne des caractères ; le second est basé sur le concept d'"images temporalisées" et donne de meilleurs résultats.

Mot-Clés : écriture, graphomotricité, oscillateurs, coordination motrice, interactivisme, reconnaissance d'écriture off-line