Université Paris Sud

Thèse

Pour obtenir le titre de

Docteur de l'Université Paris XI Orsay
Spécialité : Informatique

Présentée par

Bruno Bossard

Conception d'un système de reconnaissance automatique De gestes bimanuels : application à la réalité virtuelle et à la langue des signes

---

Soutenue publiquement le **9 juin 2006** *devant le jury composé de*

Mme Annelies Braffort	co-encadrante
Mme Sabine Coquillard	rapporteur
M Patrice Dalle	rapporteur
Mme Sylvie Gibet	examinatrice
Mme Michèle Jardino	directrice

Préparée au **LIMSI-CNRS**

Table des matières

Remerciements

Résumé

Introduction

Diverses modalités comme la parole, l'audio, la vision sont utilisées en communication homme-machine afin de permettre à un utilisateur de communiquer ou percevoir des informations. La modalité gestuelle, principalement pour des raisons techniques, a été peu ou pas exploitée. Depuis quelques années, les développement matériels et logiciels ont permis au geste de devenir plus facilement exploitable et cela jusqu'au grand public qui peut ainsi utiliser cette modalité dans certains jeux vidéos.
Dans cette thèse, nous nous intéressons uniquement à la modalité gestuelle et nous ne parlerons pas des autres moyens de communication existants.

Les gestes peuvent être utilisés de manières fort différentes : perception tactile du monde qui nous entoure, manipulation ou modification d'objets et expressions d'informations. Pour chacun de ces cas, on s'aperçoit que l'utilisation d'une seule et unique main n'est pas toujours suffisante ou satisfaisante et que, souvent, les deux mains sont utilisées. Par exemple, pour écrire, une main tient le papier tandis que l'autre effectue le tracé. Lors d'une conversation, pour indiquer une taille, on va utiliser l'écartement entre les deux mains.
Cette utilisation des deux mains peut se réaliser de manière synchrone ou non et permet, éventuellement, d'exprimer des informations de type spatial et/ou temporel. C'est à cet aspect du geste que nous nous intéressons et nous allons l'explorer au travers de cette thèse dans le but de pouvoir l'interpréter de manière automatique.

La réalisation de gestes avec les deux mains se rencontre dans quasiment tous les domaines où la modalité gestuelle intervient. Nous nous intéressons ici à seulement deux d'entre eux : la langue des signes et la réalité virtuelle. Ce choix est dû à plusieurs raisons : tout d'abord ce sont deux domaines de recherche dans le laboratoire où cette thèse a été effectuée ; ensuite la langue des signes est un des principaux sujets de recherche dans le domaine de la reconnaissance de gestes et elle possède des mécanismes d'expression d'information à l'aide des deux mains forts intéressants et qui n'ont quasiment pas été étudiés en informatique ; enfin la réalité virtuelle est un domaine dans lequel le geste peut s'avérer un média fort efficace et qui reste ouvert à de nouvelles prospectives.
Effectuer un travail de recherche sur les gestes dans ces deux domaines nécessite des connaissances fort diverses. C'est donc une véritable thèse pluridisciplinaire qui est détaillée dans ce document, on y abordera des aspects linguistiques, techniques et logiciels. Nous allons maintenant présenter l'organisation des différents éléments aux travers desquels nous allons présenter ces différents aspects ainsi que le travail que nous avons réalisé.

Ce manuscrit de thèse est décomposé en trois parties. Dans la première partie, nous allons

introduire le contexte de ce travail. Tout d'abord le premier chapitre présente certaines propriétés de la langue des signes, puis il explique la problématique des gestes à deux mains au travers de la langue des signes. Ensuite, le deuxième chapitre va détailler le processus de la reconnaissance d'une forme et donner l'état de l'art des principaux travaux menés dans le domaine de la reconnaissance automatique de gestes.

La deuxième partie est principalement dédiée à nos contributions dans le domaine de la modalité gestuelle. Le chapitre 3 contient une description globale de l'architecture que nous proposons pour effectuer l'interprétation de gestes à deux mains. Dans le quatrième chapitre, nous commençons par présenter la réalité virtuelle ainsi que les gestes que l'on y rencontre. Suite à cette présentation nous montrons que la langue des signes et la réalité virtuelle ne sont pas deux domaines si éloignés l'un de l'autre que l'on pourrait le penser, et nous proposons d'effectuer un transfert de certains gestes de la langue des signes vers la réalité virtuelle.

La troisième et dernière partie est consacrée aux expérimentations que nous avons menées. Dans le chapitre 5 on trouve la description de l'implémentation de notre architecture dans le cadre de la réalité virtuelle. Nous y donnons également notre étude faite sur un vocabulaire élémentaire de gestes de la réalité virtuelle ainsi que l'expérimentation de notre transfert de gestes de la langue des signes en réalité virtuelle. Le sixième chapitre concerne l'expérimentation de notre architecture dans le domaine de la langue des signes. Seule une portion de l'architecture a pu être implémentée et testée, c'est pourquoi la fin du chapitre est consacrée à des aspects de l'architecture qui nous paraissent importants et que nous aurions aimé pouvoir expérimenter.

Enfin dans le dernier chapitre, nous donnerons la conclusion de ce manuscrit en effectuant un bilan sur les points nouveaux que nous avons proposés ici.

Première partie

Problématiques et état de l'art

Chapitre 1

La problématique des gestes bimanuels

Ce chapitre a pour objectif d'introduire la problématique du sujet de cette thèse, d'exposer quels sont les domaines d'application et de préciser quelles sont les limites que nous nous sommes imposées vis à vis du traitement de la problématique.
Dans la première section, la thèse sera positionnée par rapport aux différentes fonctionnalités et différents types de gestes qui existent. Ce positionnement s'effectue en particulier sur la langue des signes dont quelques caractéristiques seront décrites dans la section 2. Ce court exposé de la langue des signes permettra de détailler dans la troisième section, la nature des différentes relations qui peuvent survenir entre les deux mains. Enfin une synthèse sera effectuée dans la dernière section.

1.1 Les gestes bimanuels : une utilisation omniprésente

Les gestes bimanuels sont des gestes que l'on rencontre de manière courante quelque soit le type de contexte dans lequel nos mains interviennent. Cela peut être lors de gestes "intuitifs" (par exemple, préhension d'un objet volumineux à l'aide des deux mains) ; mais aussi lorsque l'on veut intervenir sur un objet (une main tient l'objet pendant que l'autre effectue, à l'aide d'un outil, une action - écriture, sculpture, assemblage, etc) ; ou encore quand les gestes servent à exprimer des informations (par exemple, on écarte les deux mains pour indiquer une distance).
Les gestes utilisés dans ces diverses situations, qu'ils soient bimanuels ou pas, font intervenir différentes fonctionnalités : la fonction ***épistémique*** qui sert à la perception de l'environnement, la fonction ***ergotique*** qui permet d'effectuer des actions sur l'environnement et la fonction ***sémiotique*** qui permet d'exprimer des informations [Cadoz, 1994].

Ces fonctionnalités ne sont pas exclusives. En particulier, les fonctionnalités épistémiques et ergotiques sont souvent considérées ensembles et constituent ce que l'on appelle le niveau ***sensorimoteur***. Ces fonctionnalités ont déjà été bien étudiées et continuent à l'être dans de nombreux domaines : informatique, psychologie, physiologie, robotique ... L'utilisation des gestes bimanuels est particulièrement pertinente dans les cas où l'on effectue une action sur un objet. On a alors une main qui permet de tenir et orienter l'objet tandis que l'autre effectue l'action, c'est le cas par exemple lorsque l'on écrit sur une feuille de papier.
Dans le domaine de la communication homme-machine, plusieurs travaux ont permis d'aboutir à des interfaces permettant d'interagir avec les deux mains dans divers types d'applications. Par exemple, dans [Bier et al., 1994], les auteurs proposent une interface où chaque main tient une souris. La main gauche permet de choisir l'outil et la zone d'application de cet outil, tandis que la main droite effectue l'application de cet outil. On retrouve donc un peu dans cette interface le même rôle que l'on a lorsque la main gauche tient une feuille et que la main droite écrit dessus. En utilisant ainsi à bon escient l'aspect sensorimoteur des gestes bimanuels, on obtient de meilleurs résultats en précision et en temps d'exécution d'une tâche dans le domaine des interfaces homme-machine [Leganchuk et al., 1998].
Cette thèse aborde peu l'aspect sensorimoteur des gestes et se focalise essentiellement sur les gestes ayant des fonctions sémiotiques.

Les gestes bimanuels permettent d'exprimer des informations spatiales et temporelles du fait qu'il est possible d'avoir une différence de placement et de mouvement entre chacune des mains, et cela est utilisé pour chacun des types de geste servant à véhiculer de l'information. Ainsi on retrouve à tout niveau du continuum de Kendon[1] [McNeill, 1992, Kendon, 1988] l'utilisation des deux mains pour exprimer des informations spatiales et/ou temporelles. Par exemple, au niveau gesticulatoire, on peut aider à la manoeuvre d'une voiture en indiquant la distance restante à l'aide de l'écartement entre les deux mains. Les gestes bimanuels coverbaux, eux, accompagnent souvent le discours oral pour indiquer des tailles, des dispositions spatiales d'objets, etc. En mimique, les

[1] Au début des années 80, Adam Kendon introduit une classification des gestes selon un critère de complexité linguistique et d'expressivité. Cette classification s'effectue selon un axe continu où l'on retrouve ordonnés les différents domaines de communication gestuelle. Ainsi, tout en bas de l'echelle se trouvent les ***gesticulations*** (exemple ?) et les ***gestes coverbaux***, puis on trouve les ***emblèmes***, les ***mimiques***, etc, jusqu'à arriver à la langue des signes qui constitue la forme la plus évoluée de la communication gestuelle. Du fait de la continuité entre chaque domaine, on peut dire que n'importe quel geste de nature sémiotique appartient forcément à un de ces domaines.

deux mains peuvent être utilisées, par exemple, pour décrire des formes. Mais c'est en langue des signes que tout le potentiel des relations entre les deux mains est pleinement exploité ; on va s'en servir pour décrire des scènes, des actions, des formes, des similarités, des relations temporelles, etc.
C'est principalement dans le domaine de la linguistique que des recherches sur ce thème ont été effectuées. Dans le domaine informatique, pour l'instant, encore peu de travaux se sont intéressés à l'exploitation des informations contenues dans ces gestes à deux mains. Cela est dû soit au fait que les autres travaux s'intéressent à d'autres points difficiles de l'interprétation des gestes, soit au fait que les systèmes considèrent les mains comme une seule entité et ne sont donc pas opérationnels sur ce type de problème.

C'est dans ce cadre de la recherche d'informations dans les gestes bimanuels que cette thèse s'inscrit, et plus particulièrement sur l'exploitation des relations spatiales entre les mains. Deux domaines ont été choisis comme champs d'application à ce travail : la langue des signes et la réalité virtuelle. Ces deux domaines, qui peuvent paraître totalement différents, présentent toutefois des points communs. En effet, dans les deux cas, on travaille dans un espace en trois dimensions dans lequel on modélise une scène, et les gestes que l'on effectue se réfèrent à cette scène. L'idée est donc de s'inspirer de certaines interactions gestuelles de la langue des signes pour pouvoir : faciliter certaines tâches que l'on peut avoir à faire en réalité virtuelle ; profiter des techniques de reconnaissance de la langue des signes pour enrichir les interactions gestuelles de la réalité virtuelle. C'est pourquoi, dans ce travail, les deux domaines se trouvent entrelacés dans la description des différents aspects de cette recherche.

La section suivante se focalise plus particulièrement sur la langue des signes (plus exactement la langue des signes française). Elle expose les propriétés permettant de préciser les différentes problématiques que l'on peut rencontrer en reconnaissance des gestes. Toujours dans le cadre de la langue des signes, les divers cas d'interactions entre les deux mains seront explicités, ce qui permettra de mettre en évidence les difficultés à prendre en compte pour interpréter les gestes bimanuels.

1.2 Quelques caractéristiques de la langue des signes

La langue des signes est une langue pratiquée par la communauté sourde comme moyen principal de communication. Elle utilise la modalité visuelle et fait intervenir simultanément plusieurs parties du corps : les mains, les bras, le visage (expression faciale et regard) et le buste. De part ces aspects, c'est une langue qui est radicalement différente d'une langue orale. Alors qu'une langue orale est un moyen de communication mono canal, la langue des signes fait intervenir plusieurs canaux d'informations qui n'ont pas forcément un débit simultané et constant. Par exemple, le regard peut être dirigé vers l'interlocuteur avec des coups d'œil plus ou moins longs vers les mains ou des zones de l'espace, tandis que les mains elles-mêmes vont réaliser des gestes qui ne sont pas obligatoirement simultanés.
Tous ces différents canaux sont des vecteurs d'information pertinents pour l'interprétation d'une phrase de la langue des signes. Toutefois, on ne peut pas aborder toutes les composantes de la langue des signes simultanément, dans le cadre d'un travail sur le traitement automatique de cette langue. Tout d'abord cela nécessite un travail de longue haleine avec des linguistes spécialistes de

la langue. Il faut alors procéder étape par étape, commencer d'abord avec des énoncés très simples avec des signes utilisant une seule main [Braffort, 1996b]. Puis ajouter au fur et à mesure différents éléments permettant de traiter des énoncés de plus en plus complexes. D'autre part, on est limité par les techniques d'acquisition qui ne permettent pas d'effectuer, de manière fiable et pratique, la capture de certaines informations comme celles véhiculées par le visage.
C'est pourquoi, tous les aspects de la langue des signes ne sont pas abordés ici. Ce travail se focalise sur la reconnaissance d'énoncés utilisant des signes bimanuels et une exploitation plus poussée de l'espace. Seuls les bras et les mains sont traités dans cette thèse, mais en gardant à l'esprit la possibilité d'intégrer ultérieurement d'autres parties du corps. Les sections suivantes décrivent donc plus particulièrement certains types d'énoncés et structurations propres aux gestes manuels qui constituent tout de même le principal vecteur d'information.

1.2.1 L'espace de narration

La manière de structurer les phrases suit une tout autre logique que celle utilisée par les langues orales. Pour ces dernières, les phrases suivent des schémas syntaxiques bien précis qui se déroulent de manière linéaire. Comme par exemple, en français, une phrase du type *<sujet> <verbe> <complément>*. En langue des signes, la syntaxe suit des règles bien moins rigides pour ce qui est de l'ordre des signes et obéit à des règles de nature spatio-temporelle. En effet, la personne qui produit l'énoncé, et que l'on appelle ***signeur***, va utiliser l'espace autour d'elle pour structurer sa phrase.

Le signeur va construire autour de lui ce que l'on appelle la ***scène de narration***. Pour cela, il va énoncer et placer les différentes entités (actants, objets, etc) de la phrase dans cette scène en trois dimensions. Ensuite, c'est l'emplacement de l'entité qui va permettre de faire référence à celle-ci pour exprimer une action, un qualificatif ou une relation la concernant. Les entités peuvent être de nature aussi bien abstraite que concrète.

FIG. 1.1 – Extraits d'une vidéo où est posée la question "Qui a fait le film passant le 26 fév. à 9h30 au cinéma Utopia de Toulouse ?" (IRIT Toulouse)

Dans la figure 1.1, on peut voir un exemple d'utilisation de cette scène de narration. Dans la

première partie de la phrase (images 1 à 6), la personne place le contexte (lieu, date, etc) dans la scène. Elle désigne un emplacement à sa gauche (image 1) puis produit à cet endroit le signe [TOULOUSE][2] (image2). Ensuite, elle affine le lieu en indiquant que c'est un cinéma et en donnant son nom (images 3 et 4). A partir de là, toutes les autres informations qui concernent le contexte vont être reliées à ce lieu grâce au regard qui va être pointé sur l'emplacement donné au cinéma dans la scène de narration et par le fait que les signes vont être effectués au niveau de cet emplacement. Ainsi, dans l'image 5, on peut voir le signe [FILM] qui nous indique qu'on va s'intéresser à une séance de ce cinéma. Puis, toute une série de signe va permettre de préciser à quel moment passe ce film (l'image 6 correspond au "2" du "26 février").
Une fois le contexte posé, on va passer à la question elle même. Dans l'image 7, la signeuse place à sa droite une personne (noter que le regard n'est plus a gauche, mais sur ce nouvel emplacement). Aux images 8 et 9, elle revient à l'emplacement du cinéma (nouveau changement du regard), pour effectuer le signe [FAIRE] suivi de [FILM]. Cela permet d'indiquer que ce film est celui dont on a donné le contexte précédemment (à savoir qu'il passe au cinéma Utopia de Toulouse à 9h30 le 26 février). Enfin, elle effectue **entre** l'emplacement de la personne et l'emplacement du cinéma/film le signe [QUI] qui permet de relier les deux entités par une question.
Il faut bien retenir que rien n'oblige la signeuse à énoncer les entités précisément dans cet ordre ; ce qui importe, c'est d'utiliser le même emplacement pour toutes les informations concernant le film et d'effectuer la question sur l'axe reliant les deux entités de la scène. Une fois que l'on a donné un emplacement à une entité, celui ci va perdurer tout le long du discours tant que le signeur n'indique pas un changement du contexte.

Outre le fait que l'espace de narration est utilisé pour structurer la phrase, il peut servir bien évidemment à indiquer la disposition spatiale des entités de la scène (tel objet est au dessus de tel autre), mais aussi à indiquer des informations temporelles. Le signeur peut utiliser un axe temporel qui part d'un point situé derrière lui (où se trouve le passé), passe à son niveau (temps présent) et se poursuit vers l'avant (où se situe le futur). Il va alors disposer les entités le long de cet axe pour indiquer l'ordre temporel.
Il peut aussi positionner des événements temporels n'importe où dans la scène de narration, puis relier les différentes entités de cette scène pour exprimer un ordre temporel. Par exemple dans la figure 1.2, le signe [CINEMA] est effectué à droite, tandis que le signe [MAISON] est, lui, réalisé à gauche. Pour exprimer le fait que l'on rentre à la maison **après** avoir été au cinéma, le signe [FINI] est utilisé au niveau du cinéma pour indiquer que cette situation (être au cinéma) fait partie du passé. Ensuite, le signe [RENTRER] est effectué en allant de la droite vers la gauche pour indiquer que l'on rentre du cinéma vers la maison. Le fait qu'on l'on parte d'un fait fini, indique que l'action "de rentrer" a lieu après le cinéma.

L'espace de narration est très important car il constitue la structure et le support de beaucoup d'énoncés en langue des signes. La section suivante se place à un niveau de granularité plus fin, c'est à dire au niveau des signes, qui sont les éléments constitutifs des phrases.

[2] La notation [SIGNE] est celle utilisée pour désigner des signes standards. La définition de cette classe de vocabulaire sera faite un peu plus loin en section 1.2.3.

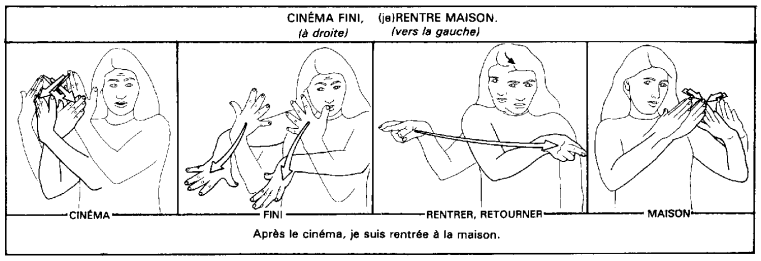

FIG. 1.2 – "Après le cinéma, je suis rentrée à la maison". (dictionnaire IVT).

1.2.2 Compositionnalité des signes

La première décomposition des signes a été proposée par [Stockoe, 1960] et part d'une approche phonologique (héritée des travaux sur les langues orales). Un signe peut alors être décomposé en plusieurs éléments comme les mots d'une langue orale le sont en plusieurs phonèmes. Dans cette première approche, un signe est décomposé en trois éléments : la configuration, le mouvement et l'emplacement. Plus tard, Battison affine le modèle en rajoutant l'orientation [Battison, 1978] qui était auparavant combinée avec l'emplacement. Ces quatre éléments sont communément appelés ***paramètres***.
Bien évidemment, d'autres modèles phonologiques ont vu le jour par la suite pour pouvoir intégrer d'autres propriétés. Par exemple, celui de Liddell et Johnson permet de modéliser en plus les phases de transitions articulatoires entre deux signes [Liddell et Johnson, 1989]. Pour cela, un signe est décomposé en deux parties. La première, qui est appelée ***hold***, correspond à la phase de tenue du signe et intègre les quatre paramètres de Stockoe/Battison plus un paramètre non manuel. La deuxième partie est appelée ***movement*** et modélise la phase d'articulation vers le prochain signe. Plus récemment, on voit apparaître d'autres approches que celle des modèles phonologiques, notamment des approches morphèmiques [Cuxac, 2000, Cuxac, 2003], et articulatoires [Boutet, 2001].

Quelque soit le modèle linguistique que l'on considère, les quatre éléments introduits par Stockoe et Battison sont pertinents d'un point de vue physiologique, mis à part le mouvement qui est considéré de manière globale dans le modèle phonologique (mouvement des mains **et** des doigts). Dans ce travail, c'est l'approche morphèmique qui a été adoptée, c'est à dire que les paramètres, même s'ils sont les "mêmes" que pour le modèle Stockoe/Battison, ne sont pas considérés comme des "phonèmes" mais comme des éléments pouvant être porteurs de valeurs iconiques. Ces quatre paramètres vont maintenant être détaillés :

- La ***configuration***. Ce paramètre correspond à la forme de la main définie par les doigts et la paume (voir exemples, figure 1.3). Il n'est pas forcément figé dans une posture rigide et peut varier comme dans le signe [ARAIGNÉE] (fig. 1.4) où les doigts sont en mouvement.

  Ce paramètre est tout particulièrement porteur d'informations iconiques [Cuxac, 2000]. Ainsi, il est courant d'avoir une analogie entre la forme de la main et la forme de ce que le signe décrit. Par exemple dans le signe [TABLE] (fig. 1.4) la main est plate comme la surface d'une table, dans le signe [ARAIGNÉE] les doigts représentent les pattes de l'araignée.
- L'***orientation***. Elle est définie par deux axes de la main, comme indiqué dans la figure 1.5.

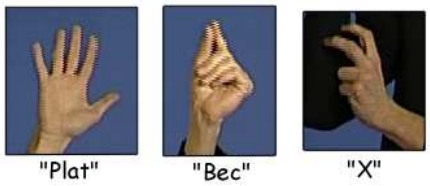

FIG. 1.3 – Exemples de configurations de la main. (LS-COLIN)

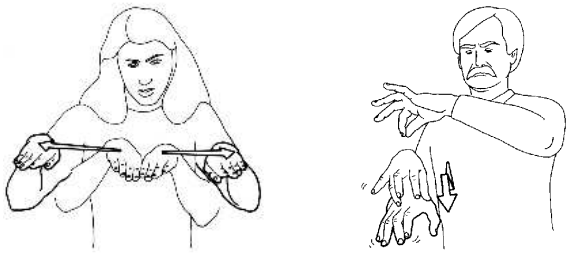

FIG. 1.4 – Les signes [TABLE] et [ARAIGNÉE]. (dictionnaire IVT).

L'orientation est utilisée par le signeur, soit pour donner une bonne vision de la configuration à son interlocuteur, soit pour donner des informations supplémentaires lorsque l'on décrit un objet ou une scène. Par exemple, si l'on décrit la forme d'un objet à l'aide d'un mouvement suivant le contour, la paume de la main va indiquer le côté intérieur tandis que le dos de la main figure lui le côté extérieur. Dans la figure 1.6, la configuration de la main symbolise une personne. L'orientation dans ce cas permet d'indiquer par exemple si la personne est debout (image de gauche), ou allongée (image de droite).

FIG. 1.5 – Les axes de l'orientation de la main.

- ***Le mouvement***. Contrairement à certaines approches qui considèrent le mouvement de la main et des doigts de manière globale (notamment celle de Stockoe), ici le mouvement correspond uniquement à la trajectoire décrite par la main (ligne, arc, cercle ...). Durant un signe, ce paramètre peut aussi bien être nul, constant, combiner plusieurs phases différentes, ou être répétitif. Le mouvement peut servir à décrire des trajectoires , des mouvements, des formes, etc.
- L'***emplacement***. Ce paramètre correspond à une zone où se situe la main par rapport au corps.
  L'emplacement peut être décrit à plusieurs degrés de granularité selon les besoins. On peut en effet, énumérer des zones plus ou moins grandes au sein desquelles sont réalisés les signes.

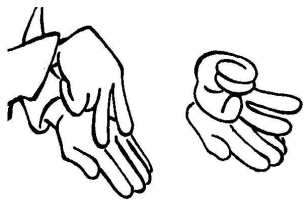

FIG. 1.6 – Deux exemples d'orientations : "une personne debout", "une personne couchée". (CISA - 2004)

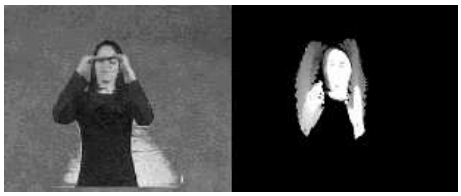

FIG. 1.7 – Exemple de mouvements linéaires effectués pas les deux mains. (IRIT Toulouse)

Par exemple, le premier signe de la figure 1.8 (GARÇON) s'effectue dans la zone du visage à l'emplacement du front (emplacement très précis). Pour le deuxième signe, on utilise l'espace de narration pour définir l'emplacement de l'entité "garçon", ce qui permet d'indiquer qu'il se situe *à droite* (zone beaucoup plus floue).
Du fait de l'utilisation de la scène de narration en langue des signes, ce paramètre joue évidemment un rôle très important lors de l'interprétation d'une phrase.

Il faut bien noter que ces paramètres sont relativement indépendants les uns des autres et amènent chacun du sens au signe. Parfois, la modification d'un seul paramètre suffit à changer complètement la signification d'un signe. De plus, ils sont de nature très différente ; par exemple, la configuration est de nature plutôt symbolique ou iconique alors que l'orientation est plutôt de nature continue. Le mouvement est de nature dynamique alors que l'emplacement est statique. Ces caractéristiques ne vont évidemment pas faciliter le travail d'un système de reconnaissance.

1.2.3 Lexiques et mécanismes de la langue des signes

Si l'on considère la "stabilité" des paramètres, différentes classes de signes peuvent être dégagées de la langue des signes, et plus particulièrement deux ensembles de signes peuvent être distingués.

Le premier est appelé lexique ***standard*** ou ***institutionnalisé*** ; il correspond à un ensemble de signes dont les paramètres et le sens sont clairement établis au sein de la communauté sourde. C'est un lexique qui est généralement complètement différent d'un pays à l'autre, et peut même différer au sein d'un pays suivant l'endroit où le sourd a appris la langue des signes. Par exemple, un sourd de Lyon n'aura pas la même manière de réaliser le signe [POULE] qu'un sourd de la région parisienne. Les signes de ce lexique peuvent être en général directement traduits en mots d'une langue orale.

La deuxième classe d'unités gestuelles (que l'on appelera ***signes de grande iconicité*** par la suite), intervient dans les énoncés faisant appel à la ***grande iconicité*** [Cuxac, 2000]. Ce mécanisme

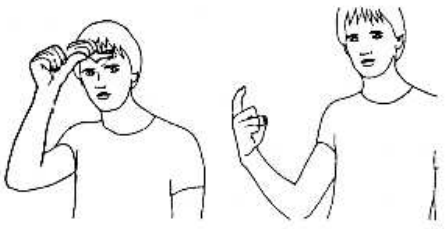

FIG. 1.8 – Exemples d'emplacements (dictionnaire IVT).

intervient dès que l'on a besoin d'avoir un propos illustratif ou iconique et cela s'effectue de diverses manières suivant ce que l'on souhaite illustrer. Ces différents moyens sont appelés ***transferts***. Par exemple, les transferts ***de taille et/ou de forme*** permettent, comme leur nom l'indique, d'effectuer la description d'un objet, d'un animal, d'une personne... Par exemple, dans la figure 1.9 le transfert sert à décrire une fleur.

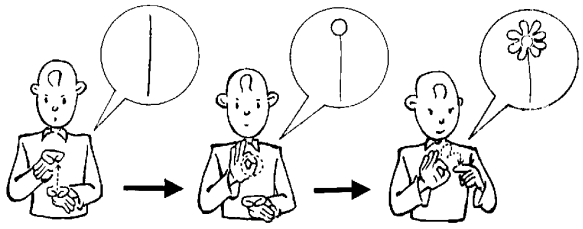

FIG. 1.9 – Transfert de taille et/ou de forme : description d'une marguerite. (CISA - 2004)

Les transferts ***situationnels*** servent à décrire une scène pour indiquer la disposition des différents éléments qui la composent, ou pour décrire le déroulement d'une action. Dans la figure 1.10, le transfert permet d'illustrer le fait qu'un chien rentre dans sa niche.

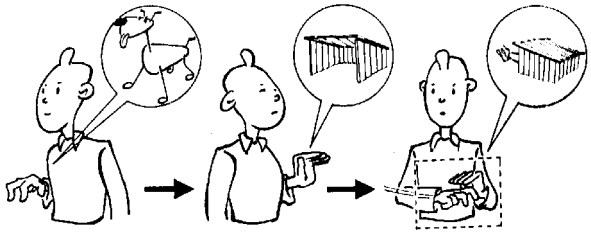

FIG. 1.10 – Transfert de situation : un chien qui rentre dans sa niche. (CISA - 2004)

Les transferts ***personnels*** donnent la possibilité au narrateur d'endosser le rôle d'une des entités du discours. Cette entité peut aussi bien être un être vivant qu'un objet inanimé. Par exemple un signeur va jouer le rôle d'un gâteau en train de cuire dans le four et va représenter le gonflement de la pâte en faisant progressivement : gonfler ses joues, ouvrir ses paupières et lever/écarter ses bras qui symbolisent la surface du gâteau. Ainsi il peut décrire de manière imagée le gonflement de la pâte (voir la figure 1.11). Il existe d'autres type de transferts plus complexes qui correspondent

à des combinaisons de plusieurs transferts.

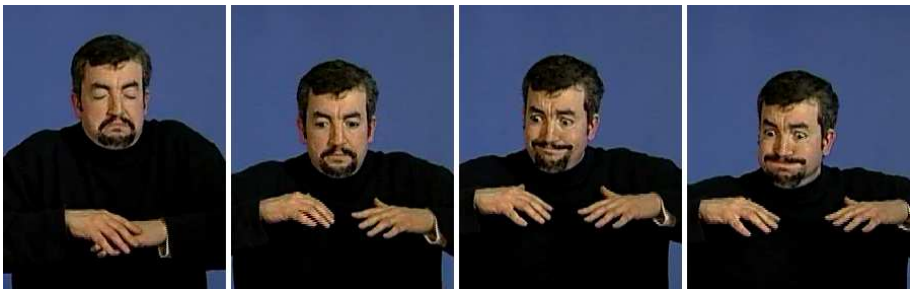

FIG. 1.11 – Transfert personnel : un gâteau qui gonfle à la cuisson. (LS-COLIN)

Notons que les paramètres dits ***non-manuels*** (regard, mimique, mouvement de la tête, des épaules et du buste) ont un rôle tout aussi important que les paramètres ***manuels***, particulièrement dans le cas des transferts.

Le lexique utilisé par la grande iconicité est constitué de signes dont les paramètres sont, soit partiellement fixés, soit totalement libres. Les paramètres laissés libres vont permettre de donner le sens final du signe ou de donner la signification de ce signe par rapport aux autre signes de la phrase.
On peut dégager deux principales classes de vocabulaire dans ce lexique : les ***spécificateurs de forme et de taille*** et les ***proformes***.
Les spécificateurs de taille et de forme ont toute liberté dans le choix des paramètres. Ces signes sont principalement utilisés dans les transferts de taille et/ou forme pour décrire des entités (objets, animaux, personnes ...) de manière visuelle. Les paramètres vont donc être choisis en fonction de ce que l'on décrit, et ce sont donc ces paramètres qui vont déterminer la signification du signe. Si l'on veut par exemple décrire un vase, une main va représenter la base pendant que l'autre va décrire la forme du vase.
Les proformes ont un rôle différent du fait qu'elles servent de pronoms. Elles n'ont pas de "sens" propre mais plutôt un rôle au niveau syntaxique. Elles ont une configuration bien précise qui permet de les relier à l'entité qu'elles représentent (par analogie de forme) ou d'indiquer quelle est leur fonctionnalité (conteneur, support ...). Les autres paramètres vont permettre ensuite de décrire la situation de l'entité représentée par la proforme dans la scène narrative. La position et l'orientation permettent ainsi de donner la disposition de l'entité au sein de la scène et le mouvement donne l'éventuel déplacement effectué par l'entité dans la scène. Ces signes sont naturellement utilisés lors des transferts situationnels qui servent à décrire des scènes et des actions.

Lorsque l'on regarde le vocabulaire standard, on ne peut s'empêcher de remarquer que certains signes présentent les caractéristiques des signes de la grande iconicité. Cela est dû au fait que lorsqu'un signe non standard est fréquemment usité avec les mêmes valeurs de paramètres par la communauté sourde pour signifier quelque chose, il devient de fait un signe standard. Par exemple, le signe [AÉROPORT] (figure 1.12) qui est recensé dans les dictionnaires de l'IVT [Moody, 1983, Moody, 1986, Girod et Vourc'h, 1981] provient clairement d'un transfert situation-

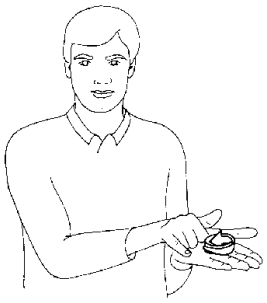

FIG. 1.12 – Exemple de transfert depuis la grande iconicité vers le vocabulaire standard : le signe [AÉROPORT]. (Dictionnaire de l'IVT)

nel où l'on décrit, à l'aide de proformes, un endroit où sont garés des avions.

Enfin, certains signes ne présentent ni tout à fait les caractéristiques des signes standard ni celles des signes faisant partie de la grande iconicité. Ils ont un sens bien défini mais certains de leurs paramètres vont être modifiés suivant le contexte de la phrase.
C'est le cas des ***verbes directionnels*** dont le mouvement va permettre d'indiquer quels sont les acteurs et objets de l'action décrite par le verbe. Ainsi, le mouvement du verbe [DONNER] ne sera pas le même entre "je te donne" et "il te donne". D'autre part, la configuration de la main peut changer suivant l'entité sur laquelle agit l'action. Par exemple, selon que l'on donne un livre, un ballon ou un paquet, le signeur va faire mine de tenir un objet mince avec une main, ou va porter un objet volumineux rond ou carré avec les deux mains (figure 1.13).

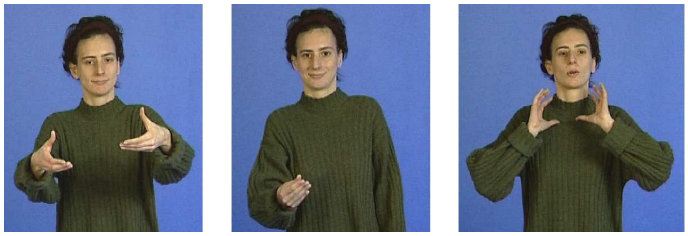

FIG. 1.13 – Le verbe directionnel [DONNER] appliqué à un colis et un livre (images de gauche), et le verbe [LANCER] utilisé pour un ballon (image de droite). (ARC-LSF)

Dans une phrase de la langue des signes, les différents types de signes peuvent se mélanger. Par exemple, dans la figure 1.14, on voit successivement l'utilisation :

1. d'un spécificateur de taille et de forme qui décrit une caisse,
2. du signe standard [POMME],
3. d'une proforme qui représente une pomme.

Cette thèse se focalise sur les interactions à deux mains, c'est pourquoi ce sont principalement les phrases faisant intervenir des transferts situationnels qui sont utilisées comme champ d'application

FIG. 1.14 – "Une caisse pleine de pommes" (Dictionnaire IVT).

de ce travail. En effet, dans ces cas là, les deux mains vont souvent interagir entre elles pour exprimer différentes informations spatiales et temporelles entre entités représentées par des proformes ou des signes standards. De plus, ce type d'énoncé a une nature assez proche de celle que l'on voudrait véhiculer dans le cas de l'interaction bimanuelle en réalité virtuelle.

La section suivante présente plus en détail, pour la langue des signes, les différents cas qui peuvent se produire lorsque les mains interagissent entre elles, mais aussi quels sont les problèmes qui découlent de ces interactions.

1.3 Nature des relations entre les deux mains

En langue des signes, différents types d'interactions entre les deux mains peuvent se présenter. Ces interactions peuvent avoir lieu au sein d'un seul signe où interviennent les deux mains (par la suite ***signe bimanuel***), mais aussi entre deux signes distincts et simultanés qui sont effectués chacun par une main (appelés par la suite ***signes monomanuels***). Ces interactions peuvent se manifester sous forme de ***synchronisme*** ou d'***asynchronisme*** et permettent d'exprimer en général des informations spatiales et/ou temporelles. Les définitions de synchrone et asynchrone sont ici les suivantes : par synchrone on entend une similarité (identité ou symétrie) des paramètres des mains durant un même intervalle de temps. Par asynchrone, on entend une dissimilarité entre les mains au niveau de leurs paramètres et de leur évolution au cours du temps ; mais cela n'exclut pas d'évoluer durant le même laps de temps et de partager des points communs qui permettent de donner la relation entre les deux mains.

1.3.1 Signe bimanuel

Lorsque les deux mains sont impliquées dans un signe, deux cas se présentent.
Dans le premier cas on voit apparaître un rôle pour chaque main. Une main est dite ***dominante*** et a pour rôle de décrire "l'action", tandis que l'autre, qui est appelée main ***dominée***, sert de référence à cette action. Par exemple, avec le signe [DÉCOLLER] (figure 1.15), la main dominée (main gauche dans la figure) représente un support qui sert de référence à la main dominante qui mime l'action de décoller de ce support. Du fait de leur rôle respectif, la main dominante en général se déplace au cours du geste tandis que la main dominée reste statique. On n'a donc, pour ce cas de signe bimanuel, aucune synchronisation au niveau des gestes des mains (pas de similarité entre la configuration, le mouvement et l'orientation). Par contre des relations temporelles et spatiales

entre les mains permettent de décrire l'action. Dans la figure 1.15, on peut voir la main dominante et la main dominée qui partagent le même emplacement et sont en contact en début de signe. Cela permet d'établir une relation entre les deux mains qui permet de comprendre l'action de la main dominante qui "décolle" de la main dominée.

Dans le deuxième cas par contre, les deux mains sont complètement synchronisées : leurs configurations sont identiques, les mouvements et orientations sont eux identiques ou symétriques. Par exemple, avec le signe [TABLE] (figure 1.15), les deux mains ont une configuration et une orientation identiques et des mouvements identiques mais de sens opposés. Dans ce cas, on n'a pas une main qui sert à exprimer une action par rapport à l'autre, mais deux mains qui collaborent pour décrire une propriété de forme (la surface plane d'une table).

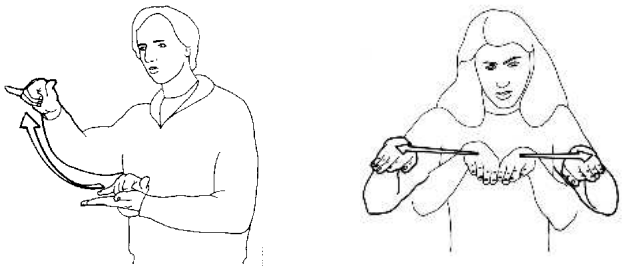

FIG. 1.15 – Les signes bimanuels [DÉCOLLER] et [TABLE] (dictionnaire de l'IVT).

1.3.2 Signes monomanuels

Lorsqu'un signe ne fait intervenir qu'une seule main, le signeur a la possibilité de réaliser un autre signe avec sa deuxième main. On peut alors retrouver dans ce cas une interaction de type main dominante-main dominée entre les deux signes monomanuels (comme dans la figure 1.16), pour pouvoir exprimer des informations spatiales et/ou temporelles. Cela peut être utilisé, par exemple, pour décrire spatialement la disposition d'objets dans une pièce. On peut avoir également des cas de synchronisation des mouvements et des orientations entre les deux mains. Cela permet d'exprimer des propriétés de similarité ou d'opposition entre les deux entités représentées par les signes. Par exemple, l'opposition des orientations permet d'exprimer le fait de "se faire face", deux mouvements similaires indiquent que les deux entités se suivent, etc.

1.3.3 Signe bimanuel versus signes monomanuels

En langue des signes, on peut utiliser dans une phrase aussi bien des signes monomanuels qu'un signe bimanuel. Or on a vu que ces deux cas sont fortement similaires : mécanismes de main dominée/dominante, synchronisation de paramètres... Cela est dû en partie au fait que ces propriétés sont issues des transferts de la grande iconicité, et que certaines expressions effectuées avec deux signes monomanuels en grande iconicité se retrouvent *institutionnalisées* sous forme d'un signe bimanuel à deux mains dans le vocabulaire standard (comme le signe tomber dans la figure 1.17).

FIG. 1.16 – Utilisation de deux proformes monomanuelles pour exprimer le fait de "tomber de quelque chose" (dictionnaire de l'IVT).

FIG. 1.17 – Signe bimanuel [TOMBER] (dictionnaire de l'IVT).

Seul le regard peut permettre parfois de différencier un signe standard bimanuel de deux signes de grande iconicité simultanés. En effet, lors d'un signe standard, le regard est généralement dirigé vers l'interlocuteur, alors que pour des signes de la grande iconicité le regard peut accompagner les mains.
Du fait que nous nous restreignons uniquement aux mains, le mélange de deux types de signes pour les mêmes types d'interaction entre les mains va poser un réel problème lors de l'étape d'interprétation dans un processus de traitement automatique de la langue des signes. Dans le cas d'un signe bimanuel, l'information exprimée par les relations entre les mains est en général incluse dans la signification du signe. Par contre, lorsque l'on a affaire à deux signes monomanuels simultanés, il va falloir extraire les informations spatio-temporelles éventuellement contenues dans les relations entre les deux mains.
Un premier problème, auquel ce travail doit s'attacher, est donc de proposer une méthode permettant de différencier un couple de signes à une main d'un signe à deux mains, dans le but de décider si l'on doit effectuer une recherche d'informations spatiales et/ou temporelles entre les mains.
Nous allons maintenant parler de la manière dont s'expriment ces informations que l'on doit rechercher dans les cas d'interactions entre les deux mains.

1.3.4 Expression d'informations sous forme de relations spatiales et/ou temporelles

On a vu dans la section précédente que des informations peuvent être exprimées via des relations entre les deux mains. Dans cette partie, on va s'intéresser à la manière dont sont exprimées ces informations et on va s'attacher plus particulièrement aux différents types de relations entre la main dominée et la main dominante.

Le signeur va jouer sur différentes propriétés (emplacement, mouvement, orientation) des signes pour arriver à exprimer des informations spatiales et/ou temporelles. Dans le cas de synchronisation des mains, ce sont les orientations et mouvements des mains qui vont être utilisés pour décrire des propriétés de forme (comme dans le signe [TABLE], fig 1.15) ou de symétrie (par exemple deux personnes face à face, fig 1.18). La synchronisation des mains permet également de décrire des situations où l'on a une similitude de comportement (personnes qui se suivent, véhicules qui avancent à même vitesse... fig 1.18).

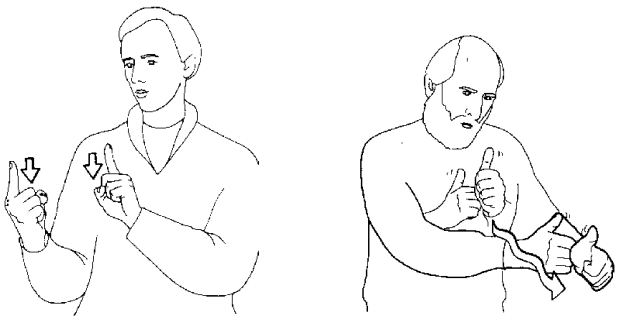

FIG. 1.18 – Signes [L'UN EN FACE DE L'AUTRE] et [POURSUIVRE] (dictionnaire de l'IVT).

A contrario, dans la situation main dominante/main dominée, l'asynchronisme des mains permet d'exprimer une "différence" entre deux entités. C'est à dire que l'on exprime le lien entre deux entités par le rapport qu'il y a entre les mains. Dans le cas de "tomber de ..." (fig 1.16) c'est la différence de mouvement/position entre les deux mains qui permet d'exprimer la relation que la main dominante tombe **de** la main dominée.

Pour pouvoir exprimer cette relation de main dominante/main dominée, il va falloir mettre en relation les deux signes formés par chaque main. Pour cela, le signeur va devoir effectuer des synchronisations temporelles et/ou spatiales entre ses mains. Dans cette thèse, nous nous focalisons sur les expressions de relations spatiales entre entités et dans ce cas il existe ce qui sera appelé des ***points de synchronisation***. Pendant un instant, les deux mains vont être agencées spatialement de manière à véhiculer une certaine information.

Par exemple, dans la phrase "Le chat est dans la voiture" (figure 1.19), on va tout d'abord énoncer le signe voiture(1ère image). Ensuite, le signeur positionne la voiture dans la scène de narration à l'aide d'une proforme (image 2), puis énonce le signe chat (image 3) qui lui même est substitué par une proforme. Enfin, la proforme représentant le chat se positionne relativement à la proforme représentant la voiture pour exprimer l'information spatiale "dans". C'est à cet instant

FIG. 1.19 – "Le chat est dans la voiture". (LS-DANCE, LIMSI-CNRS)

bien précis que la relation spatiale est exprimée. Ce n'est pas avant lorsque l'on a créé la scène, ni après quand le signeur va commencer une autre phrase en revenant à une posture neutre.

Bien évidemment, ce point de synchronisation peut se produire à n'importe quel endroit d'une phrase. Ce moment va dépendre du contexte de la phrase et de ce que signifie la phrase. Tout le problème va donc être d'arriver à détecter, pendant une séquence de signes co-occurents, ce laps de temps particulier durant lequel les mains ont une disposition particulière qui permet d'exprimer une relation entre les signes.

1.4 Bilan

Du fait que nous nous positionnons principalement sur la fonctionnalité sémiotique des gestes et que la langue des signes correspond au plus haut degré d'expression de cette fonctionnalité, seul ce domaine a été détaillé dans ce chapitre. Toutefois, tout ce qui a été vu ici reste valable pour d'autres domaines où l'on retrouve les mêmes problèmatiques liées au geste.

Nous avons constaté que les gestes, lorsqu'ils se succèdent au sein d'une phrase, s'organisent dans l'espace afin de construire un fil conducteur entre eux, pour décrire une certaine disposition spatiale ou pour indiquer un ordre temporel. De cela, on peut en déduire que l'interprétation de phrases gestuelles nécessite des systèmes capables d'intégrer cette structuration spatiale.
Outre ce premier fait, on doit également être capable de prendre en compte l'aspect multicanal des gestes. En effet, même si différents modèles linguistiques existent au sein de la communauté, tous s'accordent sur un fait : un geste est composé de plusieurs éléments. En partie du fait de cet aspect des gestes et en partie du fait de l'aspect iconique des gestes, une grande variabilité est laissée au vocabulaire gestuel. Cela, aussi, rend plus compliqué l'interprétation de gestes.
Enfin, l'utilisation de gestes bimanuels ajoute de nouvelles problématiques. Tout d'abord, nous avons vu dans ce chapitre que gestes monomanuels et gestes bimanuels peuvent alterner et qu'il n'est pas possible de les distinguer aisément les un des autres. Ce qui, là encore, rend plus hardu la tâche d'interprétation de gestes. Nous avons ensuite observé qu'entre gestes monomanuels des relations peuvent s'établir afin d'exprimer des informations de type spatiales et/ou temporelles. Le défi consiste alors à détecter et interpréter ces relations.

Dans le chapitre suivant, nous allons passer à une description et un état des lieux de la recon-

naissance de gestes. Nous revenons également sur les différents problèmes soulevés ici pour indiquer quels sont ceux qui ont été traités et à l'aide de quelle technique cela a été fait.

Chapitre 2

État de l'art en reconnaissance de gestes

Dans ce chapitre, nous allons décrire le principe de fonctionnement d'un système de reconnaissance de formes, ainsi que les travaux qui ont été effectués dans le domaine du geste et plus particulièrement dans celui de la langue des signes.
La première section donne une description d'un processus de reconnaissance dans le domaine du geste. Cela permet de positionner le contexte où l'on se place. Ensuite, la section deux présente les travaux effectués en reconnaissance de la langue des signes et indique quels sont les problèmes rencontrés dans ce domaine.

2.1 Reconnaissance automatique de gestes

La reconnaissance des formes est une étape qui permet de faire passer de l'information de notre univers (visuel, auditif, sensitif,...) à celui de la machine. Ce processus ne consiste pas simplement à transformer en binaire une source d'information analogique, mais à interpréter cette source en symboles exploitables. Cela s'effectue à travers différentes étapes qui s'effectuent la plupart du temps successivement [Belaïd et Belaïd, 1992] : l'acquisition, le traitement des données, la reconnaissance et l'interprétation (figure 2.1). Nous allons maintenant décrire ces différentes étapes dans un contexte de reconnaissance du geste.

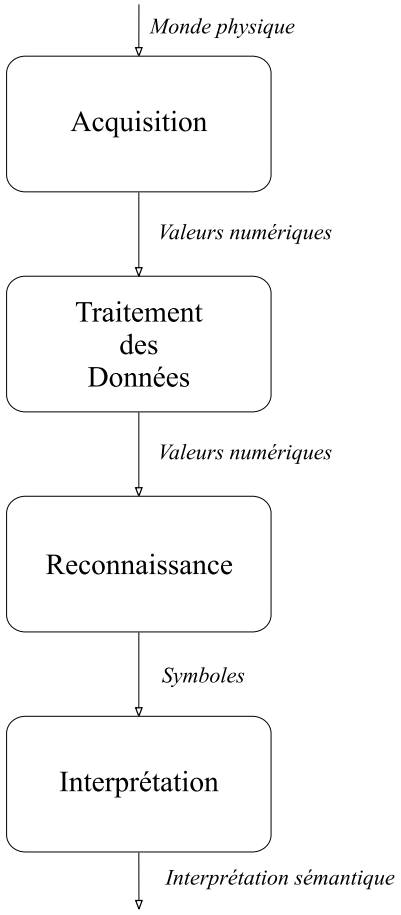

FIG. 2.1 – Les différentes étapes de la reconnaissance de formes

2.1.1 L'acquisition

La première étape consiste à capter les données physiques et à les encoder sous forme numérique. Pour cela de nombreux capteurs existent selon les différents types de sources données (sonores, mécaniques, visuels ...).

Dans le domaine du geste, l'acquisition peut se faire de deux manières : soit ***non intrusive*** (l'utilisateur est libre de tout système), soit ***intrusive*** (des marqueurs ou capteurs sont fixés sur l'utilisateur). Cela revient à effectuer l'acquisition soit via une modalité visuelle (on suit le mouvement apparent de la personne), soit via une modalité mécanique (on suit le déplacement des muscles et/ou des os de la personne).

Acquisition non-intrusive

Ce type d'acquisition se fait par des systèmes à base d'une ou plusieurs caméras. La multiplication des caméras permet d'éviter les problèmes d'occultation (par exemple, lorsque la main gauche s'interpose entre la caméra et la main droite) et donne la capacité de remonter à des informations en trois dimensions en mettant en correspondance les images issues des différentes caméras [Delamarre et Faugeras, 1998] (voir figure 2.2).
Si ce type de système donne une totale liberté à l'utilisateur, il nécessite par contre des contraintes sur son environnement. Il faut que l'éclairage soit suffisant, que l'utilisateur se détache bien du fond de l'image, etc.

FIG. 2.2 – Exemple de reconstruction 3d à partir de plusieurs points de vue (une seule caméra est affichée). (Projet Robotvis, INRIA.)

Acquisition intrusive

La méthode la moins intrusive consiste à placer des marqueurs sur le corps de la personne que l'on veut suivre, et ensuite, via un système de plusieurs caméras infrarouges, calculer la position dans l'espace de ces marqueurs (figure 2.3). Comme les autres systèmes de capture par caméra, ce système présente le problème des occultations qui génèrent des pertes d'informations.

Pour résoudre cela, il faut passer à des systèmes plus intrusifs où l'on fixe directement sur la personne des capteurs de position/orientation voir même des capteurs de flexion. On a alors une mesure directe et fiable des gestes ; par contre, on perd énormément d'ergonomie au niveau de la

FIG. 2.3 – Système d'acquisition par marqueurs (NYU Motion Capture Studio)

liberté de l'utilisateur du fait que les capteurs ont besoin d'être reliés à un ordinateur ou des boîtiers électroniques. Toutefois, cet inconvénient peut être pallié par l'utilisation d'émetteurs/récepteurs radio en remplacement des fils. Par exemple, dans le domaine du suivi de mouvements du corps, l'ensemble des capteurs de position/orientation sont raccordés à un sac à dos où se trouve un émetteur et une batterie (comme dans la partie droite de la figure 2.4).

FIG. 2.4 – Le système de capture de position/orientation MotionStar™. A gauche le système avec fils, à droite le même système avec un émetteur radio. (Ascension Technology)

Les ***gants numériques*** sont les périphériques intrusifs les plus caractéristiques du domaine du geste. Ils consistent en un gant rempli de capteurs permettant de connaître la flexion de chaque articulation de la main. Ils sont souvent associés à un capteur de position/orientation permettant de connaître le mouvement de la main (exemple dans la figure 2.5).

FIG. 2.5 – Exemple d'utilisation de gants numériques et de capteurs de position/orientation. (LIMSI-CNRS)

Selon le type d'acquisition réalisé, les données captées vont être de nature très différentes et vont nécessiter un traitement spécifique afin de pouvoir être utilisables par le système de reconnaissance.

2.1.2 Le traitement des données

Une fois les données collectées par un système d'acquisition, plusieurs traitements vont devoir être appliqués : le filtrage, l'extraction d'informations, et le choix/calcul de primitives.

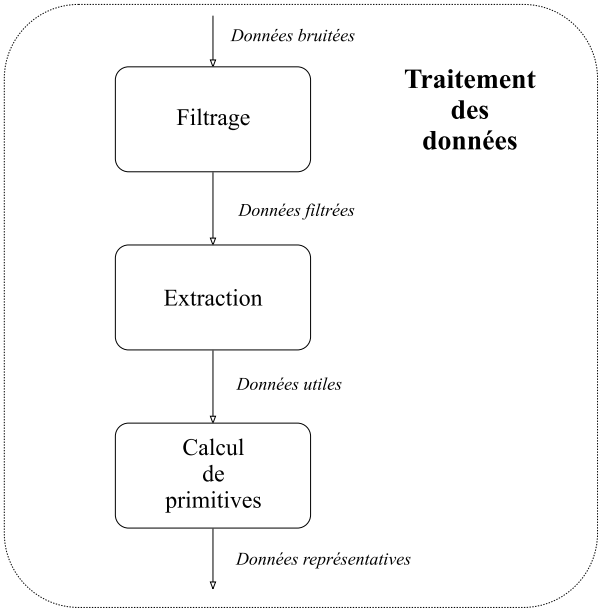

FIG. 2.6 – L'étape de traitement des données

Filtrage

La technique utilisée pour effectuer l'acquisition de données peut parfois introduire des imperfections (bruit, données non homogènes, etc). L'étape du filtrage a pour but de corriger ces imperfections. Diverses méthodes mathématiques existent pour effectuer plusieurs types de corrections. Par exemple pour "lisser" un signal, on peut effectuer une moyenne sur les dernières valeurs acquises.

Extraction d'informations

Ensuite, suivant le type d'acquisition, il va falloir extraire les données du média. Par exemple, dans le cas de capture par caméra, on va devoir dégager les informations pertinentes de l'image. Dans [Cassel et Collet, 2003], les auteurs veulent pouvoir effectuer le suivi de trampolinistes, il faut donc arriver à extraire le corps du sportif du fond de l'image. Cela est effectué grâce à une différence d'image entre un fond et l'image courante (voir figure 2.7).
Ces traitements ont principalement lieu dans le domaine de l'acquisition par caméra, les autres moyens fournissant directement les données, par exemple des coordonnées et des angles dans le cas d'un capteur de position/orientation.

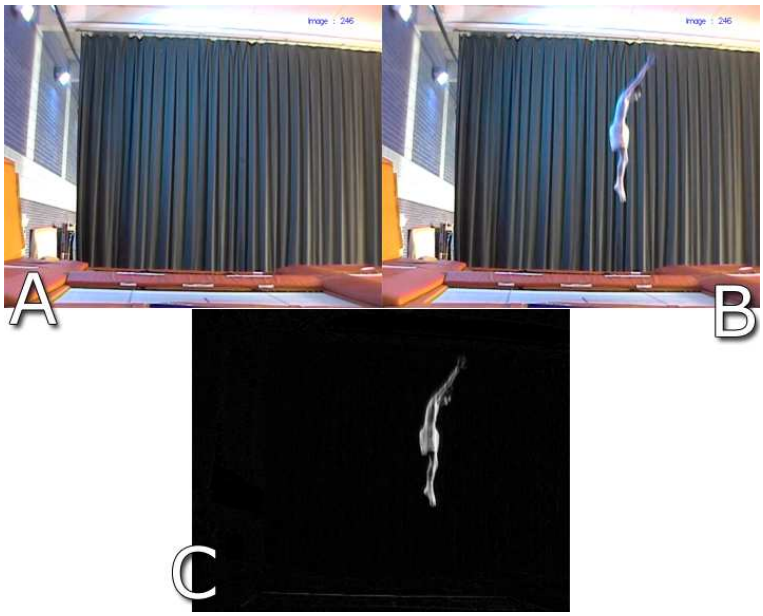

FIG. 2.7 – Extraction des données utiles d'une image par soustraction ($B - A \Rightarrow C$). (LIMSI-CNRS)

Choix/calcul de primitives

La dernière étape consiste à choisir ou calculer les données caractéristiques qui vont permettre de distinguer les différentes entités que l'on veut pouvoir reconnaître. On appelle ces données ***primitives***. Un bon choix de primitives est essentiel car c'est de ce choix que va dépendre la qualité de la suite du processus de reconnaissance. Par exemple, pour effectuer la reconnaissance de caractères manuscrits, l'angle initial, la courbure du tracé, la distance parcourue peuvent être

utilisés [Rubine, 1991]. Dans [Starner et Pentland, 1995], une fois que l'emplacement de la main est déterminé, les primitives choisies sont la position, l'excentricité et l'orientation de l'ellipse englobant la main sur l'image.

2.1.3 La reconnaissance

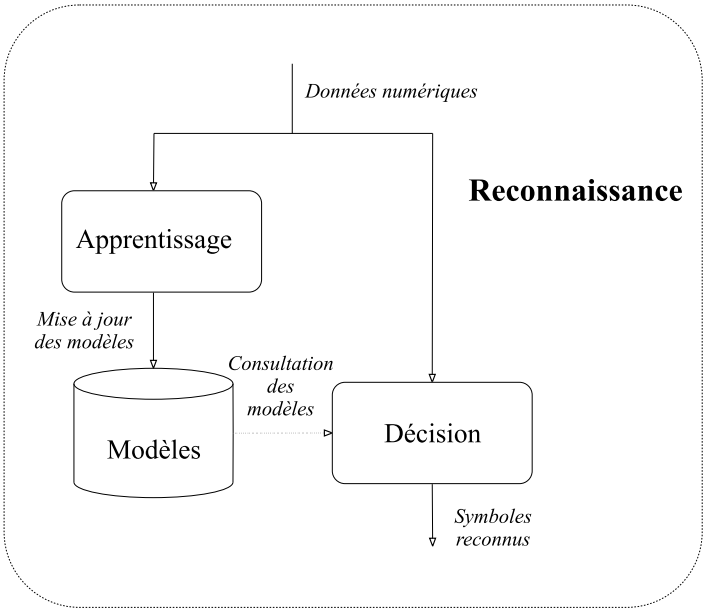

FIG. 2.8 – L'étape de la reconnaissance

La reconnaissance est l'étape qui va permettre de passer du domaine du signal continu au domaine symbolique. C'est donc lors de la phase de reconnaissance que l'on va transcrire le flux de données en une suite de symboles.

Pour effectuer cela, des méthodes basées sur la symbolique ou la statistique peuvent être mises en oeuvre. Elles effectuent la modélisation des données sous forme de ***classes***. Pour chaque élément du vocabulaire à reconnaitre, on aura une classe qui le représente sous une forme utilisable par la technique de reconnaissance (statistiques, probabilités, règles...). La constitution de ces classes requiert généralement une phase dite ***d'apprentissage***.

Apprentissage

La modélisation des différentes classes se fait le plus souvent au travers de diverses structures comme des matrices de covariance, des vecteurs moyens, des graphes, etc. L'apprentissage consiste à fournir au système de reconnaissance des ensembles d'exemples qui vont lui permettre de créer ces structures qui composent les classes. Un exemple correspond à un ensemble de valeurs (vecteur de primitives) étiqueté avec le symbole de la classe que l'on veut enrichir avec cet exemple. Les classes vont donc se préciser, s'affiner, au fur et à mesure que l'on rajoute de nouveaux exemples. On voit donc que la qualité des exemples est déterminante pour avoir une modélisation correcte des classes. De plus, il est recommandé de faire apprendre à un système différents exemples qui

varient légèrement, cela afin de ne pas avoir une modélisation trop restreinte de la classe.

Le choix des exemples peut se faire de manière supervisée ou non supervisée. Dans le premier cas, c'est l'utilisateur du système qui doit lui même fournir des exemples au système et qui décide donc des différentes classes et des symboles que l'on doit recevoir en sortie. Dans le second cas, c'est le système lui même qui doit dégager les différentes classes et effectuer l'apprentissage de ces classes à partir de la masse de données reçue en entrée.

Une fois l'apprentissage effectué, on va pouvoir procéder à la reconnaissance des données qui sont fournies au système. C'est l'étape de décision qui s'occupe d'effectuer le choix de la classe correspondant aux données.

Décision

L'étape de décision (ou inférence) revient à effectuer une comparaison des données reçues en entrée à toutes les classes stockées dans le système de reconnaissance. Chaque comparaison va donner une évaluation de la ressemblance entre les données et la classe considérée. La classe qui sera choisie comme étant représentative des données est celle qui aura eu la meilleure évaluation. Ainsi, cette étape est basée sur un principe très simple et assez robuste. Toutefois, des problèmes vont surgir lorsque plusieurs classes donnent lieu à des évaluations voisines ou lorsque des données ne correspondant à aucune classe du système sont fournies en entrée. On est alors obligé d'effectuer après l'étape de décision d'autres traitements permettant d'effectuer le choix final quant au symbole reconnu.

Dans le domaine du geste, de nombreux systèmes, utilisant des techniques très diverses, ont été testés. On peut citer par exemple la logique floue [Bimber, 1999] (réalité virtuelle), les graphes [Cutler et al., 1997] (réalité virtuelle), les automates [Latoschik, 2001] (réalité virtuelle), les arbres de décision [Kadous, 1996] (langue des signes), etc.
De toutes les techniques possibles et existantes, trois d'entre-elles sont plus souvent utilisées : les modèles statistiques, les réseaux de neurones et les modèles de Markov cachés.

Modèles statistiques

Ces techniques (instance based learning, pattern matching, etc) ont été souvent utilisées dans le domaine de la reconnaissance de l'écriture manuscrite (par exemple [Rubine, 1991]) et ont ensuite été étendues aux autres domaines du geste (comme les gestes isolés en réalité virtuelle [Newby, 1993]).

Les méthodes statistiques consistent à modéliser sous forme de vecteurs les différents éléments que l'on veut reconnaître (phase d'apprentissage) et ensuite à estimer la différence qu'il y a entre les données en entrée et les différents vecteurs appris. Le vecteur qui ressemble le plus aux données en entrée est choisi comme étant l'élément reconnu. L'avantage de ces méthodes est leur facilité de mise en oeuvre et leur temps de réponse très rapide. Cependant, elles sont peu robustes pour des vocabulaires de taille conséquente, peu adaptées pour traiter les aspects temporels et nécessitent

en général l'ajout d'un système de segmentation[1].

Les réseaux de neurones

Ces modèles qui s'inspirent du fonctionnement du réseau de neurones du cerveau ont été particulièrement en vogue en reconnaissance de la langue des signes au début des années 90 ([Harling, 1993] ; [Murakami et Taguchi, 1992] ; [Gourley, 1994]). Ils ont tendance, aujourd'hui, à être délaissés ou combinés avec d'autres systèmes pour la reconnaissance de signes manuels.
Ces modèles nécessitent un apprentissage avant de pouvoir effectuer une classification des gestes fournis en entrée du réseau de neurones. Ils ne sont pas aisés à mettre en oeuvre lorsque l'on cherche à reconnaître des données temporelles (problèmes de segmentation, de choix d'architecture de réseau, d'apprentissage [Collet, 1993]).

Les modèles de Markov cachés

Ces modèles ont d'abord été introduits dans le domaine de la parole [Rabiner, 1989] et ont pu être directement importés dans le domaine de la reconnaissance de la langue des signes ([Braffort, 1996b] ; [Starner et Pentland, 1995]). Ils consistent à modéliser un processus à l'aide d'un graphe d'états. On peut alors modéliser un geste grâce à une succession d'états qui représentent chacun une partie du signe. Le passage d'un état à un autre se fait en fonction de probabilités basées sur ce qui s'est déroulé aux états précédents.
Ils présentent le grand avantage, lorsque qu'on les met en réseau, de pouvoir modéliser une succession de gestes. D'autre part, ils sont souvent à la base de systèmes de reconnaissance destinés à traiter de grands vocabulaires. Là encore, le domaine gestuel s'inspire des travaux effectués en parole [Gauvain, 2000].

2.1.4 L'analyse et interprétation

La dernière étape consiste à analyser et interpréter les éléments que l'on vient de reconnaître. Cette analyse va s'effectuer à un niveau syntaxique et/ou sémantique. Elle va donc être uniquement présente dans des contextes où les éléments présentent une structure de nature linguistique.
Dans le domaine gestuel, ce sont principalement la reconnaissance de langue des signes, des mimiques et des gestes coverbaux qui vont nécessiter une telle étape d'analyse. Par exemple, pour les gestes coverbaux, il va falloir analyser les gestes effectués par rapport aux mots prononcés et ensuite, en fonction des liens trouvés, donner une interprétation à ce geste et à la phrase correspondante. Actuellement, encore très peu de systèmes intègrent cette phase d'analyse, ou alors de manière partielle.

Maintenant que l'on a vu comment fonctionne un système de reconnaissance et quelles sont les techniques existant dans le domaine du geste, nous allons détailler les difficultés de mise en oeuvre des techniques de reconnaissance dans le domaine spécifique de la langue des signes. Dans le même temps, nous dresserons un panorama des divers systèmes et techniques implémentés pour la langue des signes.

[1] cette tâche consiste à savoir découper une suite temporelle de données en plusieurs éléments correspondant à des symboles que l'on doit identifier. Nous reviendrons plus précisément sur ce problème en section 2.2.2.

2.2 Les problématiques rencontrées dans le domaine de la langue des signes

La langue des signes possède un certain nombre de caractéristiques qui font qu'on ne peut pas utiliser "simplement" les techniques de reconnaissance habituellement utilisées dans le domaine gestuel. Nous allons maintenant expliquer quels sont les problèmes auxquels sont confrontés ces techniques de reconnaissance et qui demandent des solutions parfois assez complexes : le lexique, la segmentation et l'interaction entre les mains.

2.2.1 Différentes catégories de signes

On a pu voir dans le premier chapitre qu'il y a principalement deux catégories de signes en langue des signes. Les signes standards qui ont une correspondance proche ou directe dans une langue orale et les signes non standards (signes de la grande iconicité, verbes directionnels ...) qui eux ne peuvent être "directement traduits".
A ces deux catégories se rajoute la dactylologie qui est un code gestuel correspondant aux lettres de l'alphabet et qui permet aux sourds d'épeler, entre autre, des noms propres .

Beaucoup de systèmes s'intéressent uniquement aux alphabets manuels et aux signes standards car la difficulté d'interprétation est moindre avec ce type de geste. En effet, à chaque classe de signes, on peut faire correspondre un mot ou une lettre.
Divers types de système de reconnaissance peuvent alors être aisément mis en oeuvre : réseaux de neurones [Harling, 1993], modèles de Markov cachés [Starner et Pentland, 1995], arbres de décisions [Kadous, 1996], systèmes statistiques, etc.

Signes standards : problème de taille de vocabulaire

Avec les signes standards toutefois apparaît un problème de complexité lié à la taille du vocabulaire. Même si ces signes sont moins nombreux que les mots des langues orales (quelques milliers d'éléments pour une langue des signes, plusieurs dizaines de milliers pour une langue orale), on se retrouve confronté au même problème avec la langue des signes : comment arriver à effectuer l'apprentissage de vocabulaires aussi grands et comment avoir une reconnaissance fiable et efficace de ce vocabulaire ?

Une première approche consiste à essayer de segmenter le vocabulaire en plusieurs parties pour répartir plus facilement son traitement sur le système de reconnaissance. Par exemple, dans [Fang et al., 2004] le système combine des arbres de décision à des modèles de Markov cachés. Les arbres de décision permettent de répartir le vocabulaire sur les modèles de Markov cachés. On a ainsi moins de modèles de Markov cachés impliqués lors de l'apprentissage et de la reconnaissance, ce qui permet d'améliorer les performances du système.
Dans [Bauer et Kraiss, 2001], le système réalise une décomposition des signes basée sur une approche phénomique [Jelinek, 1998]. C'est à dire que le vocabulaire est décomposé en éléments qui apparaissent régulièrement (on dégage ces éléments avec des techniques de clustering[2]).

[2]Une technique de clustering consiste à effectuer la division d'un ensemble de données en plusieurs groupes (cluster). L'identification des éléments qui constituent ces différents groupes peut s'effectuer suivant divers critères : fréquence d'apparition, distances entre éléments, etc.

Une deuxième approche consiste à utiliser les propriétés du vocabulaire pour décomposer les signes. On a vu dans le Chapitre 1 que les signes sont composés de plusieurs paramètres (configuration, emplacement, orientation et mouvement). Chacun de ces paramètres constitue lui même un vocabulaire de plusieurs dizaines ou centaines d'éléments [Braffort, 1996b]. C'est la composition de ces paramètres qui génère la complexité en taille du vocabulaire [Vogler et Metaxas, 1999b], aussi bien standard que non standard. Une approche paramétrique permet donc d'avoir à reconnaître seulement quelques centaines d'éléments au lieu de plusieurs milliers, mais il faut alors pouvoir traiter plusieurs éléments en parallèle.
Dans [Vogler et Metaxas, 1999a], la solution proposée consiste à effectuer une modification des modèles de Markov cachés, ce qui permet de traiter plusieurs canaux parallèles et de gérer des points de rencontre entre ces canaux. Cela a été testé sur les deux canaux que sont la main gauche et la main droite et pourrait être étendu aux différents paramètres. On trouve dans [Liang et Ouhyoung, 1998], un exemple d'architecture où les différents paramètres sont traités séparément en parallèle avant d'être comparés à des modèles de signes.

A noter que la quasi absence de corpus de langue des signes de grande taille freine grandement le développement de solutions à ce problème. Alors que dans d'autres domaines, comme celui de la parole, de nombreux corpus de tous types sont disponibles pour la recherche.

Signes non standards : problème d'interprétation

Actuellement, encore peu d'études se sont penchées sur le domaine des signes non standards du fait de la complexité à interpréter de tels signes. En effet, il s'agit de signes dont la réalisation est variable en fonction du contexte, l'ensemble des variations possibles pouvant être illimité dans certains cas.
Jusqu'à maintenant, c'est principalement les verbes directionnels qui ont été traités. On a vu au chapitre 1 que pour ces verbes, la direction du mouvement de la main va indiquer qui sont l'agent et le destinataire du verbe. En plus de savoir reconnaître le verbe lui même, un système de reconnaissance va devoir être capable de gérer la scène de narration afin de pouvoir déterminer la conjugaison spatiale du verbe.

Dans [Braffort, 1996b], deux modules de reconnaissance fonctionnent en parallèle : un qui est dédié au traitement du vocabulaire standard, et un second dédié à l'interprétation des signes non standards (proformes et verbes directionnels). Un troisième module se charge ensuite de relier entre elles les différentes entités émises par les modules de reconnaissance. Pour cela, il se base sur des règles faisant intervenir des critères basés sur la syntaxe (par exemple une proforme suit un signe standard), la sémantique (tel verbe fait intervenir tel type d'entités) et l'emplacement (le verbe part de tel emplacement, donc l'entité qui est à cet emplacement est le sujet).
On va ainsi réussir tout d'abord à identifier les différentes entités de la scène de narration, puis à interpréter les actions décrites par les verbes directionnels. Le système peut donc interpréter une phrase du type "Le garçon qui est à ma droite me donne un gâteau".
Dans [Sagawa et Takeuchi, 1999], les auteurs utilisent un algorithme qui permet de rechercher l'agent et le destinataire d'un verbe directionnel. Pour cela, ils effectuent un calcul de voisinage

spatial entre les points de départ et d'arrivée du verbe et l'emplacement des autres signes de la phrase.

A ce jour, le traitement des proformes n'a été traité que de manière superficielle et aucun système n'est capable d'effectuer l'interprétation de spécificateurs de forme et de taille. Cet aspect de la langue des signes est donc largement ouvert au développement de nouvelles solutions. Nous proposons, pour notre part, d'aller un peu plus loin dans l'interprétation des proformes.

En plus de la nécessité d'arriver à différencier les signes les uns des autres, un autre problème se pose lorsque l'on désire interpréter des phrases complètes de la langue des signes. En effet, les signes s'enchaînant les uns après les autres, il faut être capable de les séparer avant de les identifier.

2.2.2 Coarticulation : segmentation

Lorsqu'un signeur énonce une phrase en langue des signes, il ne revient pas entre chaque signe à une position de repos. Les signes se suivent dans un fondu enchaîné qui permet au signeur un moindre effort pour passer d'un signe à l'autre. Le phénomène de transition d'un signe à un autre est appelé ***coarticulation***. Ce phénomène gêne considérablement un processus de reconnaissance car il peut empêcher l'identification du début et de la fin d'un signe. De plus, les positions intermédiaires entre deux signes peuvent générer de fausses interprétations. Par exemple, pour passer de la configuration "moufle" à la configuration "A", la main va passer par la configuration "main à angle droit" qui est un passage obligé non intentionnel (voir figure 2.9).

FIG. 2.9 – Configurations *moufle*, *main à angle droit* et *A*. (Extraits du dictionnaire de l'IVT)

Il faut alors que le système soit capable d'identifier la séparation entre la configuration *moufle* et la configuration *A* afin de ne pas reconnaître l'étape intermédiaire *main à angle droit*. Cette tâche de découpage lexical, qui est essentielle pour un système de reconnaissance, est appellée ***segmentation***.

Les techniques à base de réseaux de neurones et de modèles statistiques ne sont pas très adaptés pour effectuer la reconnaissance de phrases continues de la langue des signes. On est obligé en général d'y adjoindre un système capable d'effectuer la segmentation de la phrase. Ces systèmes reposent sur la détection de changements dus au passage d'une phase de ***tenue*** du signe à une phase de ***changement*** en vue du signe suivant.
Par exemple, dans [Harling et Edwards, 1996], les auteurs ajoutent à un réseau de neurones un système permettant d'utiliser "l'effort" effectué pour tenir une certaine configuration de la main. Quand le système détecte un moment de "repos", c'est le passage de transition entre deux signes

et il effectue une segmentation.

Dans [Fang et al., 2004], c'est un réseau de neurones qui est utilisé pour effectuer la segmentation. Pour cela, le réseau possède une boucle entre ses étages qui permet de modéliser l'instant t et l'instant $t-1$ au sein du réseau. Le système est alors capable d'apprendre et de détecter les phases de transition entre deux signes.

Les modèles de Markov cachés sont, eux, par contre tout à fait adaptés à la modélisation des transitions entre signes. En particulier, la mise en réseau de modèles de Markov permet d'effectuer la reconnaissance de phrases continues de la langue des signes. Cette capacité à gérer la coarticulation en fait donc une technique favorite pour la reconnaissance de signes enchaînés. Elle est utilisée notamment dans [Starner et Pentland, 1995] sur des phrases dont la structure est figée. Dans [Vogler et Metaxas, 1997] et [Hienz et al., 1999], la reconnaissance est effectuée sur des phrases basées sur un vocabulaire allant d'une cinquantaine à une centaine de signes.
A noter que dans [Braffort, 1996b] et [Sagawa et Takeuchi, 1999] (cités précédemment à propos des verbes directionnels), en plus d'effectuer une reconnaissance de signes enchaînés, les systèmes sont capables de prendre en compte les informations spatiales de la scène de narration permettant ainsi de retrouver la structure de la phrase.

2.2.3 Interactions entre les mains

On a vu en fin de chapitre 1 que les deux mains peuvent être mises en relation afin d'exprimer des informations concernant les entités qu'elles représentent. Il va falloir exploiter ces informations au niveau d'un système de reconnaissance de gestes bimanuels. Mais avant cela, deux obstacles doivent être levés.

Les relations entre mains qui sont exploitables n'ont lieu que dans les cas où l'on a affaire à deux signes monomanuels simultanés. Dans le cas de signes bimanuels, l'information exprimée par la relation entre les mains est en général déjà incluse dans la signification même du signe. Or, les signes bimanuels et les signes monomanuels ont tendance a avoir le même type d'interactions entre les mains. Une des premières tâches d'un système de reconnaissance va être alors de savoir distinguer ces deux catégories de signes.
Actuellement, la plupart des études effectuées en reconnaissance de la langue des signes se sont penchées sur des phrases où n'apparaissent pas simultanément deux signes monomanuels. Très peu de systèmes font alors une réelle distinction entre signes monomanuels et signes bimanuels. Dans les rares cas qui abordent le domaine, on peut citer [Vogler et Metaxas, 1999a] où les deux mains sont considérées comme des canaux indépendants et où un algorithme examine les signes reconnus pour distinguer le cas signe bimanuel du cas signe monomanuel. Toutefois, le système ne gère pas le cas des signes monomanuels simultanés.

Une fois le problème de la distinction monomanuel/bimanuel résolu, un deuxième obstacle surgit. L'interaction entre les deux mains n'est pas effectuée de façon permanente durant la période où les deux mains représentent les entités mises en relations. Si l'on revient à l'exemple du "chat dans la voiture" de la figure 1.19 (chapitre 1, section 1.3.4), la relation a lieu entre la voiture et le

chat. En début de phrase (image 1), aucun problème, il s'agit d'un signe bimanuel qui représente la voiture, il n'y a donc aucune relation à chercher. Ensuite arrive la proforme qui représente la voiture (image 2), il n'y a qu'une seule entité dans la scène de narration, donc aucune relation à trouver . A partir de l'image 3, les deux entités *chat* et *voiture* vont enfin être présentes simultanément dans la scène. Toutefois, ce n'est qu'en toute fin de phrase (image 4) que la relation entre les entités va avoir lieu, alors qu'elles sont présentes dans la scène depuis le milieu de la phrase.
La tâche du système de reconnaissance va donc consister à retrouver, dans une phrase, ces instants précis d'interactions entre les mains. Actuellement, aucun système à notre connaissance n'effectue ce genre d'analyse sur les phrases de la langue des signes.

Si le système sait effectuer une différenciation des signes bimanuels et monomanuels, et s'il est capable de repérer les points de synchronisation entre les mains, alors il va pouvoir traiter les relations entre les mains. Il devra être capable d'analyser la disposition spatiale des mains, de comparer leur mouvement , leur orientation, etc.
Les seuls systèmes de reconnaissance qui ont un comportement proche de celui là sont les systèmes qui abordent les verbes directionnels ([Braffort, 1996b], [Sagawa et Takeuchi, 1999]). En effet, pour ces verbes, un système de reconnaissance est obligé d'effectuer des comparaisons des emplacements des entités dans l'espace de narration.

2.3 Bilan

Nous avons vu dans ce chapitre le détail du fonctionnement d'une système de reconnaissance de geste. Le processus permettant de passer du monde physique à une interprétation nécessite de nombreuses étapes, qui chacune peut être effectuée de diverses manières ou avec différentes techniques.
Suivant le contexte dans lequel on travaille et les différents problèmes qui doivent être traités, on est amené à faire des choix bien particuliers afin d'obtenir un système de reconnaissance fonctionnel et, de préférence, performant. Nous avons détaillé dans ce chapitre quelles sont les techniques qui ont été choisies ou créées afin de traiter certains problèmes rencontrés dans le domaine de la langue des signes.

Parmi les différents points durs du domaine, nous avons choisi d'étudier la reconnaissance de phrases en langue des signes intégrant des signes mono et bimanuels standards et non standards tels que les proformes. Dans la partie suivante, nous allons exposer nos contributions concernant l'élaboration d'un système de reconnaissance de gestes, contributions principalement axées sur le traitement de la problématique des gestes bimanuels, domaine qui, on l'a vu ici, a été peu ou pas abordé. Parmis ces contributions, nous donnons également notre vision sur la façon de gérer l'aspect multicanal, et cela en particulier pour la langue des signes.
Dans l'optique d'étudier la généricité de l'approche proposée dans le chapitre suivant, nous introduisons dans le chapitre 4 la réalité virtuelle qui, bien qu'elle semble être un domaine d'application complètement différent, présente des similitudes avec la langue des signes.

Deuxième partie

Elaboration d'un système de reconnaissance de gestes bimanuels

Chapitre 3

Proposition d'une architecture de reconnaissance

Dans ce chapitre, nous allons exposer quels sont nos choix techniques, logiciels et architecturaux pour établir un système destiné à interpréter des gestes bimanuels. Ce type de gestes étant présent dans la plupart des domaines utilisant la modalité gestuelle, l'architecture de reconnaissance décrite ici est volontairement générique. Suivant le domaine d'application, cette architecture peut ensuite être déclinée de diverses manières afin de traiter uniquement les aspects de ce domaine.

Dans la première section, nous exposons nos propositions architecturales visant à résoudre les problèmes propres à l'utilisation des deux mains : distinction gestes monomanuels/geste bimanuel et interprétation de relations spatio-temporelles entre gestes simultanés.
La deuxième section décrit les choix effectués pour les processus d'acquisition et de reconnaissance. Ces choix sont motivés par certains problèmes qui concernent principalement le domaine de la langue des signes : taille du vocabulaire, variété du vocabulaire, interprétation simultanée de multiples canaux et coarticulation entre gestes. Toutefois, certains de ces aspects se retrouvent parfois dans d'autres domaines ; c'est pourquoi, nous présentons ces choix dans l'optique d'une éventuelle utilisation en dehors du domaine de la langue des signes.

3.1 Traitement des gestes bimanuels et des relations spatio-temporelles

Dans cette section, nous présentons une architecture que nous avons proposée afin de résoudre les problèmes engendrés par l'utilisation simultanée des deux mains [Bossard, 2002, Bossard et al., 2003]. Cette architecture est décomposable en plusieurs parties chargées d'effectuer les différentes tâches d'un système de reconnaissance : acquisition, reconnaissance et analyse syntaxique et sémantique.

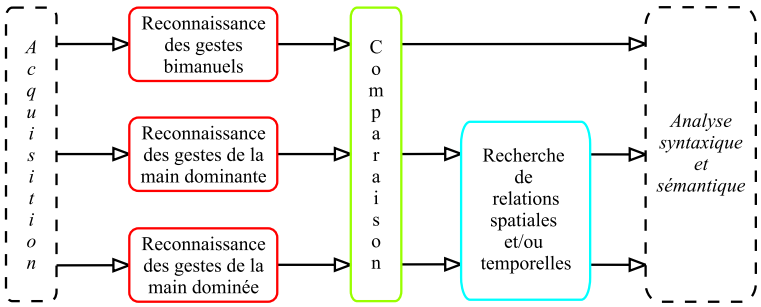

FIG. 3.1 – Architecture proposée pour traiter les gestes bimanuels et les relations entre gestes monomanuels simultanés.

Dans le cas de notre architecture, la partie effectuant la reconnaissance des gestes (modules colorés dans la figure 3.1) doit être capable d'effectuer des tâches propres au domaine des gestes bimanuels. Tout d'abord, elle doit effectuer la distinction entre gestes bimanuels et gestes monomanuels simultanés, cela afin d'effectuer une interprétation correcte. Ensuite, si deux gestes monomanuels sont produits simultanément, l'architecture doit être capable de détecter et de reconnaître les relations spatio-temporelles pouvant se produire entre ces gestes.
Nous allons maintenant voir quels sont les modules chargés d'effectuer ces deux tâches complémentaires à la reconnaissance des gestes.

3.1.1 Module de comparaison

Si l'on considère la problématique liée à l'utilisation d'une ou deux mains, trois vocabulaires gestuels sont identifiables : les gestes effectués avec les deux mains (bimanuels), ceux effectués par la main droite (généralement main dominante) et les gestes effectués par la main gauche (généralement main dominée). Chacun de ces trois vocabulaires possèdent des gestes propres à eux et donc peuvent présenter des caractéristiques différentes (i.e. on a alors besoin de primitives différentes pour chacun d'eux).
Pour interpréter correctement un geste, il va donc falloir déterminer à laquelle de ces trois catégories il appartient. Pour cela, nous avons choisi la solution suivante : pour chaque geste émis, trois modules de reconnaissance correspondant aux trois types de gestes possibles vont essayer de le reconnaître (modules en rouge dans la figure 3.2). Ensuite, le système va évaluer quel est le(s) meilleur(s) geste(s) parmi les trois reconnus ; ceux qui obtiennent "les meilleurs" scores sont alors propagés vers les modules suivants du système.

Les trois modules de reconnaissance vont travailler en parallèle à partir des données émises

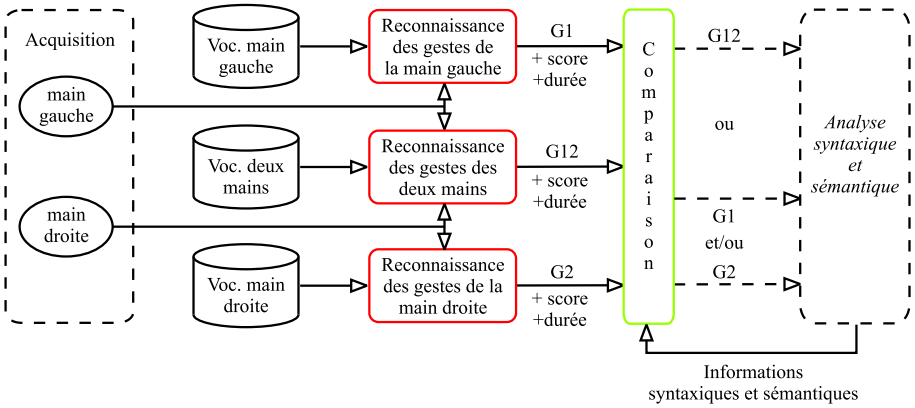

FIG. 3.2 – Les modules permettant de reconnaître et distinguer les trois types de gestes.

par la main droite et la main gauche. Mais chacun sera associé à son vocabulaire propre, c'est à dire que chacun aura effectué un apprentissage avec des gestes que n'auront pas appris les deux autres modules. En sortie, ces modules fourniront l'élément de vocabulaire reconnu, un score de reconnaissance, et la durée de cet élément (à quel instant il a commencé et à quel instant il a fini).

Parmi les gestes partageant une même période temporelle, il va falloir ensuite choisir un ou deux des éléments reconnus par les trois modules de reconnaissance. C'est un module de comparaison (module en vert dans la figure 3.2) qui va se charger de cette tâche en utilisant une fonction d'évaluation qui peut être plus ou moins sophistiquée.
Au niveau le plus simple, cela va correspondre à une fonction qui fait une évaluation "naïve" : le choix d'un signe sera fait en fonction de son score de reconnaissance et sans tenir compte du contexte. Pour des niveaux plus évolués, la fonction se base sur les signes précédemment reconnus (en utilisant des connaissances syntaxiques et sémantiques, on peut émettre des hypothèses sur le signe à reconnaître).

Suivant les résultats obtenus lors de la comparaison, on va donc avoir plusieurs situations en sortie du module de comparaison : soit il s'agit d'un geste bimanuel (G12 sur la figure 3.2), soit un unique geste monomanuel (G1 ou G2), soit deux gestes monomanuels simultanés (G1 et G2).
Dans ce dernier cas, le système va devoir rechercher la présence d'éventuels liens spatio-temporels entre les deux gestes reconnus afin de les interpréter. Nous allons maintenant voir le fonctionnement du module qui est chargé de la détection et de l'interprétation de ces relations entre mains.

3.1.2 Module de détection et d'interprétation des points de synchronisation

L'expression d'une relation spatio-temporelle a lieu durant une période temporelle précise que nous avons appelée ***point de synchronisation*** (voir section 1.3.4). Le module détaillé dans la figure 3.3 permet le traitement de ces points de synchronisation en plusieurs étapes.

Tout d'abord, il faut retrouver les plages temporelles partagées par les deux gestes. Pour cela, le module va effectuer une "resynchronisation" entre les gestes de la main gauche avec ceux de la

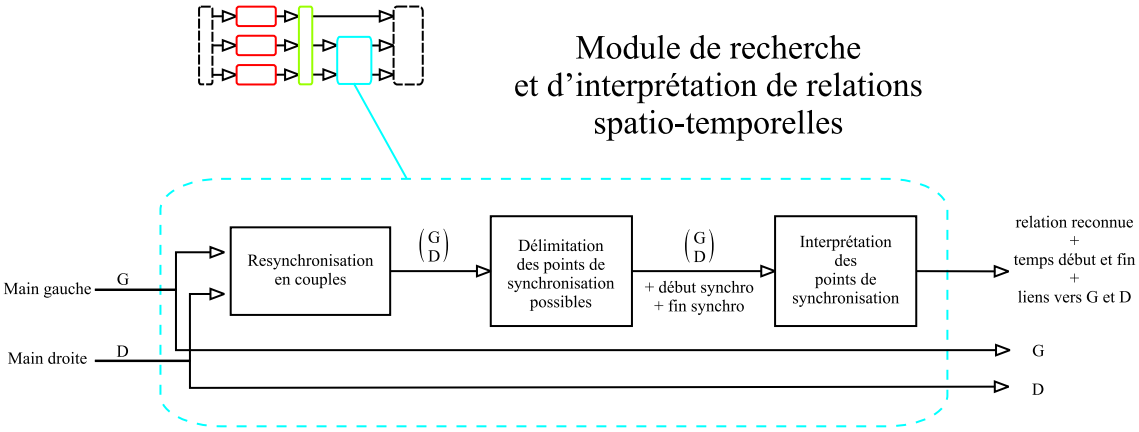

FIG. 3.3 – Le module de détection et d'interprétation de relations spatio-temporelles.

main droite. En effet, deux gestes n'ont pas forcément la même durée et ne sont donc pas reconnus au même instant par les modules de reconnaissance de la main droite et de la main gauche. Cette resynchronisation va s'effectuer en assemblant les gestes en couples selon l'instant auquel ils ont débuté et l'instant auquel ils ont fini. Par exemple, si la main gauche effectue un signe A pendant que la main droite effectue les signes B,C et D, on va obtenir les couples suivants : (A,B),(A,C),(A,D).

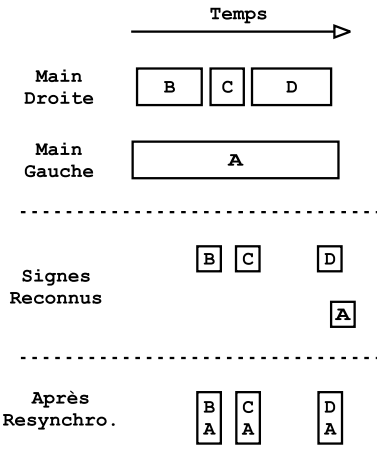

FIG. 3.4 – Création des couples main gauche/main droite.

Ensuite, le module va identifier, au sein de ces couples, les instants susceptibles d'être un point de synchronisation. Dans le cas des relations spatiales (sur lesquelles nous nous focalisons dans ce travail), lorsqu'on analyse des phrases gestuelles, on s'aperçoit que les points de synchronisation ont souvent lieu en début de couple et en fin de couple. D'autre part, ils ont bien évidemmment lieu lorsque les mains partagent un même emplacement spatial. L'idée est alors de chercher à analyser les relations spatiales en début et fin de couple en prenant en compte l'emplacement. Même si cette méthode reste assez exhaustive, elle réduit de manière importante le champ de recherche.
Pour les relations temporelles, d'autres critères peuvent être pris en compte pour l'identification

des points de synchronisation (similarité de paramètres, positions partagées, etc), mais nous n'en tiendrons pas compte ici.

Dans l'idéal, il faudrait intégrer d'autres critères pour effectuer la recherche de points de synchronisation. La nature des signes composant les couples en particulier est très importante. Par exemple, si l'on a affaire à deux classificateurs, il est très probable qu'ils soient en relation. Si un des éléments du couple est le classificateur "plat", alors il faut favoriser une recherche de relations spatiales de type "sur" ou "sous".
Ce genre de recherche demande par contre des informations de type sémantique, ce qui demanderait une communication entre ce module de recherche de points de synchronisations et un module d'analyse sémantique de plus haut niveau.

Une fois un point de synchronisation détecté en se basant sur le début, la fin du couple et sur le partage d'un emplacement, il ne reste plus au module qu'à comparer les dispositions spatiales des mains pour voir si elles sont en relation. Lorsque le module réussit à interpréter une relation spatiale, il émet alors (en plus des deux gestes) le nom de la relation spatiale accompagné d'une durée et de liens vers les gestes mis en relations.

3.2 Multiplicité des canaux d'information et coarticulation

Dans la section précédente, nous avons vu une proposition d'architecture visant à résoudre les problèmes concernant l'utilisation de gestes bimanuels et l'interprétation de relations spatio-temporelles entre les mains. Cette architecture fait intervenir, entre autre, un module d'acquisition et plusieurs modules de reconnaissance. D'autres problèmes doivent être traités au niveau de ces modules : la coarticulation et la présence d'informations dans différents canaux.
Ces deux problèmes se retrouvent en langue des signes, c'est pourquoi cette section aborde ces problèmes selon un point de vue propre à ce domaine. Toutefois, ce qui est décrit ici reste valide pour d'autres domaines.

Tout d'abord, la section présente les différentes solutions existantes qui concernent l'utilisation de multiples canaux. Ces solutions s'impliquent à différents niveaux du processus de reconnaissance : modélisation du vocabulaire, acquisition et fusion de données lors de la reconnaissance. Ensuite, nous parlerons brièvement du problème de la coarticulation avant de conclure la section sur une proposition de structure pour le module de reconnaissance. Cette structure s'inspire des approches qui nous semblent convenir le mieux à notre cas de figure.

3.2.1 Modélisation du vocabulaire

Nous visons comme champ d'expérimentation des phrases de la langue des signes utilisant des transferts situationnels, car ce type de phrase présente de nombreuses relations spatiales et/ou temporelles entre les deux mains. Dans les transferts situationnels, on peut voir apparaître tout type de vocabulaire : signes standards, spécificateurs, proformes, verbes directionnels... Ici, on va se restreindre uniquement au vocabulaire standard et aux proformes.
La reconnaissance du vocabulaire standard ne pose pas de grand problème sauf si l'on souhaite pouvoir reconnaître un grand nombre de signes. Par contre, interpréter les proformes ne va pas pou-

voir se faire directement en utilisant les techniques traditionnelles. On répertorie une quarantaine de configurations possibles pour les proformes [Cuxac, 2000], mais pour les autres paramètres il n'y a aucune valeur fixée. L'emplacement, l'orientation et le mouvement dépendent du contexte de l'énoncé et des choix du signeur. On ne peut donc considérer les signes de manière globale comme cela est effectué dans la plupart des systèmes de reconnaissance actuels qui traitent en majorité uniquement les signes standards.

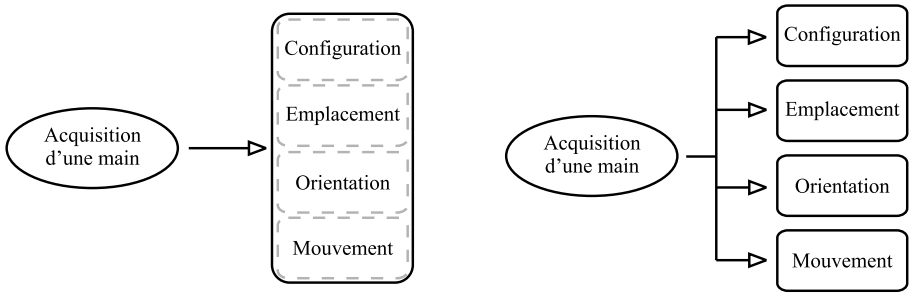

FIG. 3.5 – Traitement des paramètres en bloc (à gauche) ou de manière décomposée (à droite).

La solution est de considérer les signes sous la forme des quatre paramètres (voir figure 3.5) et d'effectuer une reconnaissance de ces quatre paramètres afin de les utiliser lors de l'étape d'analyse sémantique pour en extraire les informations spatiales et temporelles qu'ils véhiculent.

3.2.2 Acquisition des données

Le choix des techniques d'acquisition doit être fait selon les données que l'on souhaite acquérir et exploiter. Dans notre cas, on veut obtenir les quatre paramètres que sont la configuration, l'orientation, le mouvement et l'emplacement. D'autre part, du fait que l'on désire étudier les relations spatiales entre les mains, on a besoin d'une bonne précision au niveau du positionnement des mains.
Les méthodes d'acquisition basées sur des caméras ne nous conviennent pas car elles ne permettent pas une acquisition fiable de la configuration des mains. De plus, lorsque celles-ci interagissent, des problèmes d'occultation ont lieu, ce qui rend difficile l'acquisition de tous les paramètres (typiquement lorsqu'une main se situe à "l'intérieur" de l'autre pour exprimer une relation "dans"). Les méthodes de reconstruction en trois dimensions par vision permettraient de régler en partie ces problèmes, mais actuellement elles ne sont pas encore opérationnelles pour la main dans un contexte naturel.

Nous avons donc choisi de nous orienter vers une solution fortement intrusive basée sur des gants numériques et des capteurs de position/orientation, même si cela implique des contraintes physiques au niveau du signeur (câbles reliant les capteurs et les gants à des boîtiers électroniques). L'utilisation de gants numériques permet d'avoir directement la configuration de la main, et les capteurs de position/orientation permettent de remonter très facilement à l'emplacement, à l'orientation et au mouvement. Du fait qu'on utilise directement des capteurs au niveau du corps, on évite les problèmes d'occultation rencontrés par les caméras et on permet une acquisition fiable et précise des données.

3.2.3 Reconnaissance simultanée de plusieurs canaux d'information

Les signes correspondent à la conjonction de plusieurs sources d'information : les deux mains sont deux canaux qui eux mêmes peuvent être décomposés en quatre canaux correspondant aux paramètres. Effectuer la reconnaissance de signes nécessite donc d'avoir un système capable de traiter plusieurs canaux de données, et d'être capable d'effectuer la fusion de ces différents canaux pour reconnaître ces signes. Pour cela, différentes stratégies existent (figure 3.6) :

- soit l'on effectue la fusion des différents canaux au moment où l'on reçoit les données puis l'on effectue la reconnaissance du signe (**fusion précoce**),
- soit la fusion est faite pendant la reconnaissance des différents canaux (**fusion intermédiaire**),
- soit enfin on effectue une fusion après avoir traité des différents canaux (**fusion tardive**).

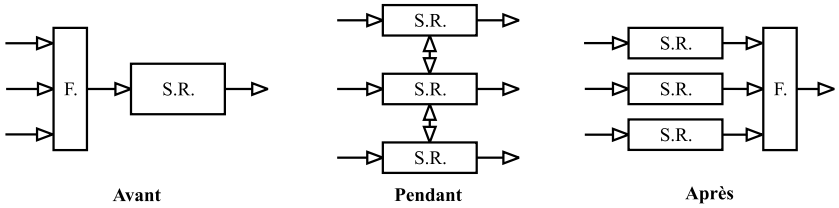

FIG. 3.6 – Les différentes stratégies de fusion de canaux de données.

Fusion précoce

La première méthode où tous les canaux sont considérés comme un unique canal permet de se ramener à un cas de reconnaissance standard. Si cela simplifie grandement la tâche de reconnaissance, ce n'est pas tout a fait adapté à la langue des signes.
En effet, la taille du vocabulaire est un des problèmes auxquels est confronté un système de reconnaissance de la langue des signes. Or, considérer les signes de manière globale nécessite d'utiliser des techniques de clustering [1] au niveau du système de reconnaissance ([Bauer et Kraiss, 2001], [Fang et al., 2004]). Par contre, on a vu que la décomposition des signes en plusieurs canaux séparés peut permettre de résoudre ce problème ([Vogler et Metaxas, 1999b]).
De plus, effectuer une fusion avant l'étape de reconnaissance ne permet pas d'exploiter les informations spatiales et temporelles contenues au niveau de chaque paramètre.

Fusion intermédiaire

La deuxième méthode effectue une fusion au cours de la reconnaissance. Dans ce cas, différentes manières d'effectuer la fusion peuvent être utilisées. Par exemple, les modèles de Markov cachés couplés [Brand, 1997] (figure 3.7), où des modèles de Markov cachés sont reliés entre eux, permettent de modéliser des processus interagissant entre eux (la reconnaissance d'un symbole au niveau d'un canal dépend de ce qui a été reconnu sur les autres canaux). Avec ce type de système,

[1] Voir la section 2.2.1 sur le problème de taille du vocabulaire.

on pense tout de suite à la modélisation des deux canaux liés que sont les deux mains ; mais le fait que les mains ne sont pas tout le temps en relation et peuvent être complètement indépendantes pose problème.

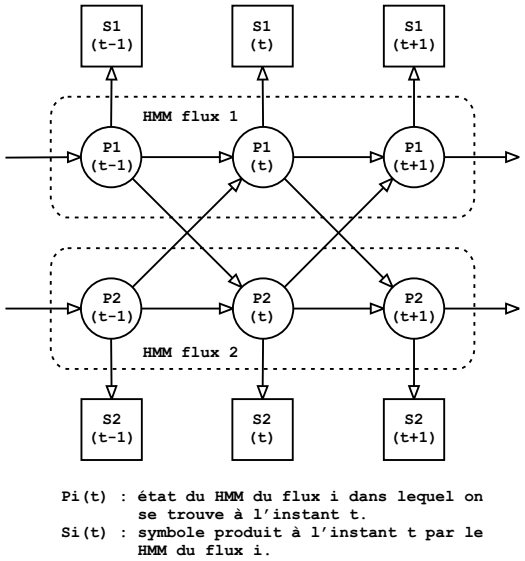

FIG. 3.7 – Modèle de Markov caché couplé.

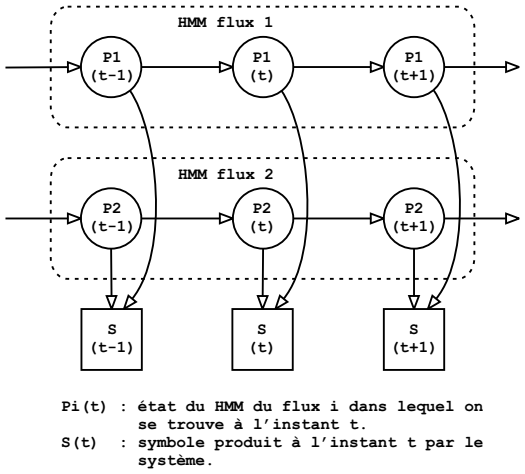

FIG. 3.8 – Modèle de Markov caché factorisé.

On peut citer aussi les modèles de Markov cachés factoriels [Ghahramani et Jordan, 1996] qui permettent plus d'indépendance entre les différents modèles de Markov représentant les canaux , et qui effectuent un couplage des sorties des états des modèles de Markov (figure 3.8). Cela implique d'avoir des processus relativement similaires pour avoir des chaînes de Markov cachées similaires (sinon comment coupler les sorties ?).

Bilan Ces méthodes de fusion en cours de reconnaissance imposent de fortes contraintes de synchronisation ou de similarité au niveau des canaux qu'elles modélisent. Or les différents canaux que l'on considère ici - configuration, emplacement, orientation, position, main gauche et main droite - ne présentent pas ces caractéristiques. Les deux mains sont soit en relation, soit complètement indépendantes. Quant aux quatre paramètres, ils sont la plupart du temps complètement indépendants et asynchrones. De plus, certains d'entre eux ne se prêtent pas à une modélisation avec des modèles de Markov cachés.
D'autre part, ces deux méthodes demandent souvent des modélisations et des apprentissages assez complexes.

Fusion tardive

La troisième méthode consiste à considérer les différents canaux de manière complètement indépendante lors de la reconnaissance et à effectuer ensuite une fusion des données issues de ces canaux. Cela permet d'éviter les problèmes induits par les deux premières méthodes : on va pouvoir garder les quatre paramètres intacts, traiter des canaux asynchrones et/ou hétérogènes, et éviter le problème de complexité de taille du vocabulaire.
La fusion va être plus ou moins complexe suivant ce que l'on a considéré comme canaux à reconnaître. Dans [Vogler et Metaxas, 1999a], les auteurs modélisent les canaux correspondant aux deux mains à l'aide de modèles de Markov cachés fonctionnant de manière indépendante et parallèle. Il va falloir gérer à la sortie de ces canaux le fait que certains signes sont bimanuels alors que d'autres ne le sont pas. La transmission des données dans leur système se fait sous forme de tokens[2] [Young et al., 1989], et ces tokens vont devoir être fusionnés ou non suivant que l'on a reconnu un geste bimanuel ou monomanuel. Cela est effectué au niveau de noeuds spéciaux qui rassemblent les sorties des canaux, et où un algorithme se charge d'examiner ce qui a été reconnu et effectue les fusions nécessaires (figure 3.9).

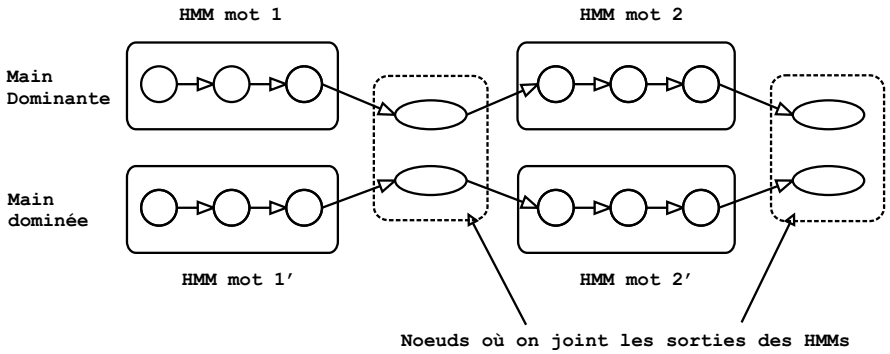

FIG. 3.9 – Modèles de Markov cachés parallèles.

Lorsque l'on effectue une reconnaissance au niveau des paramètres, la fusion doit être capable d'étiqueter les données issues des canaux avec le nom du signe. Dans le cas le plus simple, on

[2] Il s'agit de jetons qui circulent au travers d'un réseau de modèles de Markov caché. Ces jetons stockent au fur et à mesure le chemin suivi, les scores obtenus, etc.

peut se contenter de comparer la combinaison des différents paramètres avec une base de données lexicales (comme effectué dans [Liang et Ouhyoung, 1998]). Mais on peut très bien concevoir un système plus évolué capable de gérer les cas où l'on a une évolution des paramètres au cours du signe.
Ainsi, si ce troisième type de méthode de fusion permet plus de flexibilité pour gérer les différents canaux de la langue des signes, il implique des étapes de fusion assez lourdes que l'on peut presque assimiler à une deuxième étape de reconnaissance.

Nous avons choisi d'adopter le dernier type de fusion et plus particulièrement l'approche utilisée par Liang et Ouhyoung où les quatre paramètres sont reconnus par des systèmes de reconnaissance distincts. Cela nous permet de résoudre le problème de la complexité de la taille du vocabulaire et de conserver intacts les quatre paramètres en vue d'une interprétation syntaxique ou sémantique.
Les modèles de Markov cachés parallèles n'ont pas été choisis pour effectuer la reconnaissance des quatre paramètres, car ces modèles ne sont pas très adaptés pour certains de ces paramètres. En particulier, l'orientation n'a pas un vocabulaire vraiment défini, surtout lorsque l'on sort du domaine du vocabulaire standard. En effet, il faudrait être capable de reconnaître la vingtaine d'orientations principales (voir [Braffort, 1996b] pour le recensement des différentes orientations) et de pouvoir reconnaître toute nouvelle orientation comme appartenant à une classe "divers". Or les modèles de Markov cachés ne permettent pas ce genre de mécanisme. C'est pourquoi, nous préférons plutôt l'utilisation de types de systèmes de reconnaissance différents suivant les paramètres, dans le cas où le vocabulaire laisse trop de variabilité au niveau de ces paramètres.

3.2.4 La coarticulation

La coarticulation est un phénomène qu'il faut prendre en compte si l'on veut pouvoir effectuer la reconnaissance continue de signes. La solution la plus couramment utilisée pour résoudre ce problème consiste à utiliser les modèles de Markov cachés. L'idéal serait donc d'avoir quatre modèles de Markov cachés en parallèle pour éviter ce problème. Or on a vu précédemment que ces modèles ne peuvent pas toujours être utilisés pour chaque paramètre, en particulier l'orientation et l'emplacement.
Pour ces cas spéciaux, de nombreuses méthodes de segmentation sont possibles. Nous avons choisi de baser notre segmentation sur une approche de type 'hold and movement'[3], c'est à dire que les phases dynamiques séparant un signe d'un autre vont être utilisées pour effectuer la segmentation. Cette approche n'est clairement pas idéale du fait qu'elle ne permet pas de reconnaître des signes où les paramètres sont dynamiques, mais cela ne pose pas trop de problème lorsqu'au moins un des paramètres est statique et peut servir de référence à la segmentation.

3.2.5 Module de reconnaissance des signes

Notre choix d'une fusion tardive amène à effectuer la reconnaissance d'un signe sous la forme des quatre différents paramètres. Cela permet de réduire la complexité et d'être capable de modéliser les proformes et les signes standards.

[3]La modélisation de la langue des signes effectuée par Liddel et Johnson considère les signes comme étant décomposables en deux parties. Une partie ***hold*** qui correspond au signe lui même, et une partie ***movement*** qui correspond à la phase de transition vers le signe suivant.

On peut noter que la décomposition des signes en paramètres permet également de reconnaître les verbes directionnels et les spécificateurs de forme et de taille. Par contre, pour ces derniers, il est très difficile d'effectuer un étiquetage au niveau de la fusion, du fait que leurs paramètres sont complètement libres. L'interprétation de tels signes ne peut alors que s'effectuer à un niveau d'analyse sémantique. Mais une fois les différents paramètres reconnus, on peut les préserver afin de les transmettre, au travers du système, à une étape d'analyse ultérieure.

La figure 3.10 nous montre l'architecture d'un module de reconnaissance. On y trouve les quatre modules (pour les signes à deux mains on aura évidemment huit modules) qui effectuent la reconnaissance des paramètres en parallèle. La nature des systèmes de reconnaissance peut être choisie en fonction du vocabulaire, du degré de coarticulation et du domaine d'application. On peut par exemple utiliser des modèles de Markov cachés ou des modèles statistiques.

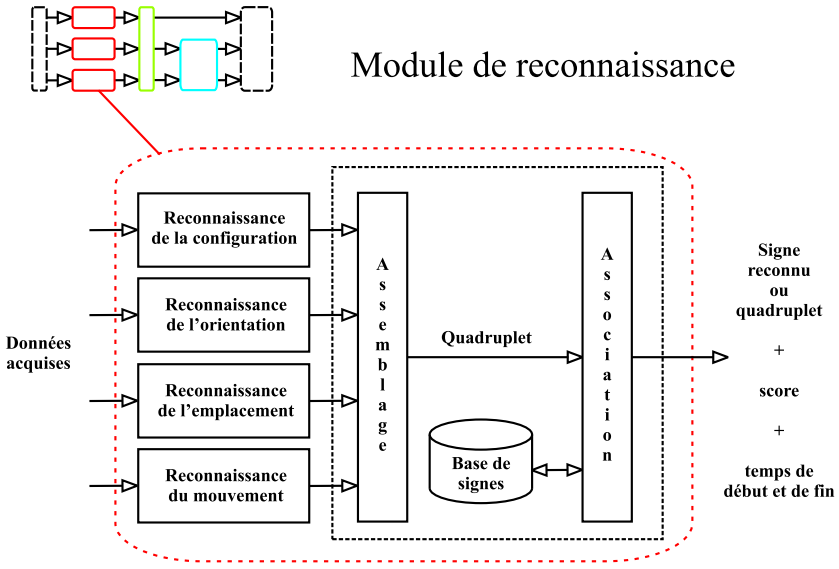

FIG. 3.10 – Structure d'un module de reconnaissance de gestes monomanuels au sein de l'architecture.

Le module de fusion qui reçoit les résultats de la reconnaissance des paramètres va devoir effectuer diverses tâches : il va tout d'abord construire des quadruplets à partir des symboles reçus depuis les quatre modules, il va ensuite chercher à associer chaque quadruplet à un signe (cela peut être effectué soit en cherchant une correspondance dans une base de données de quadruplets, soit par des méthodes stochastiques), enfin il va transmettre le signe reconnu avec un score de reconnaissance basé sur celui des paramètres et la qualité de la correspondance signe-quadruplet. Dans le cas où aucune correspondance n'est trouvée, on est probablement face à un signe non appris par le système ou un signe du vocabulaire non standard. Le quadruplet est alors transmis pour une analyse ultérieure sans qu'il ait été étiqueté.
La segmentation est effectuée lors de l'étape d'assemblage en se basant sur la variation des paramètres.

3.3 Bilan

Nous avons exposé dans ce chapitre la maquette de l'architecture que nous proposons afin de traiter des gestes bimanuels. Cette architecture se veut fortement modulaire afin de pouvoir être adaptée à différents contextes où l'on est confronté aux gestes bimanuels mais aussi à d'autres problématiques. Nous avons vu, par exemple, que diverses techniques peuvent être envisagées afin de traiter l'aspect multicanal des gestes.

Afin d'évaluer la généricité de l'approche proposée dans ce chapitre, nous avons étudié son utilisation dans le contexte de l'interaction gestuelle bimanuelle en réalité virtuelle. C'est ce domaine qui va être présenté dans le chapitre suivant.
Les expérimentations menées autour de l'architecture proposée, que ce soit pour la réalité virtuelle ou la langue des signes, sont détaillées dans la troisième et dernière partie de ce document.

Chapitre 4

Réalité virtuelle et langue des signes

Tout ce qui a été exposé dans les chapitres précédents a concerné la langue des signes, les interactions entre les deux mains, ou encore la reconnaissance de gestes. Ce chapitre a pour but d'introduire le domaine de la réalité virtuelle pour lequel nous avons étudié l'utilisation de l'architecture décrite dans le chapitre précédent. Nous établirons ainsi, dans cette partie, les points communs qui existent entre la langue des signes et la réalité virtuelle. Cela permettra d'expliciter le choix de notre architecture pour certains types de gestes en réalité virtuelle.

Dans la première partie de ce chapitre, nous allons présenter la réalité virtuelle : quel est ce domaine, comment plonger l'utilisateur dans un monde virtuel et comment lui permettre d'interagir avec ce monde. Ensuite, en deuxième section, nous exposons le domaine du geste en réalité virtuelle : de quelle manière on peut modéliser les gestes, quels types existent et comment ces gestes peuvent s'enchaîner. Enfin, dans la troisième section nous détaillons les différents points communs entre la réalité virtuelle et la langue des signes.

4.1 La réalité virtuelle

La notion de réalité virtuelle est apparue dans les années 80 (le terme "virtual reality" a été introduit par Jaron Lanier vers 1985) et est issue de différents domaines informatiques tels que le rendu d'images, l'interaction homme-machine, la simulation, etc. De multiples discussions sur la définition de la réalité virtuelle ont été menées depuis son apparition ; ici, nous nous fixerons à la définition suivante. La ***réalité virtuelle*** effectue l'immersion d'un utilisateur via des interactions sensorielles et/ou motrices dans un monde imaginaire ou un monde simulant la réalité. Ce monde virtuel est parfois appelé ***scène virtuelle*** dans laquelle sont placées les différentes entités peuplant le monde.
On peut avoir besoin éventuellement de mixer la réalité virtuelle avec le monde réel, c'est ce que l'on appelle la ***réalité augmentée***. Ce mélange peut se faire de diverses manières en impliquant plus ou moins les deux mondes, dans [Fuchs et al., 2001] les auteurs ne recensent pas moins de cinq manières différentes. Dans ce travail, nous ne nous intéressons pas à ces aspects de la réalité virtuelle où la réalité intervient. Nous nous sommes restreints aux cas où intervient uniquement un monde virtuel.

Pour effectuer une immersion de l'utilisateur dans ce monde virtuel, de nombreux types de dispositifs techniques ont été développés pour restituer différentes sensations (visuelle, sonore, tactile...), ou pour permettre à l'utilisateur d'agir sur le monde via différentes modalités (vocale, gestuelle, haptique ...). Nous n'exposerons pas ici toutes les techniques de la réalité virtuelle ; le domaine est vaste et dépasse le cadre de cette thèse. Nous allons présenter uniquement certains aspects de la réalité virtuelle qui permettent de poser quelques bases et de situer le cadre de ce travail qui a été intégré au projet VENISE du LIMSI-CNRS [Bourdot, 2001].

4.1.1 Nature du monde virtuel

Le monde virtuel dans lequel est plongé l'utilisateur peut être de natures fort différentes. Il peut correspondre à une simulation parfaite ou partielle du monde réel, à une simulation où l'on modifie certains aspects du monde réel (changement d'échelle spatiale, modification du cours du temps, etc), ou encore à un monde complètement artificiel.
La simulation du monde réel joue sur plusieurs paramètres : tout d'abord les sensations que l'on ressent doivent être semblables à celles que l'on aurait dans le monde réel (vision, audition, touché, etc). Ces différentes sensations sont restituées via des périphériques spécifiques. D'autre part, le monde simulé doit se "comporter" de manière similaire au monde réel. C'est à dire que l'on va avoir les mêmes propriétés physiques : si l'utilisateur lâche un objet, ce dernier tombe ; si on avance vers un mur, on ne peut le traverser ; etc. Tout cela peut être effectué par des programmes simulant numériquement les propriétés physiques. L'un des plus connus est ce que l'on appelle le ***moteur de collision*** qui permet d'éviter par exemple à un utilisateur de traverser un mur.

Tous ces différents paramètres interviennent dans la création d'un monde virtuel, mais on n'a pas besoin forcément de tous les utiliser pour immerger l'utilisateur dans le monde. Le choix des paramètres à utiliser va donc se faire en fonction des besoins du monde virtuel que l'on veut créer.

Dans le cas du projet VENISE, différents types de mondes virtuels sont utilisés suivant le domaine d'expérimentation. Certains permettent la restitution de phénomènes correspondant à des expérimentations scientifiques (visualisation d'écoulements de fluides ou visualisation de brins d'ADN). D'autres restituent partiellement le monde réel (scènes où sont disposés différents objets, ou scènes représentant des grands espaces). Dans notre cas, où l'on se penche sur les interactions bimanuelles, ce sont principalement les scènes composées de différents objets qui nous intéressent. Pour faire percevoir ce monde à l'utilisateur, de nombreuses modalités sont disponibles, mais seule la modalité visuelle sera utilisée dans le cadre de cette thèse.

4.1.2 Les dispositifs visuels

La modalité visuelle est présente dans la quasi totalité des applications de réalité virtuelle. Les techniques qui ont été développées visent à couvrir le champ visuel de l'utilisateur et/ou à lui permettre une vision en relief dans le but d'effectuer une immersion de l'utilisateur dans le monde virtuel.

Comme dispositifs d'affichage, on recense trois principales catégories de matériel : les moniteurs standards, les visiocasques et les écrans de grande taille.
Dans le premier cas, on rajoute en général un dispositif permettant d'avoir une perception en relief du monde virtuel. Malgré cela, l'immersion de l'utilisateur reste partielle du fait qu'un moniteur est fixe et a une taille limitée.
Les ***visiocasques*** résolvent ces problèmes de largeur de champ de vision et de point de vue. Ces dispositifs consistent à fixer devant les yeux deux écrans (en général des écrans à cristaux liquides) qui permettent d'avoir une vision en relief et couvrent le champ visuel aisément. De plus, le fait d'utiliser un casque pour supporter les écrans autorise l'utilisateur à pouvoir changer son point de vue en déplaçant sa tête. Par contre, les visiocasques posent des problèmes technologiques au niveau des écrans, et sont dédiés à un unique utilisateur.
Le troisième type de dispositif consiste à utiliser des grands écrans sur lesquels sont rétroprojetées les images. Il n'a pas les inconvénients des moniteurs et permet la présence simultanée de plusieurs utilisateurs dans le dispositif. La couverture du champ visuel d'un utilisateur se fait en jouant sur la forme, la taille et le nombre d'écrans utilisés dans le dispositif. Ces choix sont effectués en fonction des tâches que l'on désire pouvoir réaliser dans le dispositif. Par exemple un dispositif de type ***WorkBench*** [Kruger et al., 1995], où l'on souhaite avoir un "bureau virtuel", est réduit à un seul écran (image du haut dans la figure 4.1). Tandis que dans un dispositif de type ***CAVE*** [Cruz-Neira et al., 1993], où l'on souhaite être complètement immergé dans le monde virtuel, l'utilisateur est complètement entouré par les écrans (image du bas dans la figure 4.1).

Comme pour les moniteurs, ces dispositifs peuvent être améliorés avec un système permettant de donner la perception du relief à un utilisateur. C'est le principe de ***stéréoscopie*** qui est utilisé pour effectuer cette restitution de la troisième dimension. Il consiste à restituer deux images légèrement décalées pour chaque œil, on a ainsi l'impression de voir les objets en trois dimensions.
On affiche donc les deux images différentes sur le(s) écran(s). Ensuite, diverses techniques, utilisant des lunettes, permettent d'effectuer la séparation des images au niveau des yeux.

Dans le cas du projet VENISE, le dispositif d'affichage utilisé consiste en deux grands écrans

FIG. 4.1 – En haut un dispositif de type WorkBench (système Baron™ de chez BARCO), en bas un dispositif de type CAVE (Extrait de l'University Record du 5 nov. 97, Université du Michigan).

verticaux disposés en angle droit. Ce dispositif est doté d'un système stéréoscopique permettant d'avoir un rendu en relief du monde virtuel. Dans un tel dispositif, l'utilisateur se retrouve debout face aux deux écrans et plongé dans un monde en trois dimensions (figure 4.2). Il devient alors très difficile d'utiliser les moyens habituels d'interactions comme le clavier ou la souris. Pour pallier à ce problème, de nombreuses recherches sont effectuées dans le domaine de la réalité virtuelle afin de trouver de nouvelles manières d'interagir.

4.1.3 Interactions en réalité virtuelle

En réalité virtuelle, l'utilisateur peut se retrouver complètement immergé dans un monde qui lui restitue diverses sensations comme la perception visuelle en trois dimensions, la perception tactile, la perception auditive en trois dimensions, etc. L'utilisateur va vouloir interagir avec ce monde pour y effectuer différents types d'actions : navigation, créations d'objets, manipulation d'objets, etc.
Nous allons distinguer ici deux types d'interactions : d'une part celles qui nécessitent l'emploi de périphériques spécifiques, d'autre part les méthodes que nous utilisons au quotidien pour interagir avec le monde réel (par la suite, nous qualifierons ce type d'interactions de ***naturelles***).

FIG. 4.2 – Le dispositif utilisé dans le cadre de ce travail. La personne que l'on voit ici effectue un geste de *saisie* sur la théière. (LIMSI-CNRS)

Différents périphériques 3D

Lorsque l'on passe d'une station dotée d'un moniteur à un environnement de réalité virtuelle, le principal changement consiste à l'introduction de la troisième dimension. C'est pourquoi de nombreux périphériques comme les souris ou les joysticks ont pu facilement être adaptés dans le domaine de la réalité virtuelle en utilisant des capteurs de position/orientation. Par exemple, le stylet disponible avec les systèmes de la société POLHEMUS permet d'avoir l'équivalent d'un stylet de tablette graphique d'une station de travail traditionnelle (figure 4.3). Pour pouvoir effectuer à la fois des tâches de pointage et de navigation certains périphériques comme le ***Wand*** combinent à la fois les mécanismes d'un stylet et d'un joystick (figure 4.3).
Tous ces périphériques permettent uniquement d'effectuer des tâches de pointage, de navigation, et de manipulation dans l'espace.
On peut noter que tous les périphériques qui permettent de capter une position et/ou une orientation dans l'espace rendent possible la capture de gestes sous la forme de trajectoires. Les fonctionnalités de ces périphériques peuvent alors être enrichies par un vocabulaire gestuel basé sur le mouvement.

Interactions naturelles

Pour effectuer différentes tâches dans le monde réel, nous utilisons de nombreuses modalités qui peuvent être combinées. Interagir de manière naturelle dans un monde virtuel revient à utiliser ces modalités de manière similaire ou identique. L'utilisation de ces modalités naturelles permet à l'utilisateur un grand nombre d'interactions parfois très évoluées tout en gardant une certaine simplicité d'exécution pour l'utilisateur. D'autre part, elles évitent à l'utilisateur d'avoir à garder constamment un ou plusieurs périphériques sur lui pour pouvoir interagir avec le monde virtuel.

FIG. 4.3 – A gauche un stylet 3D (POLHEMUS), à droite un Wand (ASCENSION TECH.)

Pour naviguer on va utiliser les mouvements du corps. On peut par exemple utiliser des dispositifs de type tapis roulant pour connaître le déplacement de l'utilisateur dans le monde virtuel. Ou encore utiliser la position relative de l'utilisateur par rapport à un point fixe pour indiquer un déplacement dans le monde virtuel. Par exemple dans [Bourdot et Touraine, 2002b], le système utilise la position et l'orientation de la tête par rapport à un point neutre pour naviguer. Lorsque l'utilisateur quitte ce point neutre, il commence à se déplacer dans la scène, le vecteur entre le point neutre et la position courante donnant la direction et la vitesse du déplacement.
La voix peut servir à envoyer des commandes au système qui gère l'environnement virtuel. Elle peut également être combinée avec d'autres modalités afin de mélanger certains types d'interactions. On peut par exemple utiliser conjointement la voix et le geste pour simultanément pointer un élément du monde virtuel et indiquer une action à réaliser, comme par exemple "rend ceci plus grand" [Frohlich et Wachsmuth, 1997].
La modalité gestuelle sert bien évidement à manipuler les objets composant la scène. Un problème se pose en réalité virtuelle lorsque l'on veut manipuler ces objets avec les mains : ils n'ont pas de consistance physique. Cela peut entraîner des approximations lorsque l'utilisateur manipule les objets virtuels et perturber sa perception du monde virtuel. Pour résoudre ce problème, il existe les périphériques dit ***haptiques*** qui restituent la perception de *consistance* à l'utilisateur.

La manipulation d'objets n'est pas le seul type d'interaction possible. De nombreux autres types de gestes permettent d'effectuer d'autres actions. Nous allons maintenant détailler, dans la section qui suit, ces différents gestes qui permettent de nombreuses interactions en réalité virtuelle.

4.2 Les gestes en réalité virtuelle

Les interactions gestuelles en réalité virtuelle couvrent de multiples facettes du geste. L'utilisateur peut aussi bien faire intervenir les fonctions épistémique et ergotique (perception et action) que la fonction sémiotique (expression d'informations). En effet, il a besoin de pouvoir interagir directement avec le monde virtuel via les fonctionnalités sensori-motrices de la main, mais aussi de pouvoir interagir avec l'application qui gère ce monde virtuel en produisant des gestes porteurs d'informations.

On va devoir donc gérer en réalité virtuelle des gestes de nature très différente. Suivant l'aspect sur lequel se focalise principalement une application (sensori-moteur ou sémantique), on ne va pas

aborder la modélisation des gestes de la même manière. Ce sont ces différentes approches que nous allons maintenant détailler.

4.2.1 Différentes approches du geste

Modèle physique

Lorsque l'on n'utilise que la fonction épistémique et/ou la fonction ergotique, certaines approches vont modéliser le geste suivant un point de vue physique. C'est à dire que l'on ne s'intéresse pas au geste lui même mais à la manière dont il interagit avec le monde physique.
Cela est particulièrement pertinent avec la fonction épistémique lorsque l'on désire employer des périphériques haptiques. On peut alors avoir besoin d'un modèle physiologique où sont modélisés le squelette et la musculature du bras et de la main afin d'exercer les bonnes forces aux bons endroits.
Pour effectuer des manipulations d'objets (fonction ergotique), l'utilisation d'un modèle physiologique est un peu moins pertinent. Toutefois, si l'on est dans un système qui modélise le monde virtuel de manière physique (collisions, forces exercées, etc), on peut utiliser un modèle physique de la main pour détecter des actions comme la préhension des objets de la scène.
Ce type de modèle reste rare car le domaine de l'haptique est encore en développement, et la modélisation d'un monde physique nécessite des calculs assez complexes. D'autre part, on ne peut utiliser que des gestes permettant une interaction directe avec le monde virtuel (l'utilisateur est en *contact* avec les objets : ils ne sont pas situés plus loin que la longueur de son bras), ce modèle ne permet pas l'usage de gestes utilisant la fonctionnalité sémiotique.

Modèle sous forme de mouvements

On a vu précédemment que différents périphériques (pointeurs, wands, etc) existent pour effectuer la manipulation des objets dans une scène virtuelle. Comme ces périphériques disposent d'un capteur de position/orientation, on peut capter les gestes sous la forme de trajectoires. Cela permet d'utiliser la fonction sémiotique du geste afin d'enrichir les interactions de base de ces périphériques. L'utilisation d'un tel modèle est courant lorsque l'on ne dispose pas de périphériques permettant l'acquisition de la configuration de la main.
La modélisation de gestes sous la forme de trajectoire est souvent rencontrée dans les environnements immersifs de type "Workbench". On peut alors utiliser la direction et la trajectoire du mouvement pour différencier les actions à réaliser. Par exemple, dans [Laviola, 1999], où l'on utilise des gestes bimanuels, deux mouvements linéaires opposés indiquent une opération de changement de taille, deux mouvements circulaires correspondent à une rotation et deux mouvements linéaires identiques effectuent une translation. Dans [Bimber, 1999], l'auteur utilise conjointement les mouvements et le regard pour par exemple simuler la planification de matchs de basket (figure 4.4). Il utilise les mouvements de la main pour constituer un ensemble d'actions que l'on voudrait faire effectuer au système. Par exemple, on sélectionne un joueur en faisant un cercle autour de lui, on indique le déplacement d'un joueur en effectuant un mouvement parallèle au plan simulant le terrain, etc.

Par rapport au modèle physique, ce modèle ne permet pas de gérer des interactions directes entre l'utilisateur et le monde virtuel. Toutefois, par l'utilisation de la fonctionnalité sémiotique, il permet d'interagir avec ce monde via des commandes gestuelles qui indiquent à l'application qu'il

FIG. 4.4 – Exemples de gestes basés sur le mouvement . (Extrait de [Bimber, 1999])

faut effectuer une sélection, un déplacement, une rotation, etc.

Modèle linguistique

Enfin, lorsque l'on considère des fonctionnalités sémiotiques assez évoluées, ou que l'on considère des gestes sensori-moteurs sans avoir de modèle physique du monde, on emploie en général des modèles proches ou provenant du monde linguistique. En effet, les modèles linguistiques permettent de modéliser toute l'expressivité du geste pour pouvoir en exploiter le sens lors des phases de reconnaissance et d'interprétation. D'autre part, certains modèles employés en langue des signes, comme les modèles paramétriques (Stockoe, Battison, etc), sont proches d'une modélisation physique des articulations de la main et du bras (la configuration correspond aux articulations de la main, l'orientation dépend principalement du poignet, etc). D'autres modèles comme celui donné dans [Boutet, 2001], sont complètement collés à la modélisation articulaire de la main et du bras. On voit donc que les modèles linguistiques permettent de modéliser aussi bien des gestes sensori-moteurs que des gestes sémantiques.

Malgré cette puissance de modélisation, l'emploi de modèles linguistiques reste toutefois rare dans le domaine de la réalité virtuelle du fait qu'ils sont un peu trop complexes pour certains besoins simples de la réalité virtuelle. Mais au fur et à mesure que les recherches sur les interactions gestuelles en réalité virtuelle explorent de nouveaux aspects sémantiques, on tend à s'en rapprocher peu à peu.

4.2.2 Les différentes catégories de gestes

Différents types d'actions peuvent être réalisés à l'aide de gestes dans un environnement virtuel : manipulation d'objets, description d'objets, création ou modification d'objets, déclenchement d'actions, navigation, etc. Certaines de ces actions peuvent être effectuées avec des gestes naturels, d'autres nécessitent la création d'un vocabulaire spécifique.

Manipulation directe

Lorsque l'utilisateur veut manipuler un objet, deux cas se produisent : soit l'objet est à portée de main, soit il se trouve à une distance éloignée qui nécessite un déplacement de l'utilisateur. Dans le cas où l'on se trouve à proximité, il suffit de manipuler l'objet virtuel comme un objet réel. On le saisit, on le déplace ou on l'oriente, puis on le relâche. C'est ce que l'on appelle la ***manipulation directe***. Pour effectuer ce type d'action, on va utiliser un geste de manipulation qui est composé de trois phases : fermeture des doigts sur un objet, tenue de la configuration pendant le temps de manipulation, puis écartement des doigts pour relâcher l'objet. C'est un geste très simple qui permet une interaction directe entre l'utilisateur et le monde virtuel. Dans la figure 4.2 donnée précemment, on peut voir un utilisateur manipuler librement une théière qu'il tient dans la main.

Dans le deuxième cas, avant de pouvoir manipuler l'objet à distance (par geste ou un autre moyen), il va falloir le désigner. Pour effectuer cette tâche, l'utilisateur peut utiliser les gestes déictiques.

Gestes déictiques

Ces gestes naturels permettent de désigner à distance un ou plusieurs objets. Le geste le plus courant est de pointer un objet avec l'index tendu. Ce type de geste peut être combiné à d'autres modalités pour indiquer une action à réaliser sur l'objet que l'on est en train de pointer du doigt. Par exemple dans [Latoschik, 2001], les auteurs présentent une application qui utilise les gestes et la voix. Pour saisir un objet, il faut effectuer une désignation et prononcer une phrase du type "prend cet objet". Tout le problème, alors, est de pouvoir détecter les moments de synchronisation entre le geste et la voix et de résoudre les co-références dans l'énoncé multimodal.

Gestes descriptifs

Un troisième type de gestes naturels existe en plus des gestes de manipulation et de désignation. Il s'agit des gestes permettant d'effectuer des descriptions. Par exemple, pour décrire un ballon, on peut utiliser l'écartement et la forme des mains.
Ce type de gestes descriptifs peut servir soit à créer des objets, soit à faire référence à un objet de la scène. Par exemple, dans [Pratini, 2001], on peut voir la création de surfaces par extrusion à partir d'un profil, ou la création de primitives géométriques (sphère, cube...) à partir de la forme de la main. Dans [Sowa et Wachsmuth, 2001], les auteurs présentent un prototype où l'on peut désigner un objet de la scène en utilisant des gestes iconiques qui reprennent les traits saillants de l'objet. Par exemple dans la figure 4.5, on voit à gauche le geste iconique où l'on représente les six faces d'un cube. Ensuite, l'étape du milieu indique l'extraction des caractéristiques (orthogonalité, égalité de longueurs, etc) qui sont représentées sous la forme d'un graphe. L'étape finale (à droite) consiste alors à comparer ce graphe avec ceux contenus dans une base de donnée de formes où à chaque objet est associé un ou plusieurs graphes décrivant ses caractéristiques. Le graphe le plus proche de celui extrait de l'analyse du geste est retenu et donne l'objet décrit.

L'introduction d'informations de type iconique dans le geste en réalité virtuelle amène de nouvelles problématiques de nature linguistique. C'est pourquoi, il y a encore peu d'applications qui utilisent ces gestes, et lorsque c'est le cas, elles se restreignent à des aspects relativement simples.

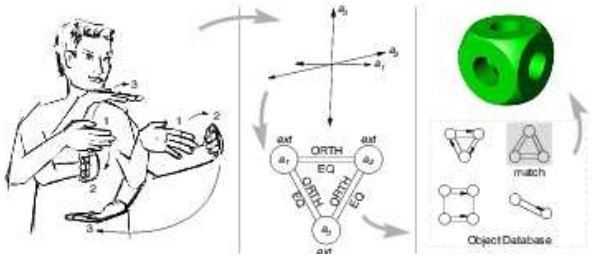

FIG. 4.5 – Exemple de traitement des gestes iconiques en réalité virtuelle. Depuis la description gestuelle d'un cube (à gauche), on extrait un graphe décrivant les propriétés topologiques (milieu) que l'on compare ensuite à un ensemble de graphes correspondant aux différents objets possibles. (AI Group,Université de Bielefeld)

Gestes de commandes

Pour certaines applications en réalité virtuelle, on peut avoir besoin un vocabulaire spécifique où chaque geste correspond à une commande que l'on désire passer à l'application. Ce type de gestes demande alors un apprentissage de la part de l'utilisateur qui doit retenir les différentes commandes. D'autre part, ces gestes peuvent entraîner une certaine fatigue de l'utilisateur si une commande doit être souvent répétée. Du fait de ces inconvénients, il est en général préférable d'utiliser plutôt une modalité comme la voix qui est plus adaptée à ce type de tâche.

Gestes de navigation

Les gestes peuvent aussi être utilisés pour naviguer dans le monde virtuel. Par exemple, on pointe du doigt l'endroit où l'on veut se diriger, et on contrôle la vitesse en éloignant plus ou moins la main du corps [Mine, 1995]. L'utilisation de périphériques est en générale préférée à ce type de gestes dont l'usage reste rare.

4.2.3 Enchaînement de gestes en réalité virtuelle

Le plus souvent, les gestes sont utilisés de manière isolée en réalité virtuelle. C'est à dire que l'utilisateur n'enchaîne pas les gestes les uns après les autres comme c'est le cas en langue des signes, mais de manière séparée avec des périodes de *repos* entre eux. Par exemple, lorsque l'on va manipuler de manière directe un objet, on effectue un geste de saisie pour pouvoir manipuler l'objet puis on revient à une position de repos avant de passer à un autre objet ou une autre tâche.

Toutefois, dans certains cas, un enchaînement de gestes est nécessaire pour exprimer un tout. C'est le cas, par exemple, si l'on fait succéder à un geste déictique, un autre geste pour indiquer une action à réaliser sur l'objet que l'on vient de pointer. De même, en général, les gestes descriptifs s'enchaînent pour effectuer une description d'un objet. Par exemple si l'on regarde la figure 4.5, on peut compter trois gestes successifs qui permettent de symboliser les côtés du cube.

4.3 Transferts entre la langue des signes et la réalité virtuelle

Le domaine de la langue des signes et celui de la réalité virtuelle semblent au premier abord fort éloignés l'un de l'autre. Toutefois, ces deux domaines présentent plusieurs points communs. Nous allons maintenant développer ces points communs et indiquer quels sont les transferts de modèles envisageables entre les deux domaines.

4.3.1 La modalité gestuelle

Un premier point commun aux deux domaines est l'utilisation de la modalité gestuelle. Cette modalité est l'essence même de la langue des signes et constitue une modalité importante en réalité virtuelle pour ceux qui s'intéressent aux interactions naturelles. Il est donc évident que la langue des signes et la réalité virtuelle peuvent bénéficier l'une de l'autre pour tout ce qui concerne les modèles, méthodes et techniques permettant de traiter le canal gestuel.
Ainsi, la langue des signes a largement bénéficié des dispositifs techniques d'acquisition de la réalité virtuelle (capteurs de position/orientation, gants numériques, capture infrarouge, etc). Au niveau de la reconnaissance de gestes, de nombreuses expérimentations ont été faites dans le domaine de la langue des signes et en particulier sur celui des signes isolés. Toutes ces expériences ont profité au domaine de la reconnaissance du geste en général, et donc à la réalité virtuelle. Toutefois, certains systèmes de ce domaine restent assez éloignés de ceux de la langue des signes du fait de l'absence de fonctionnalité sémiotique pour les gestes en réalité virtuelle.
La recherche reste très active dans le domaine de la reconnaissance de la langue des signes et vise à une amélioration constante des systèmes. Il est donc fort probable que dans l'avenir, certaines avancées techniques en reconnaissance de la langue des signes soient profitables à la réalité virtuelle.

4.3.2 Gestes bimanuels

L'aspect bimanuel est présent dans tous les domaines où sont utilisés les gestes. En langue des signes, l'utilisation conjointe des mains occasionne l'expression d'informations spatiales et temporelles comme nous avons pu le voir dans le chapitre 1. En réalité virtuelle, les gestes sont souvent moins porteurs d'informations ; toutefois on retrouve les mêmes situations entre les deux mains.
Deux situations peuvent se produire en langue des signes : soit les deux mains présentent des similitudes au niveau des différents paramètres (configuration, mouvement, emplacement et orientation), soit elles présentent, au contraire, des dissimilarités entre elles (rôles de main dominante et de main dominée en particulier).
On retrouve le même comportement des mains en réalité virtuelle comme on peut le voir dans l'étude de [Cutler et al., 1997] : certains gestes utilisent des propriétés de similarité, tandis que d'autres sont basés sur la dissimilarité. Par exemple, une opération de changement de taille se fait en utilisant des mouvements de direction symétriques, tandis qu'une opération de rotation se fait en effectuant un arc avec la main dominante et en ayant la main dominée fixe pour donner le centre de rotation (voir figure 4.6).

Du fait que l'utilisateur va éventuellement vouloir effectuer des modifications sur les objets de la scène, on va souvent avoir des interactions de type main dominante/main dominée entre les mains. La main dominée tient l'objet ou donne une référence spatiale par rapport à cet objet, et

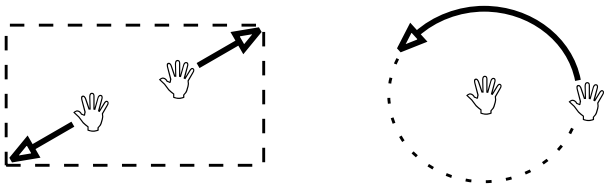

FIG. 4.6 – A gauche une opération de changement de taille avec un mouvement symétrique, à droite une opération de rotation avec une main dominante et une main dominée

la main dominante effectue une action de modification de cet objet.

On va alors être confronté aux mêmes problèmes qu'en langue des signes : à savoir distinction des gestes à une main des gestes à deux mains, interprétation des relations main dominante/main dominée pour savoir à quoi se réfère l'action à effectuer, etc.

4.3.3 L'utilisation de l'espace

Un troisième point commun entre les deux domaines est l'utilisation de l'espace en trois dimensions.

Du côté de la langue des signes, l'espace joue un grand rôle via la scène de narration. Au niveau syntaxique, il permet d'identifier et de mettre en relations les différentes entités des phrases. Au niveau sémantique, l'espace permet d'exprimer des informations spatiales ou temporelles. En particulier, les transferts situationnels de la langue des signes permettent de décrire des scènes en trois dimensions dans lesquelles on peut indiquer la disposition d'entités et le déroulement d'actions.
Du fait que l'on a la possibilité de décrire un monde en trois dimensions, il est nécessaire de pouvoir également décrire la forme des objets qui peuplent ce monde. La langue des signes dispose pour cela de mécanismes iconiques - spécialement les transferts de taille et/ou de forme - qui permettent d'effectuer la description imagée d'objets, de personnes ou d'animaux.

En réalité virtuelle, on plonge un utilisateur dans un monde virtuel qui est en trois dimensions. Selon le domaine d'application, l'utilisateur va vouloir créer des objets ayant une forme simple ou complexe. Il va devoir être capable de manipuler ces objets et de les disposer dans le monde qui l'entoure. Peut-être même que l'utilisateur va avoir besoin de fournir des renseignements sur les objets qui l'entourent, ces informations pouvant être des relations entre ces objets.

On constate donc des similarités entre les deux domaines qui, tous les deux, utilisent une scène spatiale. Il nous semble donc très intéressant de transférer les mécanismes de la langue des signes dans le domaine de la réalité virtuelle. Par exemple, les transferts de taille et/ou de forme peuvent servir à la création d'objets du monde virtuel, les transferts de situations peuvent être utilisés pour décrire une scène virtuelle.
Concernant la partie transferts de taille et/ou de forme, on peut voir que ce mécanisme a déjà commencé à être exploité en réalité virtuelle dans les travaux récents ([Pratini, 2001], [Sowa et Wachsmuth, 2001]), sans pour autant faire référence au fonctionnement de la langue des signes. De fait, ces travaux abordent le problème des gestes illustratifs avec une approche bas ni-

veau où ne sont pas utilisées les informations sémantiques. Seules les informations spatiales sont exploitées.
Il serait intéressant de voir quels pourraient être les bénéfices apportés par ces travaux sur la description de forme au domaine de reconnaissance de la langue des signes, et inversement d'étudier les avantages procurés par la langue des signes au domaine des interactions de la réalité virtuelle.

4.3.4 Lien entre les gestes et les objets composant la scène

En langue des signes, certains gestes sont modifiés en fonction des objets qui occupent la scène. Par exemple, la configuration de la main lors de l'énonciation de verbes directionnels va être fonction de la forme de l'objet concerné par l'action. Si l'on veut "envoyer un colis", les mains vont naturellement décrire la forme et la taille du colis (image de gauche, figure 4.7).
En réalité virtuelle, on pourrait transférer ce mécanisme et permettre ainsi une plus grande richesse d'interactions et se rapprocher un peu plus du comportement naturel d'un utilisateur. Par exemple, lorsque l'on saisit un objet, la configuration de la main va dépendre de la forme de cet objet. Si l'on saisit un petit objet fin, on ne va pas utiliser la même configuration de la main que pour un objet plus gros (images de droite, figure 4.7).

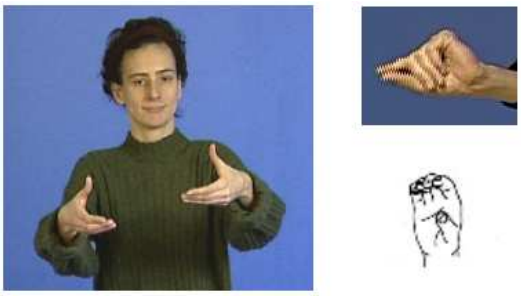

FIG. 4.7 – A gauche, le verbe [DONNER] appliqué à un colis en langue des signes. A droite, deux exemples de configuration de la main en cas de saisie d'objets en réalité virtuelle (Extraits des corpus ARC-LSF et LS-COLIN et du dictionnaire de l'IVT)

Pour pouvoir traiter ce type d'interaction gestuelle dans les deux domaines, le vocabulaire gestuel va être fixé au niveau des paramètres en fonction des objets qui vont intervenir dans la scène de narration ou la scène virtuelle. Les approches basées sur un modèle physique permettent de gérer ce problème en réalité virtuelle du fait que l'on ne cherche pas à reconnaître les gestes mais seulement à simuler les interactions physiques entre la main et le monde virtuel.
Dans les cas où l'on n'utilise pas un modèle physique, la reconnaissance de ces gestes variables reste très difficile et implique généralement de se restreindre aux cas les plus courants. Du fait de l'aspect sémantique de tels gestes, on aurait tendance en langue des signes à effectuer l'interprétation du geste après l'étape de reconnaissance lors d'une phase d'analyse sémantique.

4.3.5 Modélisation linguistique

A première vue, les gestes utilisés en réalité virtuelle ne présentent aucune corrélation entre eux. C'est le cas, par exemple, avec les gestes de commande qui en général se succèdent sans aucune relation (mise à part le but final voulu par l'utilisateur). Toutefois, d'autres gestes, comme

les gestes déictiques ou descriptifs, vont être liés à ce qui est effectué sur le moment ou dans ce qui suit. Par exemple, on peut imaginer le cas où l'utilisateur commence par décrire gestuellement un cube pour qu'il soit créé. Puis il désigne de l'index l'endroit où doit être placé le cube. Il y a alors une structure syntaxique et sémantique qui apparaît dans cette séquence de gestes que l'on pourrait traduire par "met un **cube comme ceci** à **cet** endroit".

Actuellement, les systèmes en réalité virtuelle ont une approche assez bas niveau des gestes. On peut éventuellement trouver des structures syntaxiques sous formes d'automates qui permettent l'enchaînement de plusieurs gestes pour effectuer une action complexe, mais qui imposent un certain formalisme à l'utilisateur. Pour traiter l'exemple précédent, on pourrait concevoir un automate qui modélise successivement l'action de création puis de positionnement (figure 4.8). Mais un tel automate ne permet pas, à un utilisateur, de formaliser un déroulement d'actions du type "met à **cet** endroit un **cube comme ceci**". Ce qui pourtant signifie strictement la même chose au final.

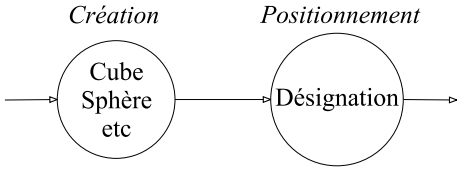

FIG. 4.8 – Exemple d'automate pour gérer l'action de création d'un objet à un endroit de la scène.

Le niveau sémantique n'est pas du tout abordé en réalité virtuelle, car cela peut sembler de trop haut niveau pour les types d'interactions qui y sont utilisés. Pourtant, l'usage du niveau sémantique permettrait une plus grande liberté à l'utilisateur pour enchaîner des gestes entre eux, ou pour combiner le geste avec d'autres modalités.

Dans le domaine de la langue des signes, des recherches se portent bien évidemment sur la modélisation sémantique des signes du fait des propriétés linguistiques de ces gestes. L'intérêt des modélisations utilisées pour la langue des signes est le fait qu'elles permettent d'intégrer les notions d'espace, ce que d'autres modélisations sémantiques (comme celles des langues orales) ne font pas forcément. Par exemple dans [Lejeune et al., 2002], les auteurs proposent une modélisation de certains aspects de la langue des signes française à l'aide de schèmes sémantico-cognitifs. Ce type de modèle permet de décrire des propriétés spatio-temporelles statiques ou dynamiques. On peut, par exemple, modéliser le déplacement d'une entité d'un emplacement à un autre (figure 4.9). Une fois que l'on a pu formaliser une relation spatiale ou temporelle grâce à ce type de modélisation, on peut appliquer des traitements sur cette représentation afin d'en extraire des informations pertinentes.

L'utilisation de ces modèles sémantiques pourrait donc être tout à fait intéressante pour la réalité virtuelle dans l'optique d'interactions gestuelles laissant une plus grande liberté à l'utilisateur tout en gardant une cohérence globale.

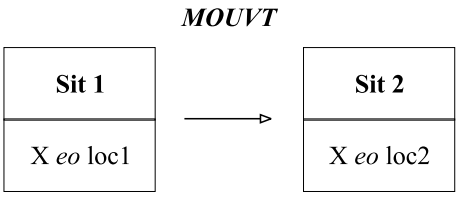

FIG. 4.9 – Un exemple de représentation de schème sémantico-cognitif où l'on décrit le déplacement ($MOUVT$) d'une entité X d'un emplacement $loc1$ à un emplacement $loc2$. Le changement de l'état de l'entité x est exprimé par le fait que l'on passe d'une situation de départ $Sit1$ à une situation d'arrivée $Sit2$. eo indique que l'entité X est localisée à l'emplacement $locx$.

4.3.6 Bilan

On vient de voir que de nombreux points communs existent entre la réalité virtuelle et la langue des signes. Dans les chapitres précédents, nous avons exposé le positionnement de notre étude par rapport aux problématiques de la langue des signes et des interactions bimanuelles. Ce positionnement porte sur des énoncés contenant des transferts de situations et où s'exercent des relations spatiales entre les mains. Ce type d'énoncé correspond à une utilisation poussée de l'espace de narration pour effectuer des descriptions.
C'est pourquoi, nous proposons en réalité virtuelle d'intégrer, au sein de l'interaction gestuelle, des gestes permettant d'effectuer la description de scènes spatiales en utilisant le principe de fonctionnement des transferts situationnels. Ce type de gestes permet soit de créer des scènes, soit de modifier des scènes en permettant une manipulation "à distance" des objets. L'approche exposée au chapitre précédent est ainsi proposée pour traiter ce type de gestes dans le domaine de la réalité virtuelle.

La troisième partie de ce mémoire de thèse décrit les expérimentations réalisées. Ces expérimentations proposent des implémentations de l'architecture proposée pour la reconnaissance automatique de gestes bimanuels dans les contextes de l'interaction gestuelle en réalité virtuelle (chapitre 5) et de l'interprétation automatique de la langue des signes (chapitre 6)

Troisième partie

Implémentations et expérimentations

Chapitre 5

Expérimentation dans le cadre de la réalité virtuelle

Nous avons vu au chapitre précédent que différents transferts de modèles peuvent se concevoir entre la réalité virtuelle et la langue des signes. En particulier, certains signes et structures de la langue des signes permettent de traiter l'aspect spatial en réalité virtuelle. Le domaine de la représentation d'objets dans l'espace a déjà été abordé dans des études utilisant des gestes descriptifs ou iconiques. Nous nous intéresserons ici plutôt à des gestes permettant la description d'une scène spatiale, c'est à dire des gestes inspirés des transferts situationnels. D'une part, cela permet de traiter un domaine non abordé par le geste en réalité virtuelle et d'autre part, cela permet d'étudier dans ce contexte notre architecture pour les interactions entre les mains.
Dans ce chapitre, nous parlons dans un premier temps, en section 1, de l'adaptation de notre architecture au domaine de la réalité virtuelle et à l'environnement logiciel du projet VENISE. Dans la deuxième section, nous passons à l'évaluation du système de reconnaissance dans le domaine de la réalité virtuelle. En section 3, nous discutons des gestes bimanuels en réalité virtuelle et enfin nous faisons dans la dernière section une expérimentation de certaines structures des transferts de situation pour effectuer la description de scènes.

5.1 Cadre d'un système de reconnaissance en réalité virtuelle

Un système de reconnaissance ne peut pas être directement importé au sein d'un dispositif de réalité virtuelle. On doit gérer tout d'abord un certain nombre de problématiques propres à une application s'intégrant dans un environnement virtuel. C'est pourquoi, généralement, une application de réalité virtuelle utilise une architecture permettant de s'interfacer avec le dispositif et de gérer un certain nombre des problématiques.
Nous présenterons un peu plus loin la plate-forme EVI3d (Environnement Virtuel et Interaction 3d) qui permet de développer des applications au sein du projet VENISE. Mais tout d'abord, nous allons voir quelles sont les problématiques auxquelles est confronté un système de reconnaissance de geste.

5.1.1 Problématiques propres à un environnement virtuel

Temps réel

Le temps de réponse est un des grands problèmes du traitement des interactions entre l'homme et la machine. Si le délai, entre le moment où l'utilisateur effectue son action et le moment où le système fait le traitement correspondant, est trop grand, l'utilisateur va avoir une sensation de décalage.
Si dans le cas d'interactions avec une station de travail, on peut tolérer un intervalle de temps relativement important, ce n'est pas le cas en réalité virtuelle et plus particulièrement avec les gestes. En effet, si un utilisateur saisit un objet, puis le déplace immédiatement, il va vouloir que l'objet suive sa main visuellement lors du déplacement. Si le temps de réponse du système de reconnaissance de gestes est trop long, l'utilisateur va avoir pendant un court instant l'impression que son geste n'a pas été pris en compte. Ces problèmes de latence induisent généralement une modification du comportement de l'utilisateur (ralentissement de ses mouvements) et une dégradation de ses performances [Vercher, 2003].
Pour éviter cette sensation de décalage, il faut que le système soit capable de traiter les gestes en ***temps réel*** (ou ***temps interactif***), c'est à dire que le délai de réaction soit suffisamment court pour que l'utilisateur ne s'aperçoive pas du délai entre le moment où il effectue le geste et le moment où le système répercute sur la scène l'action effectuée par le geste. Dans une application de réalité virtuelle, la fréquence d'acquisition et le type de système de reconnaissance vont devoir donc être choisis en fonction de cette contrainte.

Diversité des périphériques

Dans un dispositif de réalité virtuelle, différents modèles de périphériques, effectuant l'acquisition du même type de données, peuvent coexister. Par exemple, dans le dispositif du LIMSI-CNRS, différents gants numériques sont disponibles pour la main gauche ou la main droite, et tous les gants n'ont pas le même nombre de capteurs. D'autre part, on doit combiner plusieurs périphériques d'acquisition afin de pouvoir acquérir des données différentes mais faisant partie d'une même modalité. Pour les gestes, on combine typiquement des gants numériques et un système de capture de position/orientation.
Un système en réalité virtuelle doit donc être capable de gérer cette multitude de périphériques, et

en particulier être capable de passer d'un ensemble de périphériques à un autre quand l'on passe d'une application à une autre.

Nombreux utilisateurs

Dans notre cas, le dispositif de visualisation consiste en deux écrans de grande taille disposés en angle. C'est typiquement un dispositif prévu pour pouvoir accueillir plusieurs personnes afin qu'elles puissent coopérer ensemble sur la même tâche. On doit donc être capable d'avoir plusieurs systèmes de reconnaissance de gestes qui fonctionnent au sein de la même application, mais aussi on doit prévoir la possibilité de changer l'utilisateur d'un système de reconnaissance sans devoir redémarrer l'application (les gants étant souvent en nombre limité, ils peuvent passer d'un utilisateur à un autre).

Vocabulaire variable suivant l'application

Suivant le type d'application de réalité virtuelle, l'utilisateur ne va pas avoir besoin du même type d'interactions (dans un cas, il aura besoin de manipuler des objets ; dans un autre, il voudra réaliser des gestes de commandes, etc). Ainsi, à chaque application peut correspondre un vocabulaire gestuel propre à cette application.
Un système de reconnaissance de gestes doit alors avoir la capacité de traiter différents types de vocabulaire, afin d'être opérationnel sur diverses applications en réalité virtuelle.

5.1.2 La plate-forme EVI3d

La plate-forme logicielle ***EVI3d*** [Touraine et al., 2002] a été développée au sein du laboratoire LIMSI-CNRS pour permettre la création d'applications en réalité virtuelle et Augmentée. Elle peut gérer différents aspects d'une application en environnement virtuel : affichage sur dispositifs de grande taille, stéréoscopie, gestion de périphériques divers, etc. Ici, seul l'accès aux données des périphériques nous intéresse et c'est ce point que nous allons exposer avant de présenter le système de reconnaissance.

Le système EVI3d permet d'effectuer l'acquisition et l'acheminement de données à travers une architecture distribuée. Pour effectuer la distribution des données, deux canaux sont utilisés : un canal haut débit de type flot de données, et un canal de débit plus réduit où les données transitent sous forme d'événements. C'est sur ce canal événementiel que va transiter, entre autres, les données issues des différents périphériques d'acquisition du dispositif immersif. Nous nous intéressons donc uniquement à ce canal par la suite.

Le plus souvent, pour des raisons techniques (connecteurs disponibles, emplacements pour cartes d'acquisition ...), les périphériques sont connectés sur différents ordinateurs. De même, l'application peut être répartie sur plusieurs ordinateurs. C'est pourquoi une approche de type client/serveur est utilisée au sein de la plate-forme EVI3d. Le serveur, qui est appelé ***EVserveur***, est distribué sur les différentes machines où sont présents les périphériques [Bourdot et Touraine, 2002a]. Ensuite, si une application a besoin d'accéder aux données d'un périphérique, il lui suffit de se connecter au serveur en tant que client.
Mais cette architecture client/serveur ne sert pas uniquement à accéder aux périphériques gérés

par l'EVserveur. Elle permet à différents clients d'échanger des données via le serveur. Cela offre une grande souplesse au niveau d'une application qui peut alors utiliser plusieurs briques logicielles sous formes de clients. Une application qui utilise la plate-forme EVI3d est donc généralement une arborescence composée de noeuds clients et de noeuds serveurs et qui est répartie sur une ou plusieurs machines. L'organisation de ces noeuds et la transmission d'événements entre eux se déroule de la manière suivante :

- A un noeud serveur peuvent se connecter plusieurs noeuds clients ou serveurs. On va donc pouvoir former un réseau de noeuds (qui aura une forme arborescente) permettant de transférer des événements d'un noeud à un autre.
- Lors de sa connexion à un noeud serveur, un client est informé des différentes sources de données possibles (périphériques et éventuellement clients) et peut alors choisir parmi ces sources celles dont il a besoin.
- Un périphérique est géré par le noeud serveur de l'ordinateur auquel il est connecté. Les données acquises par ce périphérique sont envoyées sous forme d'événements à tous les noeuds clients qui ont demandé les données de ce périphérique. Un client reçoit donc uniquement les événements correspondant aux périphériques demandés.
- Un noeud client peut envoyer des événements au noeud serveur auquel il est raccordé, ce dernier renverra les événements à tous les autres noeuds qui sont demandeurs de ces événements. Ce noeud est alors perçu par les autres clients comme s'il était un périphérique.

Du fait de cette disposition client/serveur et de l'utilisation d'événements, le système offre une gestion simplifiée des périphériques. En effet, un client n'a pas à gérer les périphériques dont il a besoin, il n'y a pas non plus de conflit d'accès à un périphérique et un client n'a pas à se soucier de l'acheminement des données jusqu'à lui (il indique juste à son noeud serveur qu'il veut les données de ce périphérique). De plus, on dispose d'un type d'événement pour chaque catégorie de périphérique. Par exemple, il y a un événement de type 6dof (6 degrees of freedom : x,y,z,rx,ry,rz) pour les périphériques de capture de position/orientation, un événement de type bouton pour les boutons de souris 3D, etc. Grâce au typage des événements, un client peut même faire abstraction du modèle de périphérique utilisé pour acquérir les données. On peut alors intervertir un périphérique avec un autre du même type sans que les clients s'en aperçoivent.

D'autre part, du fait que les clients peuvent communiquer entre eux, cette architecture permet une grande modularité pour les applications de réalité virtuelle. En effet, un client développé pour une certaine application de réalité virtuelle peut très bien être réutilisé dans une autre, il suffit d'avoir le même protocole de dialogue d'une application à une autre.
Cela est particulièrement utile dans des applications multimodales où l'on peut gérer chacune des modalités avec un module dédié. L'application, ensuite, "n'a plus" qu'à faire une fusion des événements issus de ces différents modules. Des travaux au sein d'EVI3d ont d'ailleurs été effectués pour gérer ce problème de fusions d'événements arrivant de plusieurs sources [Touraine et al., 2002].

Dans la figure 5.1, on peut voir un exemple d'utilisation de cette architecture dans le cadre d'une application faisant intervenir le geste et la parole. Dans ce cas, l'application a besoin de deux modules : un pour interpréter les gestes et un pour interpréter la parole.
Dans cet exemple, l'arborescence se répartit sur quatre ordinateurs : un sur lequel est connecté un périphérique de capture de position-orientation (Flock of Bird™), un deuxième ordinateur avec un

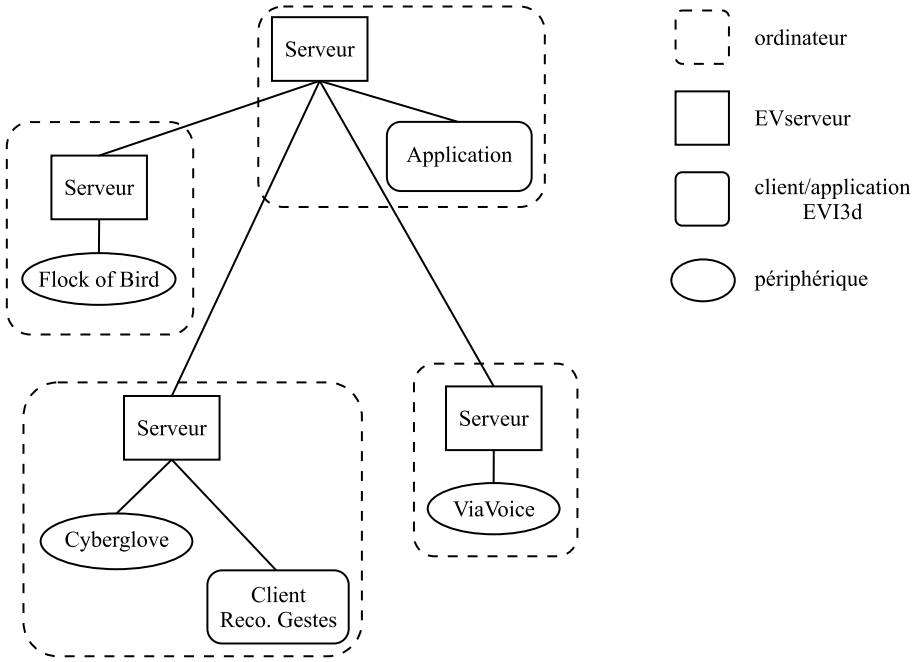

FIG. 5.1 – Exemple d'une configuration d'architecture distribuée avec la plate-forme EVI3d (Flock of Bird™ : périphérique de capture de position/orientation, CyberGlove™ : gant numérique, ViaVoice™ : logiciel de reconnaissance de la parole).

gant numérique (CyberGlove™), le troisième sur lequel est branché un microphone et qui dispose d'un logiciel de reconnaissance de la parole (ViaVoice™ d'IBM), et le quatrième ordinateur sur lequel on va effectuer l'affichage et où va fonctionner notre application. On peut voir dans ce cas ci, qu'un module de reconnaissance peut aussi bien être intégré au niveau d'un client, que d'un périphérique : le module de reconnaissance de geste est intégré sous forme d'un client, tandis que pour la voix, le programme ViaVoice™ est interfacé avec le système sous la forme d'un périphérique.
Cet exemple correspond à une configuration qui a été utilisée plusieurs fois lors de manifestations de type grand public ou conférence auxquelles à participé le LIMSI-CNRS, ce n'est donc pas seulement un exemple théorique de ce que l'on peut faire avec l'EVserveur de la plate-forme EVI3d.

5.2 Module de reconnaissance : ClientReco

Le module de reconnaissance, que nous allons présenter ici, a été développé dans l'objectif de pouvoir être utilisé dans des applications de réalité virtuelle et en particulier des applications multimodales comme l'exemple donné précédemment pour l'architecture EVI3d ([Bossard et al., 2004]). Le système de reconnaissance de gestes a donc été développé sous forme d'un client EVI3d. Dans ce qui va suivre, ce client va correspondre au module de reconnaissance de l'architecture proposée en section 3.1 (parties rouges dans la figure 5.2). Il y a donc trois instances de ClientReco qui fonctionnent en parallèle : une pour les gestes bimanuels, une pour les gestes de la main gauche et une pour les gestes de la main droite.

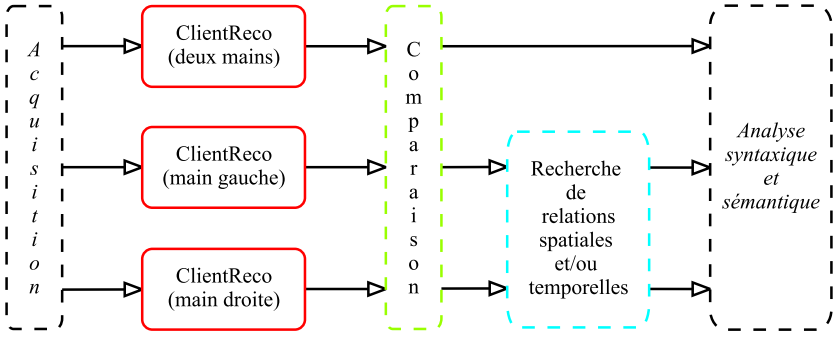

FIG. 5.2 – Place de ClientReco dans l'architecture de reconnaissance de gestes bimanuels.

Du fait de son utilisation dans le cadre de la réalité virtuelle, ce client de reconnaissance de geste ne va pas avoir affaire à des vocabulaires de grande taille (en général une vingtaine de gestes interviennent au grand maximum dans une application de réalité virtuelle). Il n'est donc pas nécessaire d'effectuer une décomposition des gestes en sous éléments pour des problèmes de taille de vocabulaire, comme cela a été proposé en section 3.2.3 dans le cas de la langue des signes. D'autre part, parmi les quatre paramètres (configuration, emplacement, orientation et mouvement) seule la configuration a besoin d'être interprétée (sauf dans le cas où l'on considère le geste de manière partielle sous la forme d'un mouvement, comme en section 4.2.1). Les autres paramètres ne participent pas à la définition des gestes et sont plutôt utilisés de manière numérique, comme par exemple dans un geste de déplacement où l'on se contente de reconnaître la saisie et le relâchement de l'objet. La position n'a pas besoin d'être reconnue comme faisant partie d'une zone spécifique. C'est pour ces raisons que dans ce qui suit, le client ne va pas traiter les quatre paramètres séparément, mais plutôt uniquement la configuration.
Au cas où l'on a affaire à un vocabulaire plus complexe qui nécessite une identification des quatre paramètres, alors le client décrit dans cette section correspond plutôt aux systèmes de reconnaissance traitant chaque paramètre (les quatre modules parallèles dans la figure 5.3). Dans ce cas, le module qui effectue la reconnaissance du geste complet serait alors sous la forme d'un autre client au sein d'une application utilisant EVI3d.

En plus de prendre en compte les propriétés du vocabulaire, le client doit aussi pouvoir traiter les aspects particuliers de la réalité virtuelle que nous avons exposés précédemment. Il faut, entre autre, prendre en compte le fait que l'application doit effectuer une réponse en temps réel vis à vis des interactions de l'utilisateur. Cela pose des contraintes sur la nature du système de reconnaissance.

5.2.1 Choix d'un système statistique

Nous avons écarté des choix possibles les modèles de Markov Cachés car ils ne permettent pas, à priori, une réponse en temps réel du fait de leur nature qui nécessite d'attendre l'arrivée à la fin de la chaîne de Markov pour avoir une réponse. Si l'on doit effectuer la reconnaissance de gestes enchaînés, on est alors obligé de patienter jusqu'à la fin de la phrase pour identifier un à un les gestes.
Nous avons également écarté les réseaux neuronaux, car l'apprentissage de ces derniers nous semble impliquer une étape initiale un peu complexe pour l'utilisateur. En effet, il ne faut pas perdre de

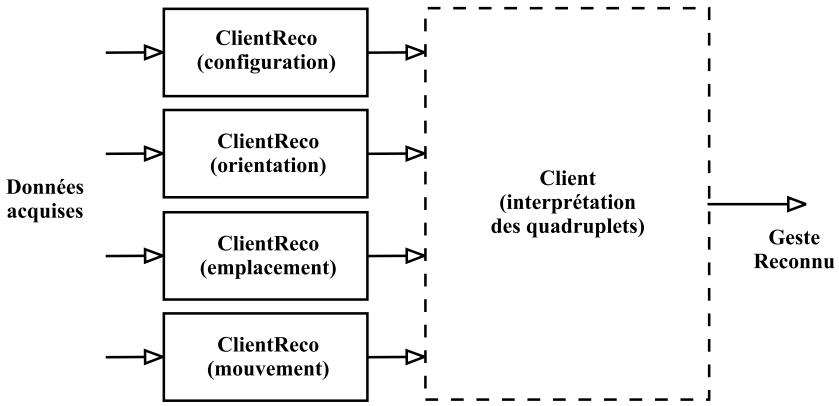

FIG. 5.3 – Utilisation de ClientReco pour reconnaître les quatre paramètres (configuration, orientation...) de manière indépendante.

vue que les utilisateurs du système de reconnaissance auront à effectuer un apprentissage et qu'ils ne sont pas forcément habitués au processus. Or l'apprentissage d'un réseau neuronal peut parfois se révéler délicat et de cet apprentissage dépend grandement la qualité du réseau.
Le choix s'est donc porté sur un système de type statistique pour éviter les inconvénients de l'un et l'autre des autres systèmes. Nous avons choisi de baser notre système sur celui proposé par Dean Rubine [Rubine, 1991] pour effectuer la reconnaissance de tracés manuscrits. Ce système a déjà été adapté et utilisé au sein du laboratoire dans le cadre de la langue des signes [Braffort, 1996a], ce qui nous permet d'avoir une certaine expertise sur les capacités du système. Ce dernier est basé sur la comparaison du vecteur de données fourni en entrée du système avec des vecteurs moyens représentant chaque classe de gestes.
Ce type de comparaison se fait très rapidement, ce qui nous permet d'avoir une réponse en temps réel du système de reconnaissance vis à vis de l'application. Il fonctionne bien avec des petits vocabulaires, et se satisfait d'apprentissages réduits. L'utilisation de vecteurs moyens n'est pas très adaptée aux gestes dynamiques et aux gestes enchaînés, mais cela peut être résolu en utilisant des historiques de valeurs au niveau des primitives et en ajoutant un processus de segmentation.
Ce système correspond donc plutôt bien à nos besoins en réalité virtuelle. Nous allons maintenant voir le détail du fonctionnement de ce système.

Description du système statistique

Un geste va être codé sous la forme d'un ensemble de primitives (par exemple les valeurs angulaires des articulations, la vitesse du mouvement, la courbure...), donc à un geste va correspondre un vecteur. Pour représenter chaque classe de geste, on utilise le vecteur moyen et la matrice de covariance associée. C'est la phase d'apprentissage qui va s'occuper de calculer ces deux éléments. Si on note $\bar{p}_i^c$ la ième composante du vecteur moyen pour la classe c, p_{ei}^c la ième composante du vecteur exemple numéro e, E^c le nombre d'exemples appris par le système pour la classe c, et P le nombre d'éléments des vecteurs, on obtient le vecteur moyen tout simplement de la manière suivante :

$$\bar{p}_i^c = \frac{1}{E^c} \sum_{e=1}^{E^c} p_{ei}^c \qquad 1 \leq\ i \leq\ P$$

On calcule ensuite chaque élément s_{ij}^c (i et j les indices de l'élément) des matrices de covariance s^c (au coefficient $\frac{1}{E^c}$ près) :

$$s_{ij}^c = \sum_{e=1}^{E^c} (p_{ei}^c - \bar{p}_i^c)(p_{ei}^c - \bar{p}_j^c) \qquad 1 \leq\ i,j \leq\ P$$

Pour effectuer la classification des gestes, on utilise des fonctions linéaires dont les poids sont calculés en fonction des vecteurs moyens et d'une matrice de covariance s obtenue à partir des matrices de covariance de chaque classe. C correspond au nombre de classes, i et j aux indices d'un élément de la matrice.

$$s_{ij} = \frac{\sum_{c=1}^{C} s_{ij}^c}{-C + \sum_{c=1}^{C} E^c} \qquad 1 \leq\ i,j \leq\ P$$

Pour chaque classe, on va donc avoir une fonction d'évaluation dont les poids sont calculés de la manière suivante (w_j^c est le poids pour la jème composante du vecteur fourni à la fonction d'évaluation, w_0^c est le terme constant de la fonction d'évaluation) :

$$w_j^c = \sum_{i=1}^{P} (s^{-1})_{ij} \bar{p}_i^c \qquad 1 \leq\ j \leq\ P$$

$$w_0^c = -\frac{1}{2} \sum_{i=1}^{P} w_i^c \bar{p}_i^c$$

Une fois l'apprentissage effectué (c'est à dire le calcul des vecteurs moyens, des matrices de covariance et des poids des fonctions d'évaluation), on peut classifier un geste en calculant un score pour chaque classe c à l'aide des fonctions d'évaluation. Le score v^c d'un vecteur x est obtenu avec le calcul suivant :

$$v^c = w_0^c + \sum_{i=1}^{P} w_i^c x_i \qquad 0 \leq\ c < C$$

La classe qui obtient le score le plus fort est considérée comme la classe du geste que l'on veut reconnaître. Bien évidemment, ce type de classification ne permet pas de déterminer si un geste appartient ou non au vocabulaire appris par le système. Dean Rubine propose de remédier à ce problème en utilisant la distance de Mahalanobis entre x et $\bar{p}_i^c$ et la probabilité que x appartienne à la classe c. Toutefois, cette solution ne nous convient pas car, d'une part, il faut déterminer des seuils permettant de juger si une classification est correcte ou non (et actuellement, il n'y a pas de processus permettant d'obtenir ces seuils de manière automatique), d'autre part, les valeurs de comparaisons obtenues dépendent de l'apprentissage effectué. Nous reviendrons plus tard sur ces problèmes de distance lorsque l'on abordera le module de comparaison de l'architecture. Maintenant, nous allons voir comment ce système s'intègre au sein de l'architecture du système ClientReco.

5.2.2 Architecture logicielle du client

En prime à la problématique du temps réel, il reste un certain nombre d'éléments à gérer : les périphériques, les utilisateurs et les vocabulaires. Le fait d'intégrer le système de reconnaissance sous la forme d'un client dans une architecture EVI3d permet de simplifier grandement la gestion des périphériques. C'est l'EVserveur qui gère les aspects matériels et l'acheminement des données depuis les périphériques jusqu'au client. Toutefois, la plate-forme EVI3d ne permet pas de régler tous les problèmes, c'est donc à notre client de les gérer à travers son architecture. Il effectue cela au travers de plusieurs tâches : choix et coordination des périphériques, calcul de primitives.

Au niveau des périphériques, on doit être capable de choisir parmi tous ceux disponibles ceux qui nous intéressent (par exemple gant numérique pour main gauche). On doit également pouvoir combiner plusieurs périphériques comme étant une unique source de données (par exemple un gant numérique et un capteur de position/orientation). Lorsque l'on fusionne les données provenant de deux périphériques, une problématique de synchronisation des données surgit. En effet, différents périphériques effectuent rarement leur acquisition de données à une même fréquence. L'architecture va donc devoir intégrer un ou plusieurs modules permettant de gérer ces deux tâches de sélection et fusion.
L'architecture client/serveur de la plate-forme EVI3d permet de gérer aisément l'aspect multi-utilisateur, il suffit de faire un client de reconnaissance de geste par utilisateur. La sélection des périphériques de chaque utilisateur se fait au niveau de chaque client qui, on vient de le voir, est capable d'effectuer une sélection des périphériques nécessaires.
Suivant les applications, on ne va pas toujours avoir le même type d'interactions et donc de vocabulaire gestuel. D'une part, la technique de reconnaissance doit être capable de gérer différents types de vocabulaires, il faut être capable d'extraire les informations pertinentes pour chacun des vocabulaires (étape de calcul des primitives). D'autre part, la diversité de modèles pour un même type de périphérique nécessite des calculs différents pour chacun des modèles (par exemple, lorsque l'on passe d'un gant numérique disposant de cinq capteurs, à un gant doté d'une vingtaine de capteurs). L'architecture du client va donc devoir intégrer un module permettant d'effectuer aisément différents calculs de primitives.

Nous allons voir comment est organisée l'architecture du client, et quels sont les différents modules qui permettent les différentes tâches que nous venons de décrire.

Description de l'architecture de ClientReco

L'application de reconnaissance de gestes a été divisée en deux parties : la partie interface avec l'utilisateur et la partie "moteur". Cette dernière effectue la gestion des périphériques, la reconnaissance et le calcul des primitives (voir figure 5.4). Seule la partie interface a "connaissance" de la partie moteur. Cela permet de faire évoluer les deux parties séparément et éventuellement on peut intégrer la partie moteur à une autre application.

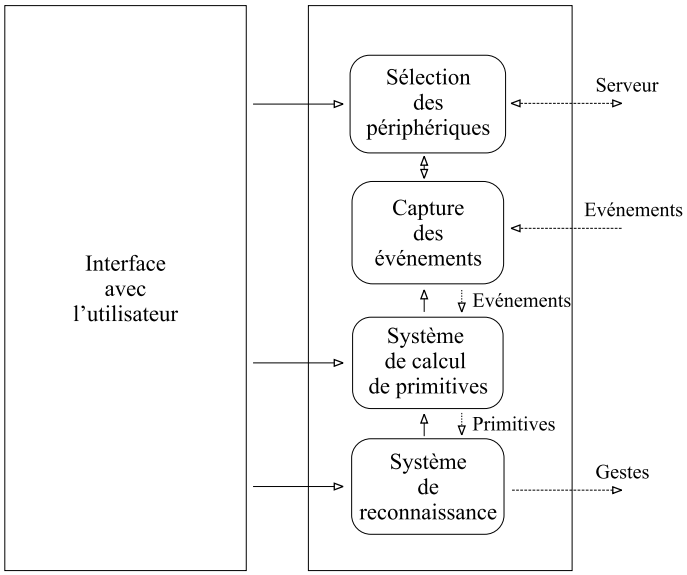

FIG. 5.4 – Architecture du client de reconnaissance de gestes

Dans la partie moteur, on trouve quatre modules :

- **Un module permettant de choisir les périphériques disponibles**. Ce module interagit directement avec le serveur, auquel le client est connecté, pour l'informer des besoins du client. On peut alors sélectionner uniquement les périphériques qui nous intéressent, et seuls leurs évènements parviennent alors au système.

- **Un module qui effectue l'acquisition des événements**. C'est à cet endroit que l'on va effectuer la fusion des données provenant des périphériques. Nous avons choisi d'effectuer cette fusion en se synchronisant sur le périphérique le plus lent afin d'éviter des problèmes de redondance de données. En effet, si l'on se synchronise par exemple sur le périphérique qui a la plus grande fréquence d'événements, pour chaque événement provenant de ce périphérique on va effectuer une fusion avec des événements déjà utilisés (voir figure 5.5). Cela fausse l'évolution dynamique des données captées par les périphériques les plus lents (graphe du haut, figure 5.5). Par exemple si les valeurs correspondent à une position, la duplication des valeurs va donner l'impression que le capteur est immobile à certains moments alors qu'il est mobile.

  Utiliser une synchronisation sur la fréquence basse n'est pas exempte de problèmes, on voit par exemple dans le graphe du bas (figure 5.5) que l'évolution des valeurs du périphérique à fréquence élevée n'est pas gardée intacte. L'idéal serait d'effectuer une moyenne pondérée entre les valeurs précédant et suivant le moment de la fusion - les poids de la moyenne étant répartis en fonction de l'écart entre le moment de la fusion, de la valeur précédente et de la valeur suivante. Ce moyen n'a pas été retenu car il entraîne une charge système assez importante et dérègle la fréquence des trames. C'est pourquoi nous effectuons la fusion au moment des trames du périphérique de fréquence basse. De plus, lorsque la fréquence des périphé-

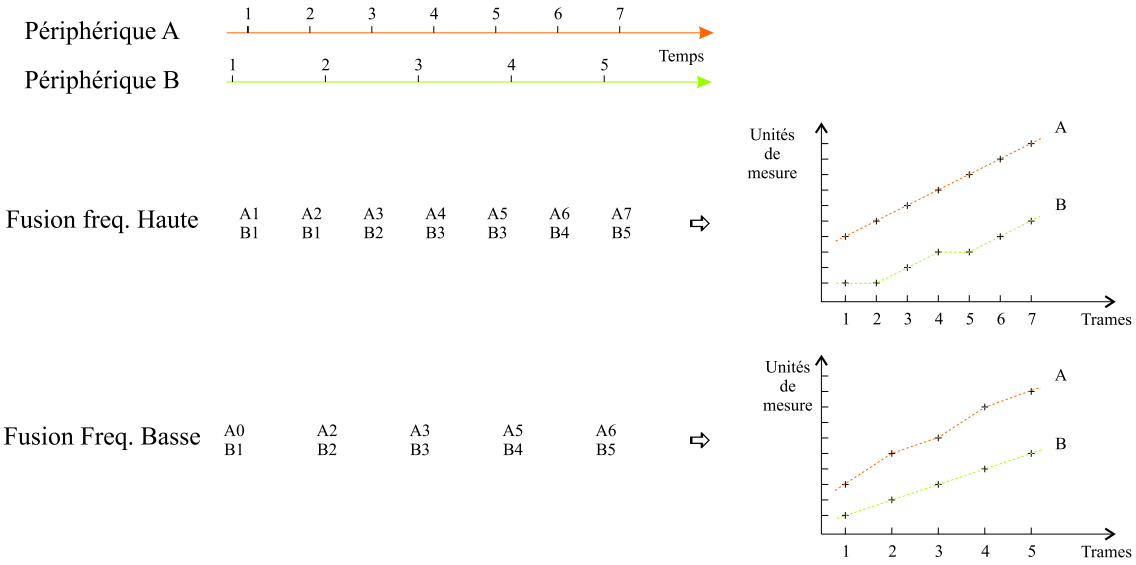

FIG. 5.5 – Illustration des différences de fusion entre une fréquence élevée et une fréquence basse. Les graphes sont un exemple où chaque périphérique capte des valeurs correspondant à une fonction affine.

riques plus rapides est voisine ou un multiple du périphérique le plus lent, les inconvénients de cette méthode sont réduits au minimum.

– **Un module pour calculer les primitives**. Ce module permet d'effectuer un calcul de primitives adapté au vocabulaire gestuel et aux périphériques utilisés. Chaque primitive peut aussi bien être une simple valeur brute issue d'un capteur, qu'une valeur issue d'un calcul complexe (par exemple calcul de courbure). Pour permettre cette grande flexibilité, le module effectue le calcul des primitives au travers d'un graphe dont chaque noeud correspond à une fonction mathématique simple ou complexe (addition, cosinus, dérivée, moyenne, produit vectoriel, etc). Quand un vecteur de données arrive à un des noeuds d'entrée, on applique la fonction de ce noeud au vecteur, puis on envoie le résultat aux noeuds suivants. Ainsi, au fur et à mesure que l'on propage les données au travers du graphe, on va effectuer le calcul des différentes primitives.
On peut créer "à la volée" un graphe complet en donnant successivement les différents calculs à effectuer, mais on peut aussi générer le graphe à partir d'un fichier XML qui contient une description sous la forme d'une liste de noeuds et une liste de transitions.

Dans la figure 5.6, on voit un exemple de graphe de calcul de primitives. A partir de la position et de l'orientation, ce graphe va calculer comme primitives la position, la vitesse, l'accélération et l'orientation. L'orientation est représentée sous la forme de cosinus et de sinus pour éviter des sauts de valeurs comme le passage de 2π à 0. On remarque aussi en sortie du graphe un noeud qui effectue la normalisation des primitives pour que chacune ait la même échelle de valeurs. A la page suivante, un exemple de description XML permettant d'obtenir la position et la vitesse comme primitives à partir d'un vecteur (x, y, z).

```
<sys_calcul_prim>

  <!-- liste des noeuds en entrée du graphe -->
  <entrees>
    <noeud id="1">
     0-2            <!-- indique l'intervalle des indices du vecteur
                         en entrée. Les valeurs correspondant à ces
                         indices sont les données utilisées par ce
                        noeud pour effectuer son calcul -->
    </noeud>
 </entrees>

  <!-- noeud en sortie du graphe -->
 <sortie>
    <noeud id="3"/>
  </sortie>

 <!-- liste de tous noeuds avec le calcul qu'ils
       effectuent et leurs liaisons
  <noeuds nbNoeuds="3">
    <noeud id="1">
      <calcul>Id</calcul>
      <suivants>
        <noeud id="2"/>
       <noeud id="3"/>
      </suivants>
    </noeud>
    <noeud id="2">
      <calcul>Dérivée</calcul>
      <suivants>
        <noeud id="3"/>
      </suivants>
    </noeud>
    <noeud id="3">
      <calcul>NormExt</calcul> <!-- effectue une normalisation
                                    des valeurs -->
    </noeud>
  </noeuds>

</sys_calcul_prim>
```

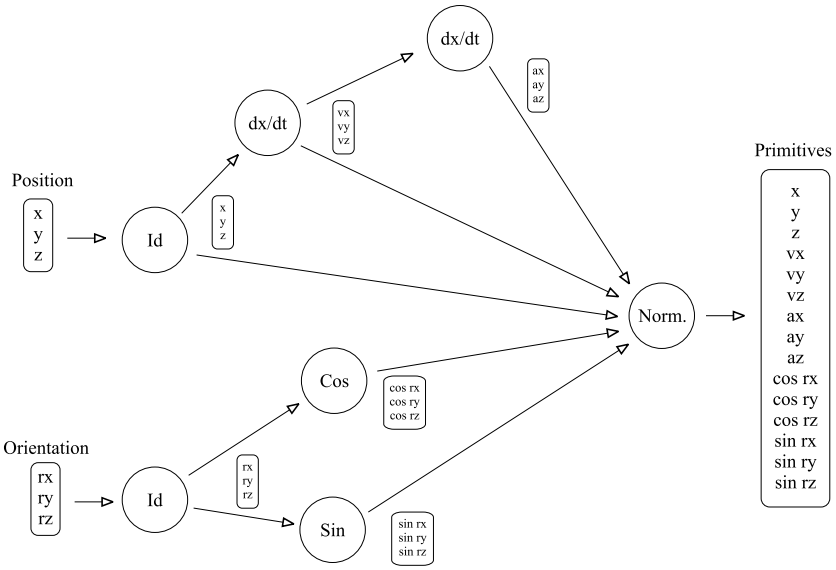

FIG. 5.6 – Exemple de calcul de primitives : position, vitesse, accélération , cosinus et sinus de l'orientation. Note : pour obtenir l'accélération on utilise la possibilité de cumuler plusieurs noeuds (ici deux noeuds dx/dt).

– **Un module effectuant la classification des gestes**. Deux parties sont présentes dans ce module : une partie où l'on va constituer un historique à partir des valeurs envoyées par le module de calcul de primitives, et une partie qui correspond au système de reconnaissance statistique. La taille de l'historique peut aller d'une trame dans le cas de postures[1] à plusieurs trames pour permettre la reconnaissance de gestes dynamiques. La taille maximale de l'historique est fonction de la fréquence des données reçues car il faut que le système puisse traiter les gestes en temps réel.
Ensuite, à partir de cet historique de vecteurs, on calcule un unique vecteur (moyenne, maximum, variation ...) qui est fourni au système statistique. Ce dernier effectue alors la classification du vecteur pour déterminer quel est le geste produit par l'utilisateur.

La partie interface va permettre à l'utilisateur d'interagir avec les différents modules du système pour modifier leurs paramètres. On va disposer de la liste des périphériques avec la fréquence de leurs événements (partie 2 sur la figure 5.7) , ce qui permet de sélectionner les périphériques voulus et de décider sur quel périphérique se synchroniser (on laisse la possibilité à l'utilisateur d'effectuer la fusion sur un autre périphérique que le plus lent). On va choisir, via l'interface, le fichier XML qui contient la description du graphe de calcul des primitives que l'on veut utiliser et les fichiers correspondants (partie 3 dans la figure 5.7). On va devoir indiquer la taille de l'historique des trames, fournir un fichier qui indique comment calculer un vecteur à partir de cet historique, et donner un autre fichier qui contient les structures correspondant aux différentes classes de gestes (vecteurs moyens et matrices de covariance) et aux fonctions linéaires permettant d'effectuer la classification des données à reconnaître (partie 1 dans la figure 5.7). On peut ainsi, à travers

[1] Le terme ***posture*** est employé ici pour désigner un geste figé.

cette interface, passer d'un utilisateur à un autre, changer de périphérique, ou encore passer d'une application à une autre sans avoir besoin de relancer le ClientReco et l'EVserveur.

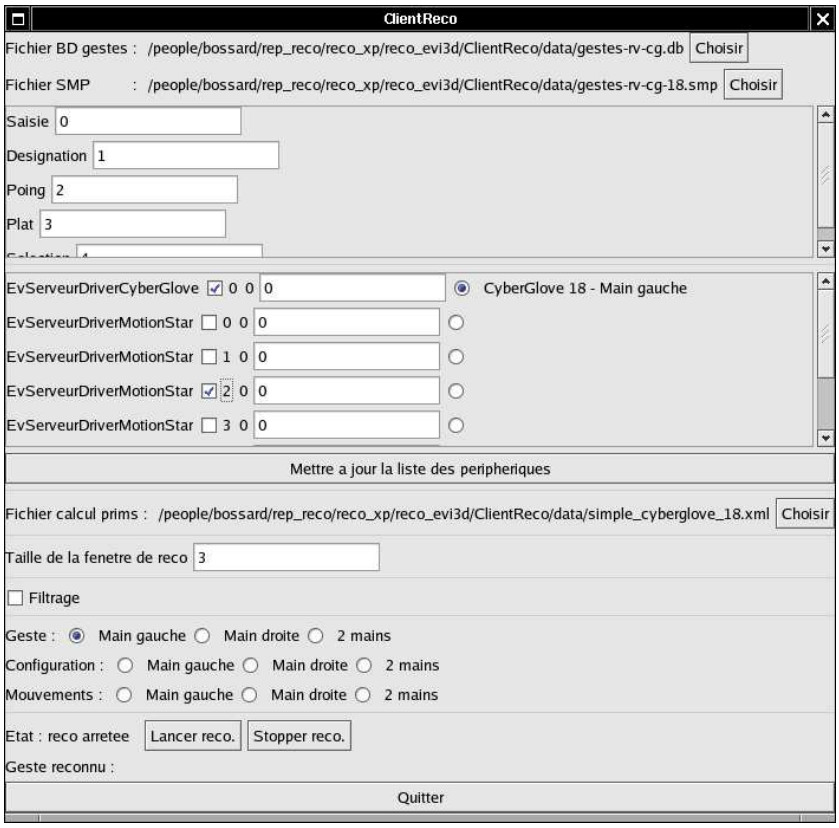

FIG. 5.7 – Interface de ClientReco.

5.2.3 Apprentissage

On vient de voir que l'on fournit au système de reconnaissance un fichier qui contient la représentation de chaque classe de gestes et les fonctions de classification. Ce fichier est créé lors de l'apprentissage qui se déroule en plusieurs étapes.

Acquisition d'exemples

La première étape consiste à enregistrer un ensemble d'exemples qui vont nous servir à effectuer l'apprentissage. Pour effectuer ces enregistrements, un client quasiment identique à ClientReco est utilisé, la seule différence se situe au niveau du module de reconnaissance remplacé par un module qui enregistre les trames acquises dans un fichier XML. Ce client est similaire à ClientReco pour que les données utilisées pour l'apprentissage aient les même caractéristiques que celles que l'on doit classifier (fusion de données et calcul de primitives identique). Ensuite, il va falloir identifier parmi toutes les données enregistrées uniquement celles qui sont utiles.

Segmentation des données

La deuxième étape consiste à segmenter les données pour isoler avec précision les exemples des données qui les entourent (il faut supprimer les valeurs correspondant à la coarticulation, aux pauses entre les exemples, etc). Une fois les différents segments identifiés, il faut les étiqueter pour effectuer un apprentissage supervisé.
Actuellement, on ne dispose pas de techniques dans le domaine du geste permettant d'effectuer de façon automatisée ces tâches de segmentation et d'étiquetage. C'est pourquoi, nous avons dû développer un outil permettant d'effectuer ces tâches manuellement.

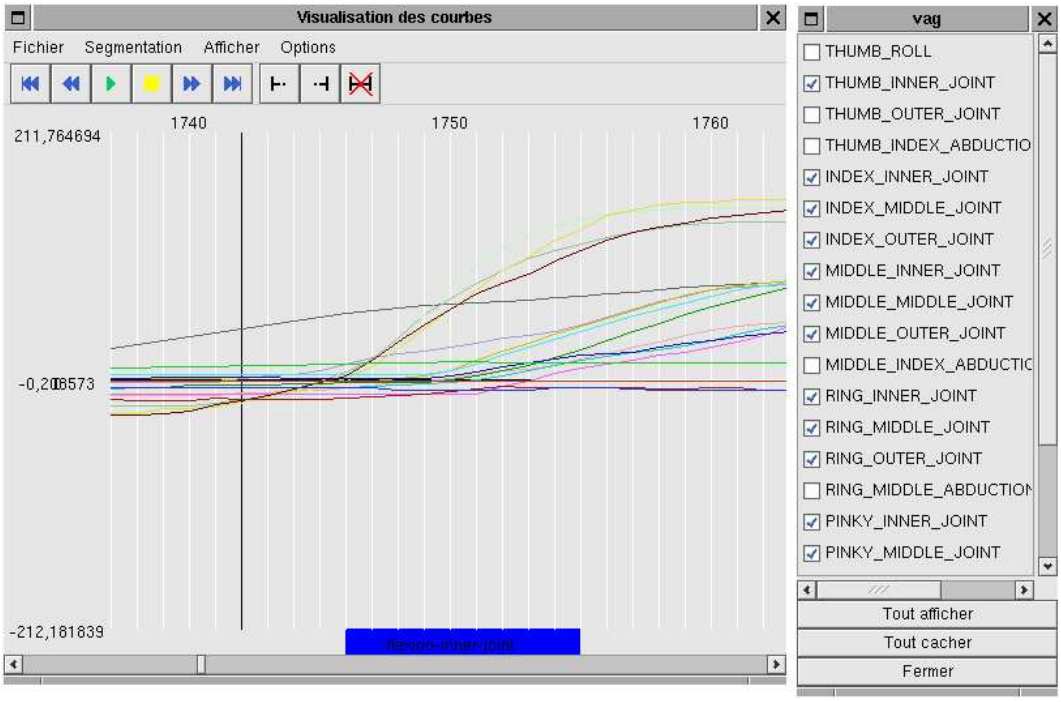

FIG. 5.8 – Le logiciel de segmentation VAG. L'exemple donné ici illustre le passage de la configuration *plat*, à la configuration *poing*. La segmentation effectuée indique la flexion de la première phalange des doigts qui précède ici celle des autres phalanges.

Cet outil permet de visualiser, sous forme de courbes, l'évolution temporelle des différentes primitives (voir figure 5.8). Il suffit ensuite d'indiquer dans ce graphique, le début, la fin et le nom de chaque exemple. Les exemples peuvent correspondre à une ou plusieurs trames suivant que le vocabulaire est constitué de postures ou gestes dynamiques. A noter que la taille de ces exemples doit être la même que celle de l'historique au niveau du système de reconnaissance. Lorsque tous les exemples ont été identifiés, on ajoute, dans le fichier XML qui contient les données, la description de ces exemples (numéro de trame de début et de fin et le nom du geste correspondant).

Apprentissage des exemples

L'apprentissage se déroule au travers de deux programmes. Le premier va effectuer la création des structures (vecteurs, matrices, etc) à partir d'informations fournies par l'utilisateur : nom

des gestes, nombre de primitives, taille de l'historique et calculs à effectuer sur cet historique. Le deuxième programme va effectuer la mise à jour des structures à partir des exemples contenus dans les fichiers de données XML. On peut ainsi effectuer l'enregistrement des exemples dans plusieurs fichiers séparés (par exemple un fichier pour chaque geste) et appeler plusieurs fois de suite le programme de mise à jour avec chacun des fichiers. Cette possibilité de répartir les exemples dans plusieurs fichiers est importante car elle permet d'utiliser plusieurs fois le même geste dans différents apprentissages.
Une fois que tous les exemples ont été appris via le programme de mise à jour, on dispose d'un fichier où toutes les classes sont correctement représentées et où l'on dispose de fonctions de classifications prêtes à fonctionner.

5.3 Evaluation du système de reconnaissance

Une fois que le système de reconnaissance ClientReco ainsi que tous les programmes nécessaires à l'apprentissage ont été créés, il a fallu évaluer la qualité de cet ensemble. Comme ClientReco, en dehors du cadre de l'architecture de reconnaissance de gestes bimanuels, est destiné à effectuer l'interprétation de gestes de la réalité virtuelle, nous avons évalué le système sur deux tâches courantes en réalité virtuelle : la manipulation directe et la désignation.

5.3.1 Le vocabulaire

Ces deux tâches nécessitent un vocabulaire gestuel très réduit, nous avons choisi de retenir uniquement deux postures statiques très simples : une posture "doigts serrés et main ouverte" pour la saisie, et une posture *index tendu* pour la désignation (voir figure 5.9).

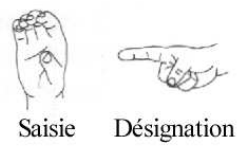

FIG. 5.9 – *Saisie* et *désignation.* (Extraits du dictionnaire de l'IVT)

On aurait pu aborder ces tâches de manipulation et de désignation sous d'autres aspects gestuels : gestes dynamiques, combinaison de la configuration et du mouvement ... Nous avons délibérément choisi de ne pas aborder ces aspects pour centrer l'évaluation plutôt sur les aspects logiciels du client de reconnaissance (intégration dans une application au travers de la plate-forme EVI3d, combinaison à d'autres modalités) et son intégration dans le dispositif immersif (qualité de la reconnaissance suivant les périphériques).

Un système de classification statistique ne permet pas de savoir si un geste, fourni en entrée au système, fait partie du vocabulaire appris ou non. Une solution consiste à utiliser des calculs de distance entre le geste reconnu et la classe dans laquelle il été classifié (par exemple la distance géométrique ou de Mahalanobis), ou à utiliser des probabilités sur le geste que l'on vient de recon-

naître. Cette solution n'est pas toujours simple à mettre en oeuvre, car il faut pouvoir estimer de manière fiable le seuil permettant de décider de la véracité du geste, et parfois les calculs effectués dépendent fortement de l'apprentissage, ce qui nuit à la robustesse du système.
Une autre solution consiste à rajouter au vocabulaire de base d'autre gestes qui permettent de couvrir les plages de valeurs que nos gestes de base ne couvrent pas. On ne peut avoir alors de gestes n'appartenant pas au vocabulaire reconnu par le système. Bien évidemment, cette solution ne peut pas être utilisée avec n'importe quel vocabulaire.

Nous avons choisi d'utiliser cette deuxième solution qui convient bien à notre vocabulaire. Aux gestes *saisie* et *désignation*, nous avons donc rajouté les gestes *plat* et *poing* (figure 5.10) qui permettent de couvrir les valeurs correspondant aux doigts fortement fléchis et très peu fléchis. Les gestes *saisie* et *désignation* couvrent eux les valeurs intermédiaires.

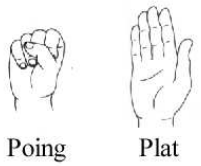

FIG. 5.10 – *Plat* et *poing*. (Extraits du dictionnaire de l'IVT)

Ces deux gestes *plat* et *poing*, vont ensuite être considérés comme une classe *autre* au niveau de l'application. Le passage à cette classe *autre* permet à l'application de détecter la fin d'une saisie ou d'une désignation.

5.3.2 Choix des primitives

Une fois que l'on connaît l'ensemble des gestes que l'on veut reconnaître, il va falloir trouver les primitives qui permettent de caractériser et différencier les gestes. Une primitive correspond à une caractéristique des gestes que l'on obtient à partir des données capturées. Pour des gestes dynamiques, ces primitives peuvent correspondre à des calculs assez évolués (vitesse angulaire des articulations, amplitude des mouvements des doigts, etc). Dans le cas de postures, les valeurs brutes reçues depuis les gants numériques peuvent être directement utilisées en tant que primitives.
Nous disposons de deux types de gants numériques pour effectuer la capture des données : un DataGlove 5 de chez 5DT et un CyberGlove 22 de chez Immersion. Le DataGlove 5 ne permet d'avoir comme information que la flexion globale de chaque doigt. Donc pour ce gant, la problématique du choix des primitives ne se pose pas vraiment, il nous manque même des données pour pouvoir distinguer certains mouvements des doigts. Par contre, le CyberGlove couvre tous les degrés de liberté de la main : les articulations des doigts, les abductions, la paume et le poignet, il faut alors déterminer quelles sont les informations nécessaires et suffisantes.
Nous allons maintenant examiner divers choix de primitives et ce qui a conduit à effectuer ces choix.

- La solution la plus simple consiste à prendre comme primitives tous les capteurs du gant numérique, ce qui représente 22 primitives pour le modèle CyberGlove. Un premier problème apparaît avec cet ensemble de primitives. Du fait du grand nombre de primitives, il faut au

minimum une dizaine d'exemples pour faire l'apprentissage du système de reconnaissance. Alors que pour les trois autres ensembles de primitives décrits ci-dessous, quatre exemples suffisent. D'autre part, il apparaît clairement que de nombreuses primitives sont inutiles. On sait par exemple, que pour la plupart des personnes, l'articulation de la dernière phalange dépend de l'articulation précédente.

- Le deuxième ensemble de primitives a été choisi en fonction des caractéristiques des gestes. Lors du geste de désignation, on a l'index tendu, donc on choisit d'utiliser les trois articulations de l'index comme primitives. Pour la saisie, il faut au moins une information par doigt, de plus on a plutôt tendance à saisir principalement à l'aide du pouce, de l'index et du majeur ; on va donc utiliser les 3 articulations du pouce, les trois articulations du majeur et les articulations de la première phalange pour l'annulaire et l'auriculaire. Cela nous fait donc un total de 11 primitives.

- Le troisième ensemble de primitives a été choisi pour vérifier que l'on ne pouvait pas réduire le nombre de primitives au niveau du pouce, de l'index et du majeur ; on va donc choisir uniquement l'articulation de la première phalange pour chaque doigt. On obtient alors un total de 5 primitives qui sont relativement similaires à ce que l'on aurait eu comme données avec un gant DataGlove5.

- Pour le quatrième ensemble de primitives, on est reparti du deuxième ensemble et on a analysé plus en détail les variations des différentes articulations et cela plus particulièrement pour le geste de désignation. En effet, c'est le seul geste de notre vocabulaire pour lequel tous les doigts n'ont pas la même posture.

  Dans les figures 5.11 et 5.12, on peut voir les courbes des flexions des doigts pour les gestes *poing* et *désignation*. Les courbes de la première figure correspondent aux trois articulations du pouce, de l'index et du majeur. Les courbes de la seconde figure correspondent aux trois articulations du pouce, l'abduction pouce-index, les articulations de la première et de la deuxième phalange de l'index et la première phalange du majeur.
  On peut voir, lorsque l'on n'utilise pas les abductions, que seules les valeurs des articulations de l'index permettent de distinguer efficacement les deux gestes. Or, les valeurs de l'index pour une *désignation* sont proches de celles d'un geste *plat* et d'une *saisie large*[2]. Par contre, l'abduction index-majeur permet de bien distinguer la *désignation* du geste *poing*, mais aussi de bien la distinguer des gestes *plat* et *saisie*.
  C'est pourquoi pour le quatrième ensemble on aura les 11 primitives suivantes : les trois articulations du pouce, les deux premières phalanges de l'index et du majeur, l'abduction pouce-index, l'abduction index-majeur et la première phalanges de l'annulaire et de l'auriculaire.

[2] Une saisie "large" correspond à un geste de saisie pour lequel la flexion des doigts est faible. Ce geste est employé lorsque l'utilisateur effectue la saisie d'un objet de taille importante.

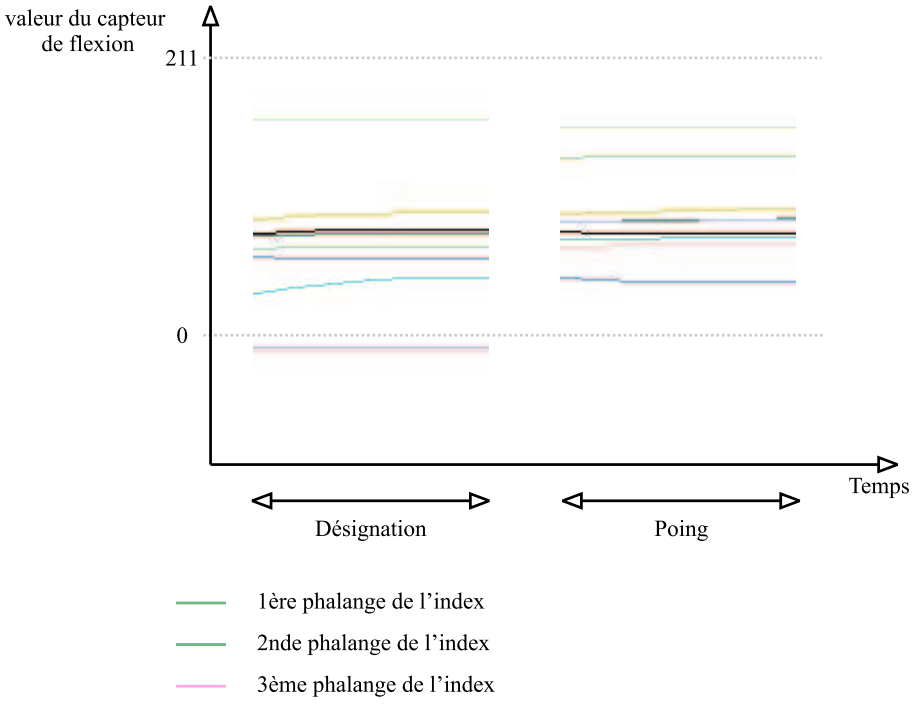

FIG. 5.11 – Articulations du pouce, de l'index et du majeur pour *désignation* et *poing*

5.3.3 Expérimentation des primitives

Pour comparer ces différents ensembles de primitives nous avons fait quatre apprentissages avec les mêmes exemples (entre 10 et 12 pour chacune des quatre classes de gestes) ; chacun des apprentissages correspondant à un ensemble de primitives. Nous avons ensuite fait fonctionner simultanément quatre systèmes de reconnaissance de gestes à partir de ces apprentissages pour comparer les gestes reconnus au geste produit. Pour chaque classe de geste, une douzaine de gestes ont été produits. Deux ou trois gestes parmi les douze étaient des cas "limites" permettant de tester la robustesse des ensembles de primitives. Le tableau 5.1 donne les résultats obtenus.

	Nb. de primitives	Nb. gestes corrects	% gestes corrects
Ensemble 1	22	34	70,8
Ensemble 2	11	44	91,7
Ensemble 3	5	39	81,25
Ensemble 4	11	47	97,9

TAB. 5.1 – Taux de reconnaissance pour les différents ensembles de primitives testés sur 48 gestes.

Le résultat obtenu pour le premier ensemble de primitives est très surprenant. Même si l'on s'attendait à ce que les résultats obtenus pour cet ensemble de primitives ne soient pas optimum, ils sont vraiment très bas. L'une des causes peut être dû au fait que cet ensemble de primitives a besoin de plus d'exemples pour l'apprentissage que les autres. Il est alors probable que le nombre d'exemples ait été insuffisant dans cette expérimentation.
L'ensemble 3 a un score plutôt satisfaisant étant donné le nombre de primitives utilisé. Toutefois, il

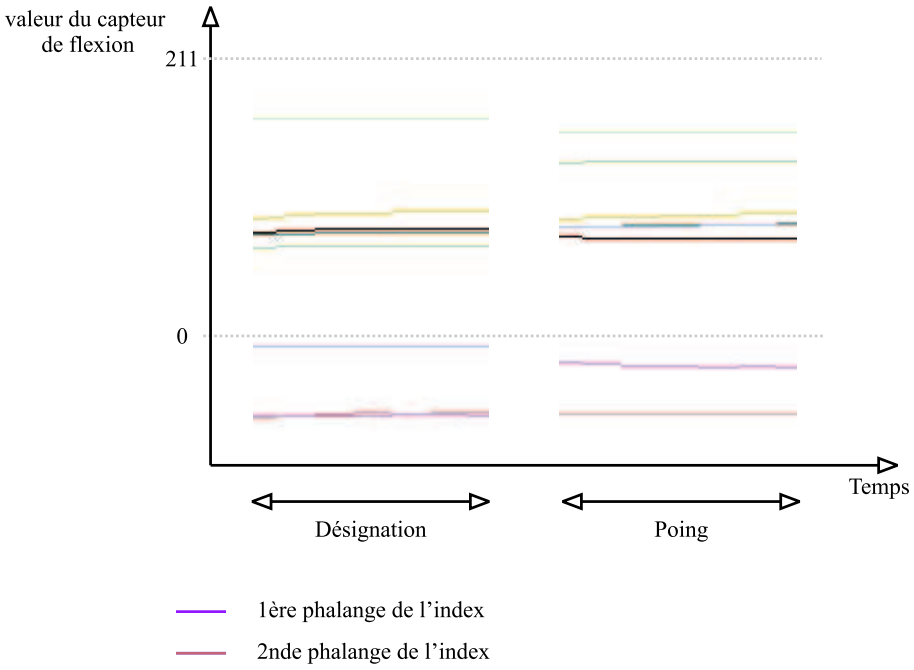

FIG. 5.12 – Articulations du pouce, des deux premières phalanges de l'index, de la première phalange du majeur et de l'abduction pouce-index pour *désignation* et *poing*

est largement en deçà des scores des ensembles 2 et 4. On remarque dans ces résultats que le choix des abductions pour l'ensemble 4 par rapport à l'ensemble 2 a été judicieux car il permet d'avoir une amélioration dans le taux de reconnaissance.

On constate, suite à cette expérimentation, que le choix des primitives n'est pas à prendre à la légère, car il peut influencer de manière importante le taux de reconnaissance d'un système statistique. D'autre part, on constate qu'un grand nombre de primitives n'est pas forcément synonyme d'une bonne classification des gestes, car du fait du grand nombre de primitives on est obligé de faire un apprentissage plus long et certaines primitives ont tendance à être redondantes.

5.3.4 Utilisation dans des applications de réalité virtuelle

Une première utilisation du système de reconnaissance a été faite avec une application test (figure 5.13) visant à utiliser le noyau de modélisation géométrique OpenCASCADE dans un environnement immersif. Pour cette application, le geste a été utilisé comme moyen d'interaction pour manipuler, désigner et sélectionner les objets. Pour l'opération de sélection, un geste spécifique (similaire au geste de désignation mis à part le pouce qui est tendu) a été rajouté au vocabulaire.

L'interfaçage de ClientReco avec l'application n'a pas posé de problème du fait de l'architecture de la plate-forme EVI3d. Le client envoie à l'application pour chaque geste reconnu un évènement qui contient un identifiant correspondant au geste. Par contre, cette méthode présente l'inconvénient d'obliger l'application à avoir une liste fixe d'identifiants pour les différents gestes. Au niveau du client, l'utilisateur a la possibilité de donner ces différents identifiants afin que le client et l'application puissent "dialoguer".

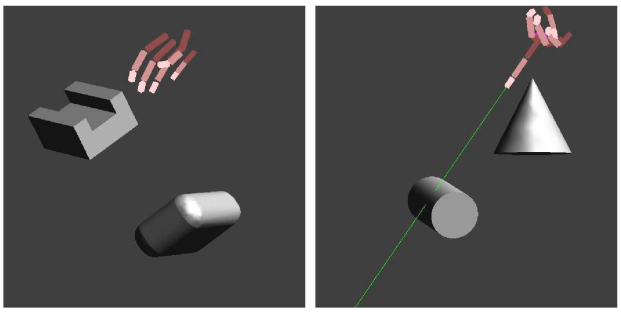

FIG. 5.13 – Application de test pour la création d'objets 3d (à droite : désignation du cylindre). (LIMSI-CNRS)

Cette première approche gestuelle en réalité virtuelle nous a permis de constater un certain nombre d'inconvénients propres à ce domaine. Tout d'abord, le fait que les objets virtuels n'offrent aucune consistance pose quelques problèmes lorsqu'ils sont manipulés. Par exemple, on peut effectuer un relâchement involontaire de l'objet du fait que l'on modifie inconsciemment sa posture de saisie en cours de manipulation. D'autre part, la saisie de l'objet pose des problèmes de détection des doigts par rapport aux objets. Lorsque l'on effectue une saisie, il faut s'assurer que l'objet se situe entre les doigts et non pas à dix centimètres de ceux-ci.
Dans un dispositif de réalité virtuelle, il peut y avoir plusieurs utilisateurs qui utilisent simultanément la modalité gestuelle, et plusieurs utilisateurs peuvent se succéder pour une même application. Le problème de la variabilité inter-utilisateur des gestes se pose alors. Normalement, pour chaque utilisateur, on doit effectuer une calibration du gant numérique et un apprentissage. Dans notre cas de vocabulaire très simple, nous nous attendions à ne pas avoir d'apprentissage à effectuer pour chaque utilisateur, mais uniquement une calibration. Si certains apprentissages marchent pour un certain groupe de personnes, ils ne fonctionnent pas pour tous. Cela s'explique par le fait que, pour différentes tailles et morphologies de mains, les capteurs d'un gant numérique ne se retrouvent pas positionnés de la même manière par rapport aux articulations.

Au sein du laboratoire LIMSI-CNRS de nombreux travaux portent sur les aspects multimodaux en interaction homme-machine. En particulier, on s'intéresse à l'usage des canaux naturels de communication en réalité virtuelle : la voix et le geste. C'est pourquoi, suite au premier essai, le système ClientReco a été utilisé dans une application de démonstration multimodale (figure 5.14) créée par Damien Touraine dans le cadre de sa thèse [Touraine, 2003]. Cette application a pour but de démontrer les différentes interactions possibles en combinant le geste et la parole. Le geste de saisie permet d'effectuer une manipulation directe de tout objet qui est à portée de main. Le geste de désignation combiné aux commandes vocales (par exemple *prend cet objet*) permet d'interagir avec les objets situés hors de portée.

La liaison entre le client et l'application se passe de la même manière que dans la première expérimentation. La seule différence consiste à utiliser le temps de début et de fin du geste en plus de l'identifiant du geste. En effet, pour les interactions combinant le geste et la parole on a besoin de connaître précisément à quel instant a eu lieu un geste. Par exemple, dans une phrase du type

FIG. 5.14 – Démonstrateur "LogoMultimodal" de la plate-forme EVI3d mettant en oeuvre notre système de reconnaissance de gestes. (LIMSI-CNRS)

Met cet objet dans ma main, il va falloir détecter qu'au moment où est prononcé *cet*, on effectue simultanément un geste de désignation.
A noter que cette application de démonstration a été très souvent utilisée avec divers utilisateurs, ce qui nous a permis de vérifier la robustesse et la stabilité du système ClientReco qui est donc bien plus qu'un simple module de reconnaissance faisant partie d'une architecture expérimentale.

5.3.5 Postures et gestes dynamiques

Pour effectuer les tests du système ClientReco, nous avons utilisé des gestes basés sur des configurations statiques de la main. Mais les interactions gestuelles en réalité virtuelle ne se résument pas à ce type de gestes. Nous allons maintenant discuter des avantages de chacun des différents types de signes et exposer nos différents tests concernant les gestes dynamiques.

Configuration statique

Les gestes que nous avons utilisés jusqu'à maintenant sont des configurations statiques de la main. Le choix d'utiliser des postures statiques peut sembler simpliste et un peu éloigné des systèmes actuels qui permettent la gestion de gestes dynamiques et de la coarticulation. Toutefois, ce choix a été motivé entre autre par le fait que le vocabulaire est suffisamment réduit et simple pour ne pas avoir besoin de gérer les problèmes de coarticulation. Les postures permettent d'avoir une interprétation quasi instantanée d'un geste contrairement aux gestes dynamiques qui ont en général besoin d'être interprétés sur plusieurs trames d'affilée. Mais surtout, les configurations statiques permettent un apprentissage très facile. En effet, on n'a pas besoin de faire de segmentation pour de tels gestes car le geste correspond à la trame courante. Un utilisateur quelconque peut donc

effectuer l'apprentissage du système, tout simplement en donnant successivement les différentes configurations constituant le vocabulaire. Il n'a pas besoin de se préoccuper du fait qu'un geste commence à telle trame et finit à telle autre.
Bien évidemment, cette approche a une contrepartie, et en particulier les postures ont le désavantage de ne représenter qu'une partie du geste réellement effectué. Ainsi parfois, une toute petite différence de la flexion des doigts peut entraîner le geste dans une autre classe du système de reconnaissance. Cela est particulièrement dérangeant lorsque l'on manipule un objet que l'on "tient" en main et que tout à coup, l'objet est "lâché" sans qu'on l'ai souhaité.

Configuration dynamique

L'utilisation de configurations dynamiques pour la saisie permet d'éviter cet inconvénient. On va alors utiliser le moment où l'on ferme les doigts pour débuter une saisie et le moment où l'on ouvre les doigts pour achever la saisie. L'avantage de ce type de gestes est que l'on n'a pas besoin de se préoccuper de la manière dont l'objet est tenu (nombre de doigts utilisés, forme de la main...) entre la fermeture et l'ouverture des doigts.
Nous avons expérimenté ces gestes en utilisant comme primitives la variation de la flexion des doigts au cours du temps. On se ramène ainsi à une information statique, dans le sens où c'est la variation angulaire qui cette fois ci est statique. Les premiers tests effectués avec trois gestes (fermeture, ouverture et statique) ont été plutôt encourageants, mais cela constitue un vocabulaire très limité pour la réalité virtuelle. Les primitives utilisées pour un gant CyberGlove correspondent à la dérivée des articulations du pouce et des deux premières articulations des autres doigts. Pour un DataGlove5, nous utilisons la dérivée de la flexion donnée pour chaque doigt. Nous avons pu remarquer que ce type de geste est très peu sensible à la variabilité inter-utilisateur du fait que l'on ne se base pas du tout sur la forme de la main.
Nous avons ensuite essayé d'ajouter des configurations statiques (désignation, plat...) aux gestes *fermeture* et *ouverture*. Mais plusieurs problèmes au niveau des primitives se sont alors posés. Tout d'abord, il faut avoir un bon équilibre entre les primitives permettant de caractériser les configurations statiques (primitives basées sur la forme de la main) et les primitives permettant de caractériser les configurations dynamiques (primitives basées sur le mouvement des doigts). En effet, si un des deux ensembles est de taille plus importante, il va avoir un poids plus important au niveau du processus de classification.
Le second problème qui se pose est la normalisation des primitives. En effet, pour les primitives correspondant à la dérivée de la flexion d'une articulation, il n'y a pas de maximum clairement défini. Il devient alors difficile de donner la même échelle de valeurs pour toutes les primitives. Nous avons testé un calcul de primitives où la normalisation se fait de manière dynamique, c'est à dire que l'utilisateur commence par effectuer plusieurs fermetures et relâchements pour déterminer les minima et les maxima. Mais ce système n'a pas donné de résultat concluant du fait d'une trop grande variation de ces calculs entre différentes utilisations du système de reconnaissance. Il est probable qu'une autre approche pour la modélisation du mouvement permettrait d'éviter ce problème. Nous n'avons pas approfondi plus avant ce problème qui est une des perspectives d'amélioration du système.

Mouvement

Le mouvement est par essence même une composante des gestes dynamiques. Nous nous sommes intéressés aux gestes basés sur le mouvement dans l'optique d'effectuer la description de formes à l'aide de droites, d'arcs et de cercles. Le principal obstacle, dans l'utilisation de ces gestes pour une application de réalité virtuelle, est leur temps d'interprétation. En effet, un mouvement couvre en général un grand nombre de trames, ce qui est incompatible avec une réponse en temps réel qui nécessite de petits historiques de trames. Il est alors impossible de reconnaître en temps réel un geste décrivant un cercle. Même la différenciation entre un mouvement linéaire et un arc devient difficile. En effet, la proximité des points servant à analyser le mouvement ne permet pas de différencier un mouvement linéaire d'un mouvement ayant un très faible rayon de courbure. Nous reviendrons dans le chapitre suivant, qui traite la reconnaissance de gestes en langue des signes, sur ces problèmes de différenciation arc/droite.
Du fait que nous utilisons de petits historiques temporels pour les autres gestes d'interaction, nous ne nous sommes pas penchés plus en avant sur ces gestes utilisant le mouvement. Si nous avions à le faire, on décomposerait alors le geste en deux paramètres (i.e. un ClientReco pour la configuration et un pour le mouvement) ce qui éviterait ce problème de conflit temporel.

5.3.6 Bilan sur le module de reconnaissance

Dans les expérimentations que nous avons faites pour évaluer les capacités du client de reconnaissance de gestes, nous avons abordé divers types de gestes. Concernant les postures, le système de reconnaissance nous donne toute satisfaction sur les points que nous voulions résoudre vis à vis de la réalité virtuelle (temps réel, robustesse, modularité). Les résultats concernant les gestes dynamiques sont un peu plus mitigés et indiquent que des mises aux points sont encore nécessaires. Néanmoins, l'utilisation de postures suffit amplement à effectuer les tâches de base en réalité virtuelle : manipulation et désignation/sélection. Nous considérons donc le système comme étant opérationnel.
Maintenant que nous avons un module de reconnaissance opérationnel pour notre architecture de reconnaissance de gestes bimanuels, nous allons pouvoir passer aux autres parties de l'architecture dans le but de traiter des gestes bimanuels ou des relations spatiales.

5.4 Module de comparaison

En réalité virtuelle, on rencontre rarement des gestes bimanuels (i.e. geste faisant intervenir les deux mains) dans les tâches de manipulation d'objets. Ces gestes sont déjà plus courants dans les applications qui nécessitent l'utilisation de gestes descriptifs où les deux mains sont utilisées pour souligner des aspects de symétrie. Par exemple, lorsque l'on représente les deux côtés d'un cube, le geste utilisé consiste à utiliser les deux mains avec la même configuration, les emplacements en vis à vis et une orientation opposée. Cela permet de souligner l'aspect parallèle des deux faces.
Comme en langue des signes, on doit alors être capable de distinguer les gestes monomanuels simultanés des gestes bimanuels. L'architecture proposée en section 3.1 propose de traiter ce problème en utilisant trois systèmes de reconnaissance et un module de comparaison (figure 5.15).

Le module effectuant la comparaison est alors implémenté sous la forme d'un client EVI3d qui a pour source de données trois systèmes ClientReco (voir figure 5.15). L'architecture de ce client

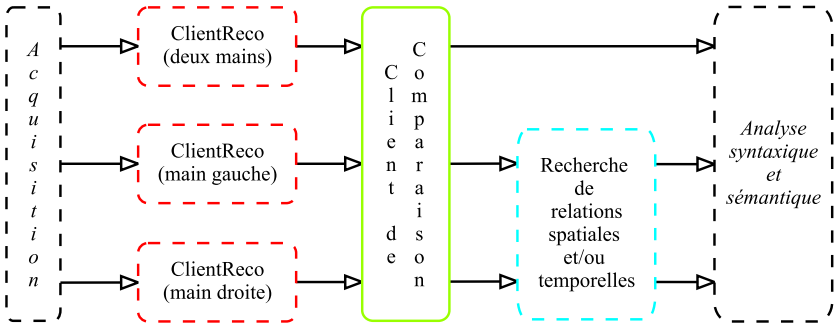

FIG. 5.15 – Place d'un client de comparaison dans l'architecture de reconnaissance de gestes bimanuels.

est fortement similaire à ClientReco (figure 5.16).

La partie qui effectue la sélection des périphériques dans le client de reconnaissance de geste effectue ici la sélection des trois ClientReco qui correspondent aux gestes de la main gauche, de la main droite et des deux mains. Pour la partie synchronisation des évènements, nous avons choisi de considérer que ceux ci correspondent à des postures (donc pas de durée). Il suffit donc d'attendre que chacun des systèmes de reconnaissance émette un geste avant de lancer le processus de comparaison. Dans le cas de gestes dynamiques (qui ont une durée temporelle), il faudrait analyser les temps de début et de fin des gestes pour voir les correspondances et comparer ceux qui partagent une même période temporelle. Pour effectuer la comparaison, nous utilisons des distances entre le vecteur de données et le vecteur moyen de la classe reconnue. En plus de ces scores, il peut être intéressant de prendre en compte le contexte de la scène virtuelle. Par exemple, si les deux mains sont situées au niveau d'un même objet, il est probable que l'on soit en présence d'un geste bimanuel, tandis que si chaque main est à proximité d'un objet différent, il y a probablement deux gestes monomanuels.

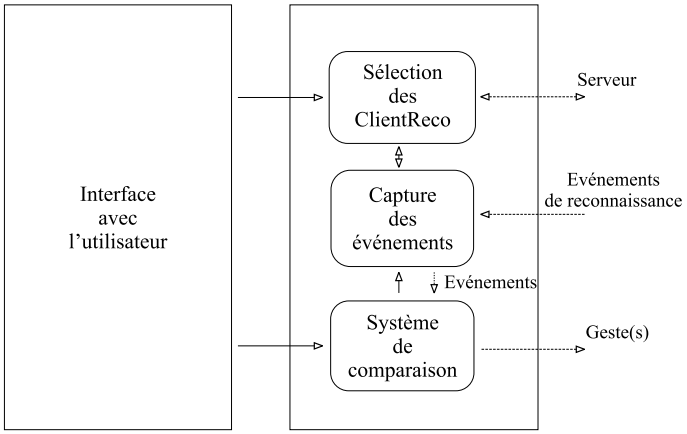

FIG. 5.16 – Architecture du client de comparaison de gestes

Dans notre expérimentation (tâches de manipulation directe, de désignation et description de

scène), nous n'avons pas utilisé de gestes bimanuels. C'est pourquoi le processus de comparaison n'a pas été évalué dans le domaine de la réalité virtuelle. C'est dans le chapitre suivant, qui aborde le domaine de la langue des signes, que le choix et l'évaluation des méthodes de calcul de scores sont étudiés.

Nous allons passer maintenant à l'autre aspect de l'utilisation des deux mains, à savoir la détection de relations entre deux gestes.

5.5 Module de détection de relations entre les mains

En réalité virtuelle, lorsque l'on utilise les deux mains dans des tâches de manipulation, il arrive souvent que les deux mains soient en relation avec un rôle main dominante/main dominée. Par exemple, la main dominée tient l'objet pendant qu'une arête de l'objet est manipulée par la main dominante. Mais la relation établie entre les mains ne nécessite aucune interprétation car elle fait intervenir des propriétés sensori-motrices du geste (la main dominée donne une référence spatiale à la perception proprioceptive[3] de la main dominante qui effectue une action en fonction de cette référence).
Par contre, lorsque l'on utilise des gestes utilisant des propriétés sémiotiques (description de la forme d'objets, ou de la disposition d'objets), une interprétation des relations entre les mains devient nécessaire. Nous avons choisi d'illustrer cet aspect à travers l'expérimentation en réalité virtuelle de mécanismes permettant la description de scènes en langue des signes.

5.5.1 Expérimentation du transfert situationnel en réalité virtuelle

Nous avons vu précédemment que la langue des signes dispose d'un mécanisme permettant d'effectuer la description de scènes spatiales : il s'agit des transferts situationnels. Avec ce mécanisme, on peut indiquer qu'un objet se situe sur un autre, que plusieurs objets sont disposés en parallèle, etc. Notre idée est d'appliquer ce mécanisme dans le domaine de la réalité virtuelle afin de pouvoir décrire des scènes virtuelles. Cela peut se faire au moment de la création de la scène virtuelle, ou plus tard lorsque l'on souhaite déplacer des objets lointains et qui sont en relation spatiale avec d'autres objets [Bossard, 2005].
Ce type de gestes n'appartient pas au domaine habituel des gestes utilisés par une personne. C'est pourquoi, l'utilisateur va devoir passer un certain temps pour "apprendre" ce mécanisme gestuel nouveau pour lui.

L'expérimentation de ce mécanisme en réalité virtuelle a été effectué par Benjamin Colin dans le cadre d'un stage de DEA. [Colin, 2004]. Ce travail a été fait en utilisant l'architecture qui a été proposée en section 3.1. Nous allons maintenant voir quel est le vocabulaire et la structure syntaxique utilisés pour cette expérimentation.

Gestes et phrases pour des descriptions de scènes

L'expérimentation se place dans une application de réalité virtuelle où sont présents des objets nous environnant dans le monde réel. Une table et un verre ont été choisis comme objets pour tester

[3]La perception proprioceptive correspond, entre autres, aux informations retournées par nos articulations, ce qui permet de situer nos mains sans l'aide de la vision.

les différentes interactions possibles. Nous avons besoin alors de gestes capables de représenter ces deux entités de manière iconique (proformes dans le cas de phrases de la langue des signes), nous utiliserons l'appellation de ***proformes*** pour ces gestes par la suite. A la table va correspondre une proforme ayant une configuration *plate*, et au verre une proforme utilisant la configuration C (figure 5.17).

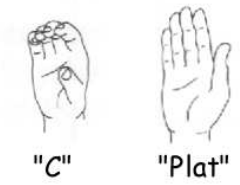

FIG. 5.17 – Les proformes représentant le verre (C) et la table (*Plat*). (Extraits du dictionnaire de l'IVT)

Le choix de ces proformes a été effectué bien évidemment en fonction de la relation iconique qui les relient à l'objet qu'ils représentent. Ensuite, nous avons besoin d'un geste de désignation permettant d'indiquer les objets auxquels on va vouloir faire référence. Pour cela, nous utilisons le geste habituel où l'index est tendu.
Comme pour les tests qui ont été effectués avec le geste de saisie et de désignation, on va rajouter des gestes permettant de "couvrir" les différentes configurations non couvertes par les configurations *plat*, C et *désignation*. On va ajouter le geste *poing* et un geste *repos* qui correspond à une configuration intermédiaire entre *plat* et C où les doigts sont "relâchés".
Ce vocabulaire peut être utilisé aussi bien par la main gauche que la main droite, toutefois le geste de désignation est plutôt destiné à être utilisé par la main dominante.

L'enchaînement des gestes va suivre des schémas syntaxiques bien précis afin de faciliter l'interprétation des références iconiques et des relations spatiales entre entités.
Pour créer une référence iconique, deux gestes vont être utilisés : tout d'abord un geste de désignation va être effectué vers un objet de la scène pour indiquer qui est concerné par la référence, ensuite une proforme est effectuée (soit par la même main, soit par l'autre main) pour représenter l'objet (voir figure 5.18). Une fois que la proforme garde une position fixe et que l'on stoppe le geste, cette position va servir de référence à l'objet représenté par la proforme.

Pour utiliser ces références spatiales, il suffit de produire la même proforme, et de la positionner à l'emplacement de la référence (figure 5.19). La proforme représente ensuite l'objet qui a été mis à cet emplacement, on peut alors utiliser cette proforme pour fournir une autre référence spatiale ou une relation spatiale par rapport à une autre proforme.

Lorsque chaque main effectue une proforme, on peut indiquer une relation spatiale entre elles. Il faut tout d'abord effectuer la relation spatiale avec les mains (une main au dessus de l'autre pour la relation *sur*, une main à droite de l'autre pour indiquer la relation *à droite*, etc), puis tenir cette disposition pendant un instant (figure 5.20).

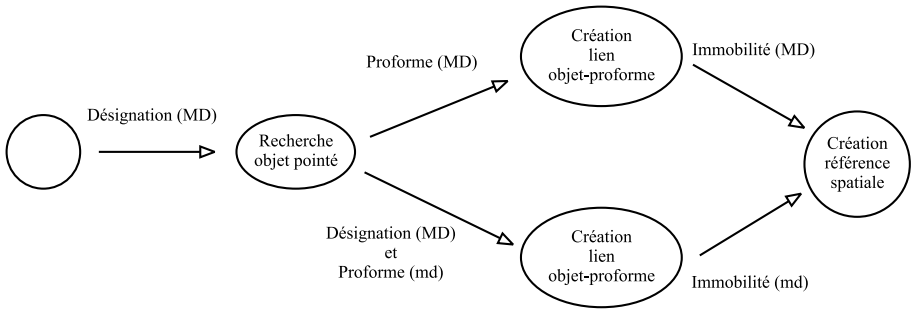

FIG. 5.18 – Enchaînement de gestes pour lier un objet à une proforme, puis ajouter cette proforme au contexte spatial (MD : main dominante, md : main dominée).

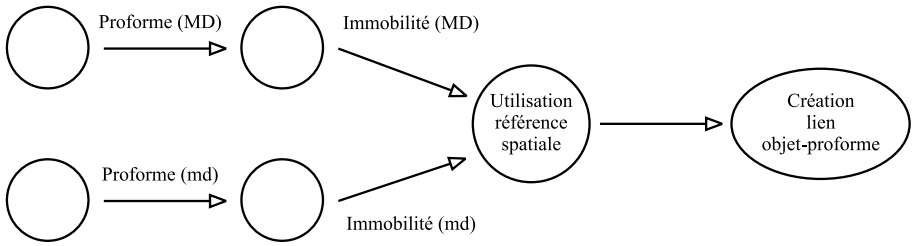

FIG. 5.19 – Enchaînement à effectuer pour utiliser une référence spatiale.

Si l'on veut par exemple indiquer que *le verre est sur la table*, on va tout d'abord désigner la table de la main droite, effectuer la proforme *plat* avec l'autre main. Puis désigner le verre de la main droite et effectuer avec la main droite la proforme C. Enfin positionner la proforme C au dessus de l'autre proforme et tenir la position pendant un instant.

Maintenant que nous avons vu quel est le vocabulaire et le type de phrases que nous devons interpréter, nous allons passer à la description du système permettant d'interpréter ce type de gestes.

5.5.2 Description du module d'interprétation spatial

Contrairement aux autres modules de l'architecture, le module de détection de relations ne va pas être implémenté sous la forme d'un client. En effet, du fait du lien existant entre les proformes et les objets de la scène virtuelle, le module est directement intégré à l'application qui gère la scène. Une implémentation sous forme de client impliquerait un dialogue constant entre l'application et le client pour maintenir une représentation à jour de la scène au niveau du client.
Dans le vocabulaire qui a été choisi, il n'y a aucun geste bimanuel, les modules de reconnaissance de gestes vont donc être directement raccordés au module de détection de relations spatiales. La figure 5.21 donne une vue de cette architecture simplifiée où l'on n'a pas besoin du module de comparaison du fait de l'absence de gestes bimanuels.

Nous avons vu dans la section 3.1 que le module de détection effectue plusieurs tâches : tout d'abord, il effectue une resynchronisation entre les gestes reconnus, puis il recherche les points de synchronisation entre ces gestes, et enfin il recherche des relations spatiales (ou temporelles) au

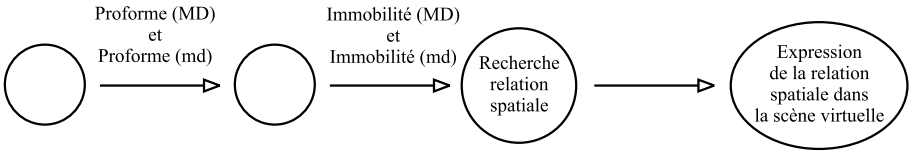

FIG. 5.20 – Enchaînement nécessaire pour exprimer une relation spatiale (MD : main dominante, md : main dominée).

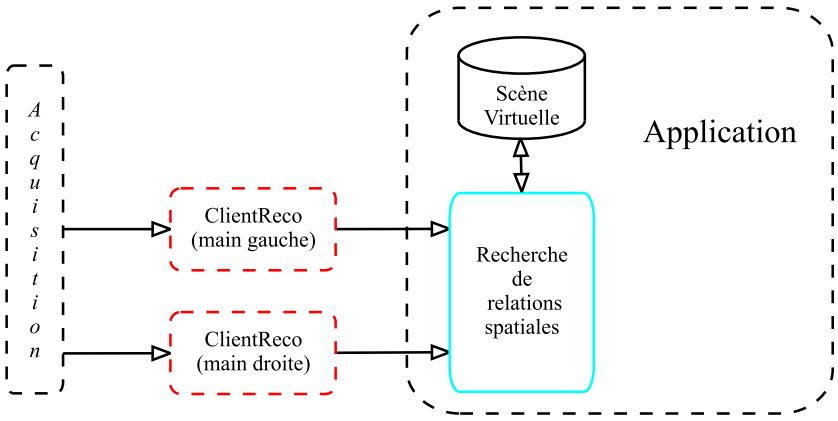

FIG. 5.21 – Place du module de détection de relations spatiales dans l'architecture de reconnaissance de gestes.

niveau de ces points de synchronisation. L'étape de resynchronisation est nécessaire du fait que les gestes reconnus pour la main gauche et la main droite n'ont pas forcément la même durée et sortent donc des systèmes de reconnaissance à des moments différents. Dans l'expérimentation qui a été menée, les clients de reconnaissance ont été paramétrés avec un historique d'une trame du fait qu'il s'agit de gestes correspondant à des postures statiques. Il n'y a donc pas lieu d'effectuer une resynchronisation des gestes au niveau du module de recherche de relations spatiales, car les gestes parviennent au module avec le même décalage temporel qu'au moment de leur production. Il ne reste donc plus qu'à effectuer la recherche de points de synchronisations et de relations spatiales.

Par contre, une nouvelle étape vient se rajouter en supplément de ces tâches de recherche. En langue des signes, la mise en rapport d'un objet avec une proforme s'effectue via des mécanismes syntaxiques (ordre des signes ou utilisation de la scène de narration). Dans notre adaptation des transferts situationels en réalité virtuelle, on doit explicitement faire le lien par une désignation de l'objet. Une fois l'objet désigné, soit on utilise directement la proforme dans une relation spatiale, soit on le place dans une scène de narration locale à l'utilisateur pour une utilisation ultérieure.

Liens objet/proforme

L'application doit donc intégrer des mécanismes permettant de faire ce lien objet/proforme et de gérer l'ensemble des liens qui ont été indiqués par l'utilisateur. Dans la figure 5.22, les parties

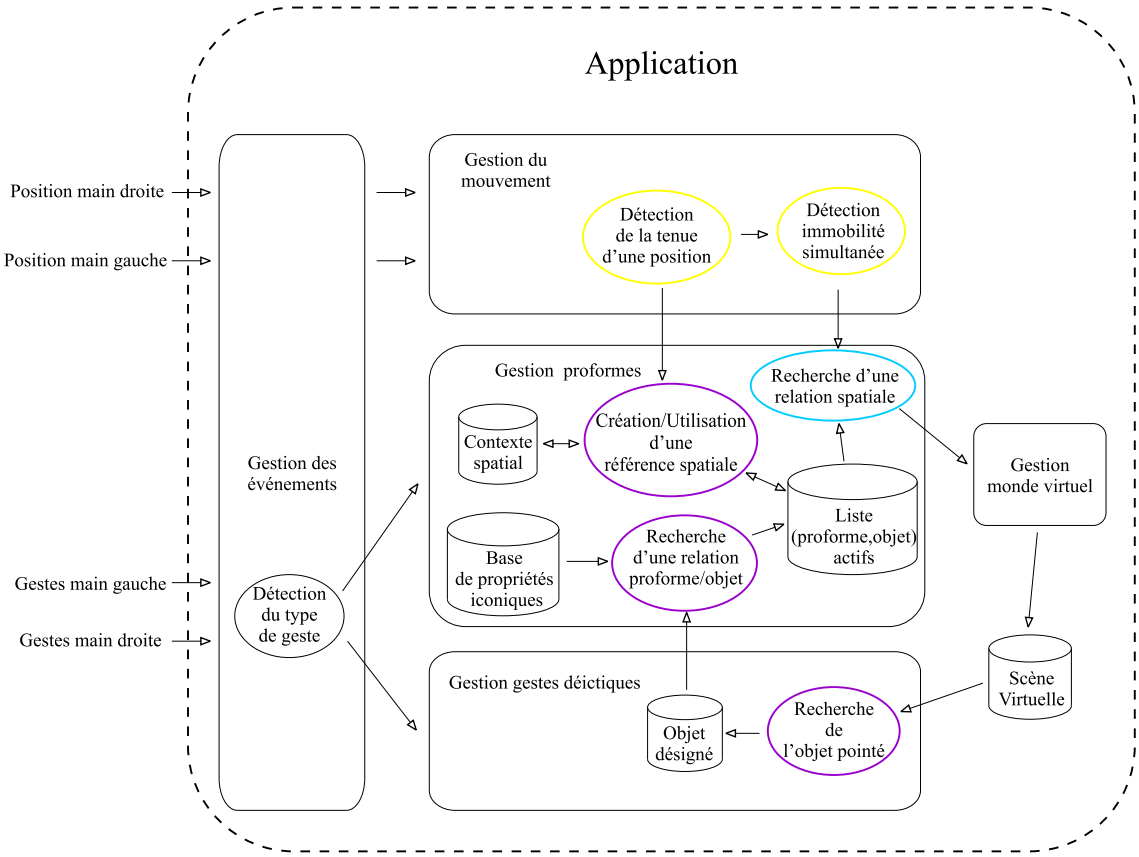

FIG. 5.22 – Descriptions des différentes parties qui interviennent dans le module de recherche de relations spatiales.

.

indiquées en violet correspondent à ces mécanismes.

Le module qui effectue la gestion de gestes déictiques permet de détecter l'objet que l'utilisateur est en train de désigner (donnée stockée dans *Objet désigné* jusqu'à la désignation suivante). Lorsque successivement ou simultanément à un geste de désignation, une proforme est effectuée, le module de gestion des proformes regarde si cette proforme est déjà liée à un objet et si ce n'est pas le cas, il examine si elle a un rapport iconique avec le dernier objet désigné. Pour cela, il consulte une base de connaissance qui indique les objets et les gestes reliés par des propriétés iconiques (*Base de propriétés iconiques* dans la figure 5.22). Si l'objet et la proforme sont en relation, alors le couple est ajouté dans la liste des couples (proforme, objet) actifs (note : cette liste contient au maximum deux couples, car elle correspond aux proformes actuellement produites par les mains).

Une fois un lien établi, l'utilisateur peut décider d'en faire une référence spatiale pour utiliser le couple plus tard. Pour cela, il doit positionner la proforme pendant un court instant au même endroit, puis passer à un autre geste (ces références spatiales sont stockées dans le *contexte spatial*). Pour utiliser un couple archivé dans le contexte, il suffit de refaire la même proforme et de le positionner à l'endroit de la référence spatiale pendant un court instant. Le couple est alors réinséré

dans la liste des couples (proforme, objet) actifs.

Détection des points de synchronisation

Une fois que l'on sait gérer les couples proforme/objet, il va falloir interpréter les éventuelles relations qui existent entre deux couples. Mais pour cela, il faut tout d'abord détecter les points de synchronisation entre les proformes. Cette détection est lancée bien évidemment uniquement lorsque la liste des proformes/objets contient deux couples. Dans la proposition d'architecture de la section 3.1, la détection des points de synchronisation se fait sur les moments de corrélations entre la main gauche et la main droite (gestes débutant au même moment, finissant ensemble, etc). Ici, du fait de la syntaxe gestuelle adoptée, on va se baser uniquement sur le mouvement et non pas sur toutes les composantes du geste.

En effet, les deux mains doivent tenir un bref instant la même position pour donner une relation spatiale. Il va donc suffire de détecter les pauses simultanées des mains (figure 5.23) pour lancer le processus d'interprétation de relations spatiales. Les parties indiquées en jaune dans la figure 5.22 effectuent cette détection de points de synchronisation.

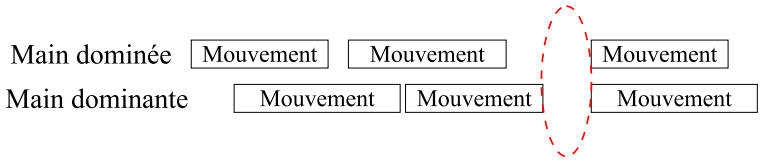

FIG. 5.23 – Recherche de pauses simultanées.

Interprétation des relations spatiales

Une fois qu'un point de synchronisation a été détecté, le système va devoir examiner l'éventuelle présence d'une relation spatiale. Pour cela, il va devoir vérifier la proximité spatiale des deux mains, et si la distance est suffisamment faible, il va calculer le type de relation spatiale qui est établie entre les proformes. Une fois que la relation spatiale a été établie, le module de gestion des proformes va pouvoir indiquer au module qui gère la scène virtuelle quels sont les objets impliqués et donner la relation qui existe entre ces deux objets. La relation spatiale est alors répercutée sur la scène.
C'est la partie indiquée en bleu sur la figure 5.22 qui effectue la détection et l'interprétation des relations spatiales.

5.5.3 Implémentation et expérimentation du système

Maintenant que nous avons vu le principe général du fonctionnement du module de détection de relations spatiales, nous allons voir comment les différentes parties ont été implémentées et quelles sont les réflexions que nous avons retirées de l'expérimentation de ces parties.

Détection des pauses

On a vu que l'utilisation de l'immobilité est utilisée comme déclencheur d'un certain nombre d'actions : ajout d'un couple au contexte spatial, utilisation d'un couple du contexte spatial et détection d'une relation spatiale. Pour détecter l'immobilité de la main, on calcule la vitesse de la main entre sa dernière position et la position courante. On compare ensuite cette vitesse par rapport à un seuil pour déterminer si la main est immobile ou non. L'utilisation d'un seuil est nécessaire car, d'une part il arrive qu'on ne tienne pas une position de manière parfaitement immobile et d'autre part la technique d'acquisition qui est basée sur des champs magnétiques induit une légère variation au cours du temps. L'estimation du seuil s'est faite de manière expérimentale à partir de plusieurs gestes où l'on a considéré la position comme immobile.
Nous nous sommes aperçus que, dans certains cas, on détecte une invariance de la position qui ne correspond pas à un déclenchement d'action. Par exemple, lorsque l'utilisateur "affine" sa désignation, la position de la main peut être quasi invariante. C'est pourquoi l'orientation et la configuration de la main ont été rajoutées ultérieurement pour détecter les moments où la main est totalement immobile, ce qui arrive uniquement lorsque l'on désire déclencher une action, ou que l'utilisateur n'effectue aucun geste. Les tests effectués pour détecter l'immobilité sont donc les suivants :

$$\frac{\sqrt{(x_{t_1}-x_{t_0})^2+(z_{t_1}-z_{t_0})^2+(z_{t_1}-z_{t_0})^2}}{t_1-t_0} < seuil\_mouvement$$

$$\frac{\sqrt{(rx_{t_1}-rx_{t_0})^2+(rz_{t_1}-rz_{t_0})^2+(rz_{t_1}-rz_{t_0})^2}}{t_1-t_0} < seuil\_rotation$$

$$\sum_{i=1}^{N} |a^i_{t_1} - a^i_{t_0}| < seuil\_configuration$$

Avec t_0 la date de la trame précédente, t_1 celui de la trame courante, a^i le ième capteur du gant numérique et N le nombre de capteurs sur le gant.

Représentation des propriétés iconiques

Pour cette expérimentation, il a été décidé de stocker directement les différents couples geste/objet possibles. La liste suivante peut être utilisée comme base de connaissance pour effectuer les liens iconiques :

```
PLAT   TABLE
PLAT   VOITURE
POING  BOULE
C      VERRE
C      VOITURE
```

L'utilisation de ce type de liste implique d'avoir un étiquetage au niveau des objets de la scène. Par exemple, lorsque l'on va désigner l'objet voiture dans la scène virtuelle, l'étiquette de cet objet va être *VOITURE*. D'autre part, il nécessite d'avoir une connaissance a priori des objets que chaque proforme peut représenter.
Dans la scène virtuelle qui nous a servie à effectuer les tests, seuls la table et les verres étaient

présents. La voiture et la boule sont dans la liste à titre d'exemple. On voit par exemple qu'un objet (la voiture) peut avoir plusieurs proformes suivant la relation spatiale que l'on désire exprimer. Si l'on veut mettre un objet *sur* la voiture, on va utiliser la proforme *plat* (qui désigne alors le toit de la voiture), tandis que le geste C va être utilisé pour une relation *dans* (C représente le fait que l'habitacle peut contenir un objet).

Dans notre expérimentation, nous avons utilisé uniquement des scènes contenant un nombre réduit d'objets. Le parcours de la liste qui donne les relations iconiques objet/geste se fait donc de manière rapide. Mais si jamais, pour des scènes plus complexes, on veut représenter pour chaque proforme un grand nombre d'objets, on voit qu'un problème de parcours de listes peut vite poser problème. Une première solution consiste à utiliser des tables de hachage ce qui améliore grandement le temps d'accès au bon couple proforme/objet. C'est cette solution qui a été adoptée dans l'optique de futures expérimentations avec des scènes plus complexes.
Une deuxième solution pourrait être de calculer les propriétés iconiques des gestes comme dans [Sowa et Wachsmuth, 2001]. Il suffit ensuite de comparer les caractéristiques topologiques de la proforme avec celles de l'objet sélectionné. Cette solution nécessite par contre d'enrichir tous les objets de la scène virtuelle avec leurs différentes propriétés topologiques ce qui peut être assez lourd à effectuer.

Gestion du contexte spatial

Le contexte spatial est géré sous la forme d'une liste de triplets <*position, proforme, objet*>. Pour insérer un nouveau triplet dans le contexte, il faut associer un objet et une proforme, positionner la proforme et garder la main immobile un court instant. C'est la position de la proforme qui est utilisée pour remplir le premier champ du triplet.
Ensuite, pour utiliser une référence spatiale, il faut positionner la même proforme (non associée à un objet) au même endroit. Comme on retrouve rarement strictement la même position, une erreur d'environ 15cm est acceptée (à peu près la largeur d'une grande main). Lorsqu'une proforme P non associée est donc immobile à une position (x, y, z), on va chercher dans la liste le triplet $<(x_r, y_r, z_r),\ P,\ O>$ pour lequel la distance $\|(x, y, z) - (x_r, y_r, z_r)\|$ est minimale et inférieure à 15cm.
Actuellement, on parcourt la liste du début à la fin pour trouver le meilleur triplet. Cela est dû au fait que la liste est non triée. Un utilisateur n'utilise pas a priori un contexte spatial contenant plus d'une dizaine d'objets du fait de la difficulté à mémoriser un grand nombre de positions spatiales. Nous considérons donc qu'il n'y aura pas de problèmes de complexité de taille pour le parcours de la liste.

Dans un premier temps, le contexte spatial n'a pas été représenté visuellement dans le monde virtuel. Durant les expérimentations, nous nous sommes vite aperçus que l'accumulation d'entités dans le contexte spatial et le moindre déplacement rendent très difficile la réactivation d'une entité en positionnant la proforme au bon endroit. C'est pourquoi, les différents entités sont actuellement représentées à l'aide de sphères dans le contexte spatial (voir figure 5.24).

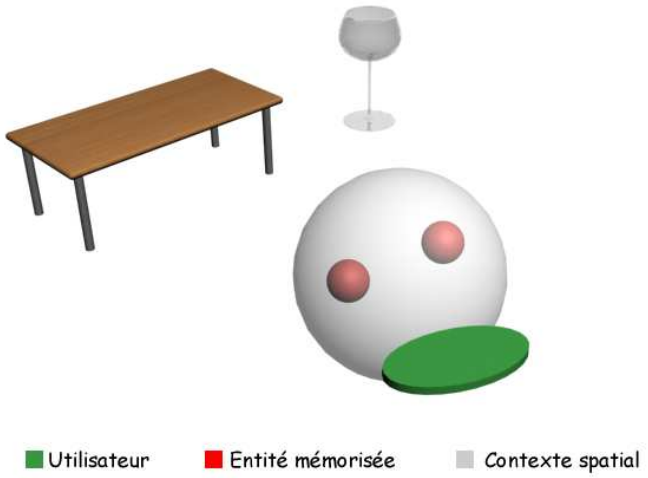

FIG. 5.24 – Illustration du contexte spatial avec deux entités mémorisées.

Recherche de relations spatiales

Lorsque l'on va avoir simultanément un couple proforme/objet et une immobilité pour chaque main, on va pouvoir déclencher la recherche d'une relation spatiale entre les deux mains.
L'interaction qui a lieu entre les mains correspond à donner une position à la main dominante par rapport à la position de la main dominée. Il va donc falloir dans un premier temps déterminer quelle est la main dominée. La main dominée jouant le rôle d'un locatif stable, elle ne bouge pas pendant que la main dominante se déplace par rapport à elle. Donc lorsque la main droite s'immobilise pour exprimer la relation spatiale, la position de la main dominée n'a pas bougé par rapport à la position du dernier triplet fourni par cette main au contexte spatial. Tandis que la main droite, elle, a quitté la position du dernier triplet ajouté dans le contexte spatial. Donc, en se basant sur l'état du contexte spatial, on arrive à déterminer quelles sont la main dominante et la main dominée.
Ensuite, on va vérifier si les mains sont suffisamment proches pour exprimer une relation spatiale. Cela se fait tout naturellement en comparant la distance entre les deux mains à un seuil. Ce seuil a été réglé arbitrairement à 25cm.
Enfin, on va examiner la position de la main dominante par rapport à celle de la main dominée pour voir quelle relation spatiale est exprimée. La position utilisée pour chaque main lors des expérimentations correspond au centre de la paume. On va constituer à partir de la position de la main dominée un repère qui va nous servir à tester la disposition de la main dominante (figure 5.25).

Chaque axe va être décomposé en trois intervalles : $]-\infty;-val[$,$[-val;val]$ et $]val;\infty[$. Par exemple, sur l'axe vertical (axe des z), cela va permettre de déterminer si l'on se trouve en dessous, au même niveau, ou au dessus. En combinant des comparaisons entre la position de la main dominante et les intervalles des trois axes, on va pouvoir déterminer les différentes relations spatiales. Par exemple, si la position se situe entre les valeurs val et $-val$ sur les axes x et y et dans l'intervalle $]val;\infty[$ sur l'axe z, alors la main dominante se situe *au dessus* de la main dominée.

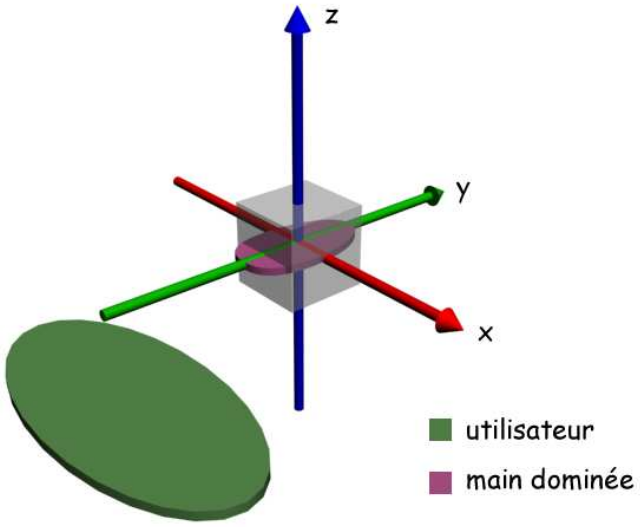

FIG. 5.25 – Repère utilisé pour examiner la disposition spatiale de la main dominante. Un côté de la boite englobante de la main est de taille $2val$.

Lors des tests effectués sur différents cas de relations spatiales, nous avons constaté des difficultés pour interpréter correctement les relations spatiales lorsque les mains sont très proches l'une de l'autre. Cela est dû au fait que l'on utilise la position correspondant au centre de la paume, et que *val* est choisi de manière arbitraire (5cm). Par exemple, on ne peut déterminer si la main dominante est au dessus lorsqu'elle est en contact physique avec l'autre main (on interprète une relation *dans*). L'utilisation de gants numériques DataGlove5 (très peu d'informations sur la forme de la main) pendant les tests a déterminé en partie ces choix limitants.
En effet, l'utilisation de gants CyberGlove, avec lesquels on connaît la position de chaque phalange, aurait permis d'effectuer le choix de valeurs différentes. Comme position de référence, on peut alors utiliser le barycentre de la main, ce qui permet d'avoir un "vrai" centre quel que soit la configuration et l'orientation de la main. Comme bornes pour les intervalles des trois axes, on peut choisir les limites de la boite englobante de la main dominée, ce qui permet une grande précision lorsque les deux mains sont très proches l'une de l'autre.

5.6 Bilan et Perspectives

Bilan

Nous avons exposé dans ce chapitre nos choix concernant l'implémentation de l'architecture proposée au chapitre 3. Le système ClientReco qui a été implémenté pour la partie reconnaissance peut sembler limitant du fait de la technique de reconnaissance choisie. Toutefois il nous semble être suffisant pour le vocabulaire gestuel généralement rencontré en réalité virtuelle ; d'autre part il répond à nos besoin de robustesse et de simplicité vis à vis des utilisateurs. L'implémentation des parties concernant la comparaison gestes monomanuels/geste bimanuel et la détection de relations

entre les mains dépend ensuite des besoins de l'application de réalité virtuelle que l'on développe.

Dans ce chapitre, nous avons également expérimenté l'ajout de composantes de la langue des signes en réalité virtuelle. Il s'agit des transferts situationnels qui ont été adaptés pour permettre une nouvelle manière d'intéragir avec des objets distants. La nouveauté réside principalement dans le fait que l'on manipule une entité relativement à une autre et non pas en absolu. D'autre part, l'interaction fait appel à la notion d'iconicité vis à vis des objets, ce qui a été peu utilisé en réalité virtuelle.
Actuellement, seule la "faisabilité" d'un système permettant d'utiliser ce type d'interaction a été testée. Il reste toute une expérimentation à mener afin de valider d'un point de vue ergonomique ce type d'interaction qui se base sur un mécanisme de la langue des signes.

Perspectives

Les quelques relations spatiales (dessus, dessous, à gauche, etc) qui ont été introduites dans notre protype ne permettent pas des interactions très évoluées. Un des points futurs à développer consiste à enrichir l'ensemble des relations afin d'exprimer des informations plus complexes. En particulier il serait intéressant d'ajouter des gestes permettant d'indiquer des contraintes entre deux objets ; par exemple en utilisant l'orientation et la position des proformes on peut indiquer qu'un objet est perpendiculaire, parallèle ou tangent à un autre.
Si l'introduction des transferts situationels en réalité virtuelle se révèle bénéfique, il peut être également intéressant d'utiliser alors les transferts de taille et/ou de forme. Ce mecanisme gestuel se révèle en effet particulièrement puissant pour décrire le contour d'un objet, ce qui est tout à fait pertinent pour la réalité virtuelle.

En dehors de l'utilisation de la langue des signes en réalité virtuelle, l'architecture pour les gestes bimanuels a d'autres perspectives d'utilisation. Par exemple, elle peut aider à la distinction entre un geste de saisie à une main et un geste de changement de taille qui consiste à "saisir" l'objet à deux endroits différents puis à "l'étirer" ou le "compacter" (le risque étant de confondre le geste de changement de taille avec deux saisies simultanées). Nous pensons également que ce type d'architecture peut être très intéressant pour toutes les interaction gestuelles basées sur une relation main dominante/main dominée où l'on peut effectuer simultanément une manipulation et une modification d'un objet.

Pour expérimenter cela, nous projetons d'utiliser l'architecture au sein de thèmes de recherche du projet VENISE, en particulier dans le domaine de la CAO en milieu immersif [Convard et Bourdot, 2003]. En effet, dans ce domaine, la création, la modification et la manipulation d'objets se fait actuellement par l'intermédiaire d'un wand. L'introduction du geste permettrait une interaction plus naturelle, voir même plus performante, avec les objets. Par exemple, la main dominée tient et oriente l'objet pendant que l'autre main modifie l'objet en déplaçant un de ses sommets.

Chapitre 6

Expérimentation dans le cadre de la langue des signes française

Après le chapitre 5, où nous avons détaillé l'implémentation de notre architecture dans le cadre de la réalité virtuelle, nous abordons ici le domaine de la langue des signes. Le développement du prototype, que nous décrivons dans ce chapitre, a alterné avec celui du précédent chapitre ; cela a eu pour conséquence la redondance de certaines parties ou l'utilisation de parties déjà développées dans l'autre domaine. C'est pourquoi certains points de ce chapitre vont paraître similaires (en particulier au niveau de l'acquisition des données), ou moins détaillés. Nous commençons, dans la section 1, par présenter le corpus qui nous sert pour les expérimentations dans le cadre de la langue des signes. Ensuite, dans la section suivante, nous allons discuter de l'acquisition des données et de la manière de les rendre le plus synchrone possible. La troisième section traite de la manière dont sont implémentés les modules de reconnaissances de l'architecture. Puis la section 4, détaille les recherches qui ont été faites dans le cadre du module de comparaison. La section suivante enchaîne sur la détection et l'interprétation des points de synchronisation. Enfin, avant de conclure, nous traitons dans la section 6 des éventuelles extensions possibles à un prototype pour la langue des signes, à savoir la réalisation d'un module d'analyse syntaxique et sémantique et l'aide à la reconnaissance.

6.1 Corpus d'expérimentation

Pour mener l'expérimentation d'un système de reconnaissance, il faut disposer d'un corpus. Ce corpus doit être établi de manière à pouvoir tester les caractéristiques du système. C'est à dire que l'on doit avoir dans le corpus des phrases représentatives de(s) problème(s) que doit résoudre le système. Et il faut que chaque problème soit abordé à travers différents cas afin d'être sûr que le système ne traite pas uniquement un cas particulier.
Cette section décrit le corpus utilisé pour évaluer notre architecture de reconnaissance. Nous donnons également les raisons qui nous ont amenés à faire certains choix au niveau de la syntaxe, du vocabulaire et des phrases.

6.1.1 Structure des phrases

Notre objectif n'étant pas d'élaborer un module d'analyse syntaxique et sémantique, nous avons formaté le corpus de manière à contrôler ces aspects. Ainsi, pour faire nos évaluations nous avons choisi des phrases figées pour concentrer l'évaluation uniquement sur les problèmes que nous voulons traiter, à savoir l'utilisation mixte de signes bimanuels et monomanuels, et l'expression de relations spatiales entre signes monomanuels.
Nous avons sélectionné des énoncés types exprimant une relation spatiale entre deux entités. Chacune de ces phrases est composée de quatre signes : deux signes standards monomanuels ou bimanuels, et deux proformes pour les représenter dans le reste de la phrase. La structure syntaxique utilisée afin d'exprimer une relation spatiale est la suivante :

- Le premier signe est un signe standard qui introduit une première entité. Ce signe peut être monomanuel ou bimanuel et doit énoncer une entité qui peut servir de référence spatiale stable.
- Le second signe est une proforme qui permet de positionner cette entité dans la scène de narration. Il est effectué avec la main dominée, car il va servir de référence lors de l'expression de la relation spatiale.
- Ensuite, on produit un deuxième signe standard qui donne la deuxième entité. Lui aussi peut être bimanuel ou monomanuel ; par contre, il ne correspond pas forcément à une entité qui sert de référence spatiale.
- Le dernier signe est une proforme qui représente la deuxième entité. Il est effectué avec la main dominante et est mis en relation spatiale avec la première proforme qui sert de référence. Nous qualifierons par la suite la proforme ou l'entité ***d'actant***.

L'expression de la relation spatiale ayant lieu en fin de phrase, la première proforme (qui donne la référence) peut être tenue ou non tout au long de la phrase suivant la nature du deuxième signe standard. Dans la figure 6.1, on peut voir ce qui peut se produire sur les trois canaux gestuels suivant que la proforme est maintenue ou pas (respectivement cas 2 et 1). Le signe noté *s1* correspond au premier signe standard, le signe *p1* représente la première proforme. De même *s2* et *p2* correspondent respectivement au second signe standard et à la deuxième proforme.

Cette structure de phrase permet de satisfaire nos besoins. D'une part elle permet de mêler un signe bimanuel avec des signes monomanuels simultanés (cas 2 dans la figure 6.1), d'autre part elle met en relation spatiale les deux proformes utilisées.

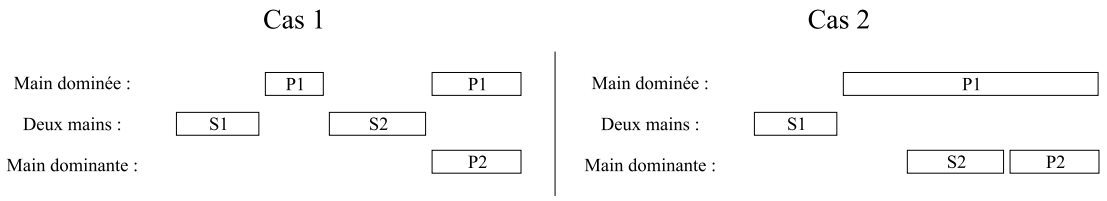

FIG. 6.1 – Deux exemples de structures de phrases du corpus d'expérimentation.

Cette syntaxe, très figée, nous permet également de nous affranchir du problème d'établissement d'un lien entre une proforme et le signe auquel elle correspond, problème qui sort du cadre de cette thèse. Dans notre cas, on sait tout de suite à quelle entité est relié une proforme, alors que dans une phrase "normale", il faudrait utiliser la scène de narration voire même les propriétés de la forme et des fonctionnalités de l'entité.

6.1.2 Vocabulaire

Les signes standards choisis sont [VERRE], [VOITURE], [TABLE] et [CHIEN] (voir figure 6.2). Ils permettent de couvrir différents cas : signe monomanuel, signe bimanuel, signe statique, signe dynamique, etc (le tableau 6.1 page 104 donne les propriétés de chaque signe). En plus d'avoir été choisis pour leurs caractéristiques gestuelles, ces signes ont également été sélectionnés en fonction des propriétés des entités qu'ils représentent. Par exemple, une voiture peut aussi bien servir de support que de conteneur, ce qui permet de faire intervenir deux différentes proformes pour une même entité.

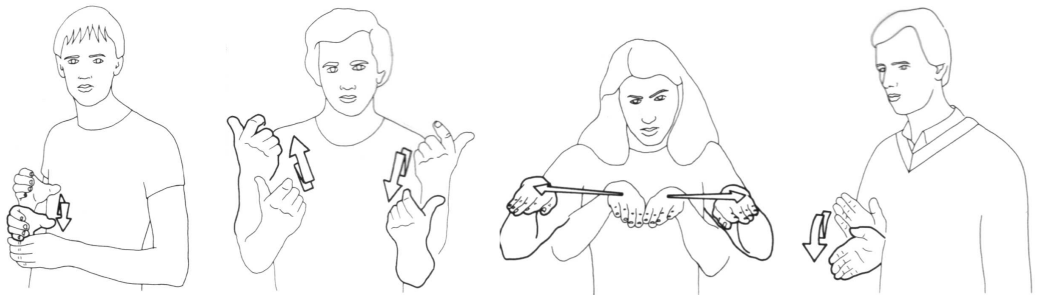

FIG. 6.2 – Les signes standards du corpus. De gauche à droite : [VERRE], [VOITURE], [TABLE] et [CHIEN]. (Dictionnaires de l'IVT).

Les proformes utilisées (figure 6.3) sont les suivantes :

- **main plate**. Cette proforme est utilisée pour représenter une entité plate ou servant de support. On va tout naturellement l'utiliser pour la *table* et pour la *voiture* lorsque cette dernière sert de support.
- **C**. Suivant son orientation, cette proforme peut avoir différents rôles. Ici, nous nous restreignons au cas où elle est utilisée pour des conteneurs et des cylindres horizontaux ou verticaux.

Cette proforme est utilisée pour représenter le *verre* et la *voiture* lorsque cette dernière sert de conteneur.

– **X**. Cette proforme est le plus souvent utilisée pour représenter les pattes d'un animal ou les jambes d'une personne assise. Dans ce corpus nous l'utilisons pour symboliser l'entité *chien*.

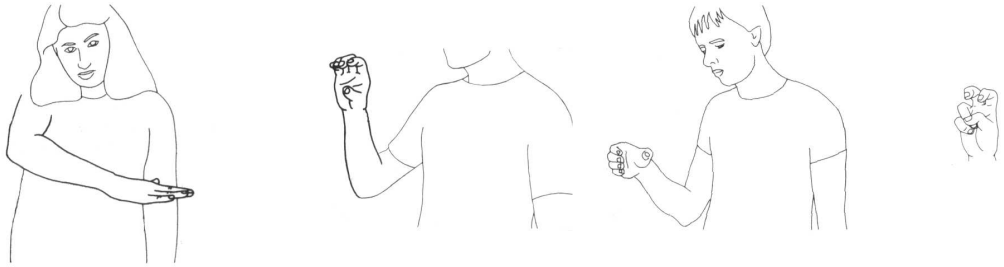

FIG. 6.3 – Les proformes du corpus. A gauche la proforme **main plate**, au centre **C** avec les deux orientations possibles, à droite la proforme **X**. (Dictionnaires de l'IVT).

On voit que le choix effectué pour les signes standards permet de tester deux situations : utilisation de plusieurs proformes pour une même entité (cas de l'entité *voiture*), ou utilisation d'une même proforme pour plusieurs entités (entités *voiture* et *verre*). Dans le cas de notre corpus, on pourrait distinguer les entités par la différence d'orientation de la proforme *C*, mais ça n'est pas nécessaire du fait de la structure syntaxique rigide que nous avons adoptée pour les phrases.

Le tableau 6.1 (page 104) nous donne les caractéristiques des différents signes du corpus. Dans l'ordre, on peut voir : le type de vocabulaire, le nombre de mains qui interviennent dans le signe, le canal gestuel qui est susceptible de produire le signe (2M : deux mains, MD : main dominante, md : main dominée) et ensuite les paramètres. Pour les proformes de notre corpus, seules la configuration et l'orientation sont fixées. Le mouvement peut varier suivant la phrase (en général, on observe un déplacement suivi d'une immobilisation pour définir une position), et l'emplacement est choisi en fonction de la relation spatiale que l'on souhaite exprimer.

On constate à travers ce tableau que ce vocabulaire de taille réduite permet tout de même d'avoir une certaine diversité au niveau des propriétés : présence de paramètres aussi bien statiques que dynamiques, vocabulaire bimanuel, monomanuel main dominée, monomanuel main dominante, etc. Cette diversité nécessitera d'élaborer un module de reconnaissance relativement polyvalent, nous parlerons de cet aspect plus loin.

6.1.3 Corpus

Nous avons vu en début de section que deux types de phrases sont utilisés dans notre corpus. Le premier type correspond à la succession de signes suivante : <standard 1>, <proforme 1>, <standard 2>, <proforme 2>+<proforme 1>. Dans le deuxième type de phrase, la *proforme 1* est maintenue : <standard 1>, <proforme 1>, <standard 2>+<proforme 1>, <proforme 2>+<proforme 1>. Le tableau 6.2 décrit une liste de sept phrases qui permettent, à partir du vocabulaire disponible, d'exprimer trois relations spatiales : *sur*, *sous* et *dans*. Le choix de ces trois relations spatiales permet de tester les trois possibilités de disposition spatiale sur un axe. Depuis ces trois cas, on peut se ramener très facilement à d'autres relations spatiales (devant , derrière, à gauche,

Nom	Type	Nb. de mains	Main	Configuration	Mouvement	Emplacement	Orientation
[TABLE]	standard	2	2M	plat	dynamique	neutre	statique
				plat	dynamique	neutre	statique
[VOITURE]	standard	2	2M	S	dynamique	neutre	statique
				S	dynamique	neutre	statique
[CHIEN]	standard	1	MD	plat	statique	ventre	dynamique
[VERRE]	standard	2	2M	C	dynamique (MD)	neutre	statique
				C	statique (md)	neutre	statique
X	proforme	1	MD	X	variable	choisi	statique
C	proforme	1	MD	C	variable	choisi	statique (verticale)
C	proforme	1	md	C	variable	choisi	statique (horizontale)
main plate	proforme	1	md	plat	variable	choisi	statique (horizontale)

md : main dominée, MD : main dominante, 2M : les deux mains interviennent.
variable : mouvement dépendant du contexte, neutre : zone située devant l'utilisateur, choisi : emplacement dépendant du contexte

TAB. 6.1 – Description des caractéristiques des différents signes du corpus.

	Phrases en français	Phrases en LSF				phrase
		standard 1	proforme 1	standard 2	proforme 2	
1	Le verre est dans la voiture	[VOITURE]	*C*	[VERRE]	*C*	type 1
2	Le chien est dans la voiture	[VOITURE]	*C*	[CHIEN]	*X*	type 2
3	Le chien est sous la voiture	[VOITURE]	*main plate*	[CHIEN]	*X*	type 2
4	Le chien est sur la voiture	[VOITURE]	*main plate*	[CHIEN]	*X*	type 2
5	Le chien est sur la table	[TABLE]	*main plate*	[CHIEN]	*X*	type 2
6	Le verre est sur la table	[TABLE]	*main plate*	[VERRE]	*C*	type 1
7	Le verre est sous la table	[TABLE]	*main plate*	[VERRE]	*C*	type 1

TAB. 6.2 – Les phrases du corpus d'expérimentation.

à droite) en utilisant les axes horizontaux au lieu de l'axe vertical.

Ces phrases ont été choisies de manière à ce qu'il n'y ait qu'une variation au niveau de chaque élément de la phrase : entité servant de référence, rôle de cette entité, entité servant d'actant et relation spatiale. En utilisant certaines phrases de ce corpus, on arrive à tester individuellement chaque élément. Ces ensembles de phrases sont les suivants :

- Les phrases 1 et 2 permettent de tester deux entités (actants) différentes pour une même relation spatiale et une même entité de référence.
- Les phrases 2, 3 et 4 donnent les trois différentes relations spatiales en utilisant les mêmes entités.
- Les phrases 2 et 3 testent l'utilisation de proformes différentes pour une même entité.
- Les phrases 4 et 5 correspondent à l'utilisation d'entités différentes pour la même proforme.
- Les couples de phrases (3,7) et (5,6) permettent la comparaison des deux types de phrases : celui où l'on maintient la proforme (type 2), et celui où on la relâche temporairement (type 1).

6.2 Acquisition des données

Pour établir le corpus et effectuer les évaluations, la première étape qui s'impose est l'acquisition de données (premier module dans l'architecture du système. Voir figure 6.4). Pour notre expérimentation nous avons utilisé comme périphériques d'acquisition des DataGlove 5™ [1] (5DT) pour la flexion des doigts et un système Flock of Bird™ (Ascension Tech.)[2] pour la position et l'orientation de chaque main. Ces périphériques effectuent leurs mesures à des fréquences différentes : 100Hz pour le système de capture de position/orientation et 200Hz pour les gants numériques.
Pour des raisons matérielles et logicielles (configuration du port série, vitesse de lecture du port série ...), les fréquences auxquelles les données arrivent au système sont différentes : environ 60Hz pour les positions/orientations et environ 190Hz pour les flexions des doigts.

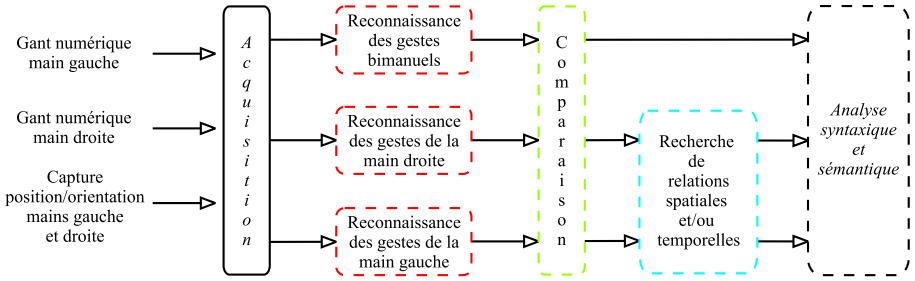

FIG. 6.4 – Place du module d'acquisition dans l'architecture.

Dans notre architecture (figure 6.4), nous voulons traiter des gestes bimanuels synchrones pour lesquels les mains débutent le même geste en même temps, comme dans les signes [TABLE] et [VOITURE]. Une synchronisation entre les données captées sur la main gauche et celles captées

[1] Gants numériques donnant une valeur de flexion globale pour chaque doigt.
[2] Système de capture de position/orientation à l'aide de capteurs électromagnétiques.

sur la main droite semble alors nécessaire. Nous devons ensuite comparer les signes reconnus par chacun des systèmes de reconnaissance (main gauche, main droite, deux mains). Même si les décalages possibles entre les différents canaux restent petits face à la durée des signes comparés, avoir une homogénéité temporelle entre les données fournies aux différents modules de reconnaissance nous parait utile pour assurer une comparaison de signes optimale. L'architecture effectue également l'interprétation de relations spatiales qui sont exprimées durant des points de synchronisation temporels. Pour détecter ces points de synchronisation, le système a besoin de "resynchroniser" les signes issus des canaux gestuels de la main gauche et de la main droite. On a alors besoin d'une datation unique pour les signes de la main gauche et ceux de la main droite.
Pour ces diverses raisons et devant l'hétérogénéité des fréquences des périphériques, nous avons décidé d'effectuer une synchronisation des données avant que celles-ci soient fournies de manière homogène aux systèmes de reconnaissance. Cela permet alors d'avoir une cohérence temporelle des données transitant dans chaque module de l'architecture.

6.2.1 Décalages entre les différents canaux gestuels

Du fait que les données des canaux de la main droite et de la main gauche sont dispersées sur plusieurs périphériques (un gant numérique et un système de capture de positions/orientations), une première tâche à réaliser est une fusion des données issues des différents périphériques. On effectue alors deux fusions afin de regrouper d'une part les données de la main droite et d'autre part les données de la main gauche.

Nous avons vu dans le chapitre précédent les réflexions qui ont été menées, au niveau du système ClientReco, sur la fusion de données asynchrones. Une synchronisation sur la fréquence la plus basse est utilisée afin de pouvoir effectuer la fusion de données arrivant de divers périphériques. Même si cette solution présente des inconvénients, elle permet une synchronisation simple et rapide des données.
Par contre, ClientReco, tel qu'il est utilisé au niveau de l'architecture en réalité virtuelle, ne permet pas forcément une bonne homogénéité entre les différents canaux (main droite, deux mains, main gauche). En effet, la synchronisation sur la fréquence la plus basse s'effectue au niveau de chaque module de reconnaissance. Rien n'empêche que l'on soit confronté à une situation où le périphérique le plus lent ne soit pas le même pour l'ensemble des modules. On aurait alors des fusions différentes et, comme conséquence, des données différentes pour chaque système de reconnaissance.

Dans le cadre de la langue des signes, les expérimentations sur la synchronisation de données ont été menées avant la création du système ClientReco. Une toute autre approche a été utilisée : plutôt qu'une synchronisation sur les données, une synchronisation sur les différents modules de reconnaissance a été réalisée.
Cela revient à se synchroniser sur le périphérique le plus lent si les systèmes de reconnaissance sont plus rapides que ce dernier. Mais la grande différence vient du fait que la synchronisation est alors effectuée par l'ensemble des modules et non pas au niveau de chacun. Cela permet d'assurer des données identiques pour les trois modules de reconnaissance.
Pour effectuer cette tâche de synchronisation, nous avons implémenté les différents modules sous forme de ***threads***.

Threads et sémaphores

Un ***thread*** ou ***processus léger*** est un programme qui fonctionne au sein d'un processus système. A l'intérieur d'un processus, on peut avoir plusieurs threads qui fonctionnent "simultanément", c'est ce que l'on appelle le ***multithreading***. La grande différence entre l'utilisation de plusieurs threads et l'utilisation de plusieurs processus systèmes réside dans le fait que les threads partagent le même espace mémoire (celui du processus auquel ils appartiennent). Ils peuvent donc tous écrire et lire dans une même variable.
Cette utilisation partagée de la mémoire, si elle présente de grands avantages, n'est pas sans poser des problèmes de conflit d'accès. C'est pourquoi, parallèlement aux threads, différents mécanismes existent pour permettre une synchronisation des threads sur les variables qu'ils partagent. Les ***sémaphores*** sont l'un de ces mécanismes.
Un ***sémaphore*** permet de gérer une liste de demandes d'accès à une ressource. Pour cela, il dispose d'un compteur **S**, de deux opérateurs **P** et **V** et d'une file d'attente. Les opérateurs effectuent les actions suivantes lorsqu'ils sont appelés par un thread sur un sémaphore :

```
P(S):
  si S>0
    alors S = S-1
    sinon
        endormir le thread
        placer le thread dans la file d'attente

V(S):
  S = S+1
  si la file d'attente n'est pas vide
    alors réveiller un thread de cette file
```

Ces opérateurs **P** et **V** vont servir respectivement à verrouiller ou libérer l'accès à une ressource. Suivant l'initialisation du compteur **S**, on va permettre l'accès à un ou plusieurs threads. Dès qu'une ressource est saturée en accès, les threads qui ont besoin de cette ressource sont mis en sommeil jusqu'à ce qu'elle soit de nouveau libre.
Certaines implémentations des sémaphores ajoutent un autre opérateur **P'** qui permet d'éviter à un thread d'être obligatoirement mis en sommeil et inséré dans la file d'attente. Le thread doit alors vérifier à intervalles réguliers la disponibilité de la ressource.

```
P'(S):
  si S>0
    alors
        S = S-1
        renvoyer vrai
    sinon renvoyer faux
```

Implémentation du système

Ces mécanismes de traitement en parallèle et de synchronisation correspondent tout à fait à nos besoins concernant l'architecture (fonctionnement en parallèle des modules de reconnaissance)

et la synchronisation des données (on veut synchroniser les modules pour l'accès aux données).

Notre système se situe dans un cas appelé ***producteur-consommateur*** en programmation multithreads : d'une part le module d'acquisition doit effectuer la capture de données (producteur), d'autre part les modules de reconnaissance doivent utiliser ces données (consommateurs). Cette situation soulève un problème de conflit d'accès sur les variables partagées entre le module d'acquisition et les modules de reconnaissance.
En addition à ce problème nous allons rajouter ceux propres à nos besoins : nous voulons que les modules de reconnaissance accèdent simultanément aux mêmes données, et nous tenons à ce que ces données soient les plus récentes possible.

La figure 6.5 décrit les ensembles de sémaphores utilisés (losanges), entre les différents threads (ovales) de l'architecture, afin de traiter les différents problèmes que nous venons de citer. Seules deux variables (*données main gauche* et *données main droite*) sont communes aux threads producteurs et consommateurs. L'annexe A donne en détail l'implémentation des threads et des sémaphores donnés dans cette figure.

Le module d'acquisition est chargé de la gestion des trois périphériques (DataGlove™ main gauche, DataGlove™ main droite et Flock of Bird™) qui émettent leurs données à des moments différents. De plus, ces périphériques émettent leurs données sous forme de flux continus. On va alors trouver au niveau du module d'acquisition trois threads qui se chargent chacun d'un périphérique et qui effectuent une acquisition en continu des données.
Au niveau de chaque module de reconnaissance de signes, nous avons choisi l'utilisation d'un unique thread qui effectue la synchronisation et la reconnaissance. On pourrait utiliser plusieurs threads (par exemple un thread pour la reconnaissance de chaque paramètre) mais cela n'apporte aucun avantage et nécessite l'ajout de sémaphores supplémentaires.

Les sémaphores, dans le cadre du modèle producteur-consommateur, sont uniquement utilisés afin d'empêcher la lecture de données simultanément à l'écriture de ces mêmes données. Dans notre cas, l'emploi des sémaphores va être beaucoup plus compliqué.
On peut identifier trois principaux groupes de sémaphores dans notre implémentation. Un premier groupe (en bleu dans la figure 6.5) qui est chargé de gérer le problème producteur-consommateur et qui synchronise les threads producteurs sur les threads consommateurs. Ce sont les modules de reconnaissance qui décident du moment où une trame de données va leur être fournie. On voit ici la différence par rapport au système ClientReco qui, lui, suit l'ordre d'arrivée des trames.
Le deuxième groupe de sémaphores (en vert) sert à effectuer la synchronisation entre les différents modules de reconnaissance afin qu'ils effectuent tous ensemble l'acquisition de la même trame de données. Le troisième groupe (en jaune dans la figure) permet au module d'acquisition d'indiquer aux modules de reconnaissance lorsque de nouvelles données sont disponibles pour chaque périphérique.

L'utilisation de ces trois ensembles de sémaphores au niveau des différents threads nous permet de choisir précisément l'ordre de l'exécution des différentes étapes des threads. La figure 6.6 décrit l'exécution d'un cycle complet de l'enchaînement des étapes des threads.

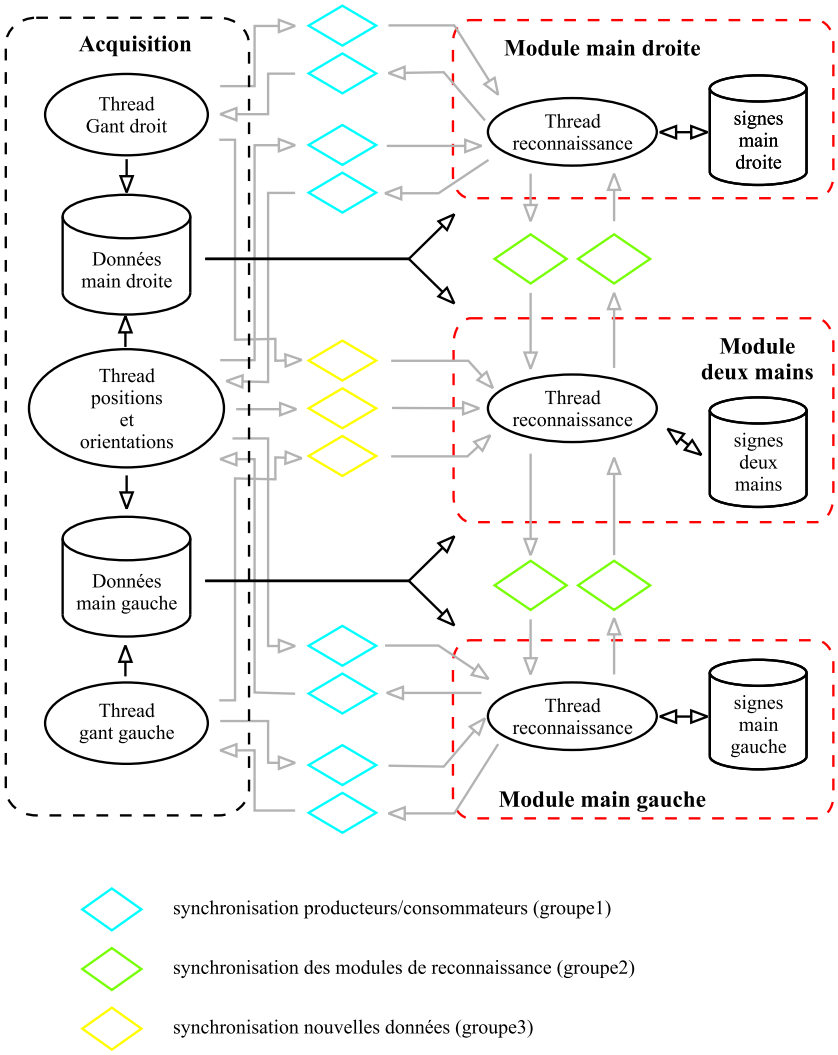

FIG. 6.5 – Description de l'organisation des threads et sémaphores utilisés.

Nous sommes assurés d'avoir le comportement suivant au niveau du système :

1. Tout d'abord les threads producteurs effectuent chacun l'acquisition d'une trame de données.
2. Ensuite, si les threads consommateurs sont prêts, les threads producteurs transfèrent les données dans des variables partagées.
3. Puis ils autorisent, via les sémaphores du groupe 1, les threads de la main droite et de la main gauche à utiliser ces variables.
4. Ces deux threads indiquent alors au troisième (deux mains) que les données sont prêtes (sémaphores du groupe 2).
5. Chacun des threads consommateurs effectue alors le traitement des données qu'il a lues. Lorsque les threads de la main droite et de la main gauche ont fini, ils se mettent en attente du troisième thread (sémaphores du groupe 2).
6. Ce dernier (thread deux mains) attend de son côté, via les sémaphores du groupe 3, que les threads producteurs aient chacun reçu de nouvelles données.

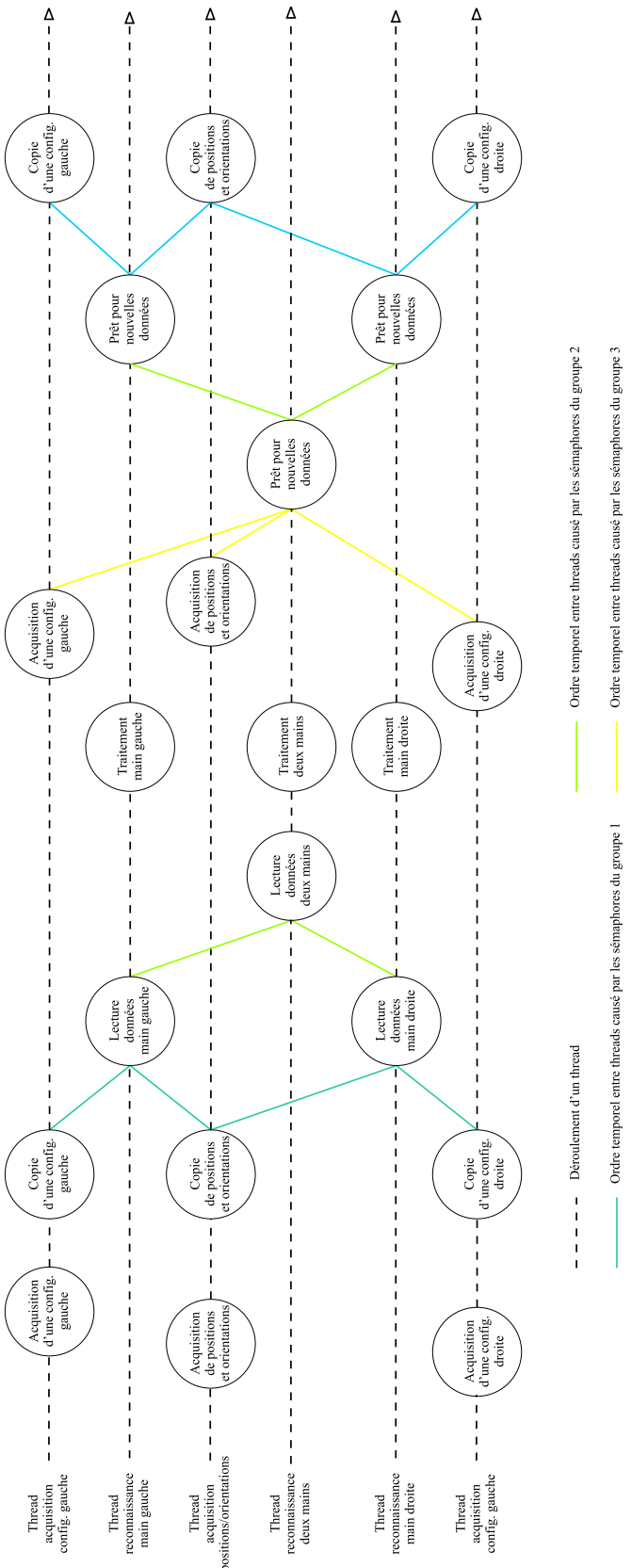

FIG. 6.6 – Exemple d'enchaînement de l'exécution des threads.

7. Lorsque c'est le cas, alors il indique aux deux autres threads consommateurs qu'il est prêt à traiter de nouvelles données.
8. Les threads main droite et main gauche informent alors les threads producteurs qu'ils sont prêts pour de nouvelles données.

Un nouveau cycle de traitement des données peut alors recommencer pour traiter les dernières données acquises. On remarque au niveau des threads producteurs que, entre le moment où sont signalées de nouvelles données et le moment où la copie des données est effectuée, plusieurs acquisitions de nouvelles données peuvent être effectuées. C'est ce mécanisme qui assure au système l'utilisation des données les plus récentes.

Cette implémentation multithread nous permet, grâce à l'utilisation conjointe de plusieurs sémaphores, d'obtenir ce que nous voulions, à savoir des données récentes et synchrones qui circulent au travers des modules de l'architecture.
Les expérimentations de cette architecture multithread ont été menées sur un ordinateur fonctionnant sous le système Linux. Nous avons pu constater une dégradation importante de la fréquence de traitement des trames par rapport à la fréquence du périphérique le plus lent même lorsque le temps de traitement au niveau des systèmes de reconnaissance est nul. Cela vient à priori du fait que le réveil d'un thread sur la plateforme Linux peut prendre un temps important (jusqu'à 10ms). Comme nous utilisons successivement plusieurs sémaphores pour contrôler l'ordre d'exécution des threads, un thread va subir plusieurs mises en veille. On peut ainsi avoir un retard allant de 20 à 30 ms sur l'ensemble du système, cela se traduit par une baisse de la fréquence du système qui fonctionne au minimum à environ 30Hz.
Cette fréquence reste toutefois acceptable pour traiter des signes, c'est pourquoi nous n'avons pas cherché à exploiter d'autres solutions (utilisation d'un système Linux temps réel, exécution des threads en tant qu'administrateur, intégration de tous les systèmes de reconnaissance au sein d'un unique thread, etc).

6.2.2 Différence de fréquences entre périphériques

Si les synchronisations effectuées au niveau du système multithread permettent une homogénéité temporelle entre les différents canaux gestuels, elles n'effectuent pas une synchronisation temporelle au sein de ces canaux. En effet, nous conservons le décalage entre les périphériques lents et les périphériques rapides qui sont utilisés pour un canal. Nous avons alors cherché dans un deuxième temps à avoir des données les plus synchrones possibles au sein d'un canal.

La solution consiste à utiliser les trames les plus proches temporellement pour effectuer la fusion des données. Les périphériques dont nous disposons fonctionnent à une fréquence de 190 ou 60Hz. Du fait que ces fréquences ne possèdent aucun diviseur commun, on ne peut pas effectuer une simple suppression de données pour se ramener à une unique fréquence.
On va devoir se baser sur la fréquence de chaque périphérique afin de choisir les trames qui nous intéressent. Dans notre cas, les intervalles entre deux trames sont de 5ms pour les DataGlove™ et de 17ms pour les Flock of Bird™. Le plus simple est de se synchroniser sur les événements du Flock of Bird™ pour rechercher les trames qui sont temporellement proches (en effet il faudra attendre moins longtemps pour recevoir la trame suivante des gants numériques).
Donc, lorsque l'on reçoit la position/orientation d'une main, on va comparer la date courante par

rapport à celle de la dernière configuration reçue. Si l'écart est supérieur à la moitié de la période des DataGlove™, on attend la configuration suivante sinon on envoie directement les données aux systèmes de reconnaissance concernés (voir figure 6.7 pour un exemple de synchronisation).

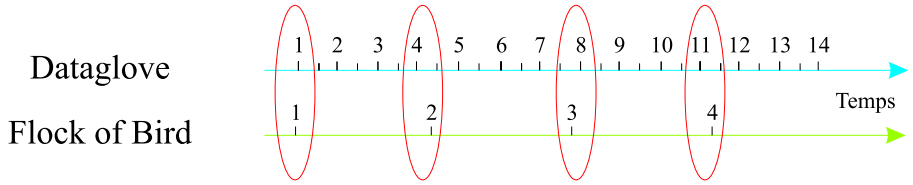

FIG. 6.7 – Exemple de synchronisation des périphériques.

Nous avons expérimenté cette synchronisation en modifiant l'architecture multithread exposée précédemment. Les threads producteurs effectuent alors une recherche des trames les plus proches temporellement avant de copier les valeurs dans les variables partagées.
Le principe fonctionne bien et permet d'avoir les données les plus synchrones possibles, mais un problème de fréquence a de nouveau surgi. L'utilisation de sémaphores additionnée au décalage induit par l'utilisation de trames suivantes conduit à un fonctionnement du système proche de 20Hz. D'autre part, il arrive que certaines trames de données du Flock Of Bird soient absentes (mesure impossible, ou problème au niveau du port série), et comme nous nous synchronisons sur ces trames, de grosses irrégularités peuvent apparaître au sein des données captées par les systèmes de reconnaissance.

Pour capter de manière fiable l'évolution dynamique de certains signes rapides, une fréquence minimale de traitement de données doit être assurée. Nous situons de manière empirique cette limite aux alentours de 15-20Hz, passer en dessous de cette borne ne permet pas de mesurer correctement certaines variations très brèves qui peuvent avoir lieu au niveau de la configuration ou de l'orientation.
C'est pourquoi, pour ces raisons de limite de fréquence, nous n'avons finalement pas retenu ce calcul de synchronisation qui permet d'avoir les données le plus proche temporellement. Toutefois, il pourrait être utilisable dans un autre système moins lent que celui que nous avons ici.
Malgré le fait que seule la synchronisation entre canaux gestuels (sémaphores) a été utilisée lors de nos tests, la déformation des données induite par la désynchronisation des périphériques n'a pas été trop gênante. Cela est probablement dû à la capacité que les systèmes de reconnaissance ont à gérer la variabilité au niveau des valeurs calculées depuis ces données asynchrones.

6.3 Modules de reconnaissance

Après avoir effectué l'acquisition des données, il faut les interpréter. Dans le cas de notre corpus, où se mêlent signes monomanuels et signes bimanuels, ce sont trois modules de reconnaissance qui sont chargés d'effectuer l'interprétation des données (voir l'architecture dans la figure 6.8).

Dans un premier temps, nous nous sommes penchés plutôt sur l'utilisation de modules basés sur des modèles de Markov cachés. Ce type de modèle permettant de modéliser facilement les phénomènes de coarticulation, il est tout à fait adapté pour la langue des signes où les gestes sont enchaînés.

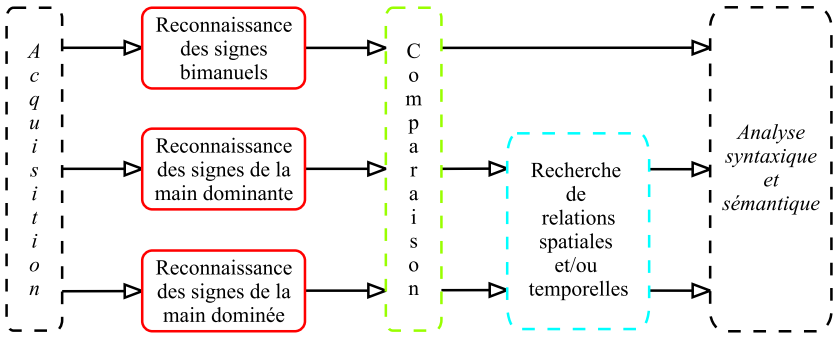

FIG. 6.8 – Place des modules de reconnaissance dans l'architecture.

D'autre part nous voulons reconnaître, indépendamment les uns des autres, les quatre paramètres (configuration, emplacement, mouvement, orientation) afin de pouvoir exploiter l'information contenue dans chacun. De plus, cette décomposition permet de traiter des vocabulaires de grande taille. C'est pourquoi nous avons exploré les pistes des modèles de Markov cachés multi-canaux et en particulier celle des modèles de Markov cachés parallèles. Ces derniers ont été développés par Christian Vogler [Vogler et Metaxas, 1999a] et permettent de traiter plusieurs canaux de données de manière indépendante tout en permettant une fusion des données issues de ces différents canaux.

Toutefois, pour la reconnaissance de l'orientation et de l'emplacement, les modèles de Markov ne nous paraissent pas la technique la plus appropriée. Ces paramètres, selon le type de signe, peuvent être assez libres et ne possèdent pas un vocabulaire bien délimité, contrairement à la configuration et au mouvement qui ont un nombre de conformations assez précis. Du fait de cette différence entre les paramètres constituant un signe, nous penchons au final pour l'utilisation de systèmes distincts pour ces paramètres. Nous allons maintenant voir comment nous avons choisi de traiter chacun des paramètres.

6.3.1 Décomposition en paramètres

Pour reconnaître chaque paramètre, différents systèmes de reconnaissance peuvent être utilisés. Les modèles de Markov cachés nous semblent plus adaptés pour les paramètres dynamiques ou présentant une forte coarticulation (par exemple la configuration). Le pattern matching (système statistique ou calculs géométriques) semble plus approprié pour les paramètres qui sont plutôt statiques et peuvent produire des valeurs non apprises par le système (par exemple l'emplacement qui peut être aussi bien prédéfini qu'impromptu). Les réseaux de neurones permettent de donner, lors d'une interprétation, l'évaluation de l'écart d'un paramètre entre plusieurs éléments de son vocabulaire (par exemple l'orientation peut être verticale, horizontale ou présenter une valeur intermédiaire).

Etant donné le vocabulaire utilisé dans le corpus d'expérimentation, l'utilisation de modèles de Markov cachés et du pattern matching nous semblent le plus approprié pour les différents paramètres. Toutefois, pour des raisons de durée de développement, en tenant compte de l'implémentation dans le même temps du système de reconnaissance de gestes pour la réalité virtuelle, et

au vu de la simplicité du vocabulaire du corpus, nous avons finalement principalement utilisé un système statistique[3] afin d'interpréter les différents paramètres.
Ce système statistique a été intégré au système multithread décrit dans la section précédente. Au niveau de chaque thread de reconnaissance, plusieurs instances du système sont créées pour les différents paramètres. La figure 6.9 décrit le contenu de l'un de ces threads.

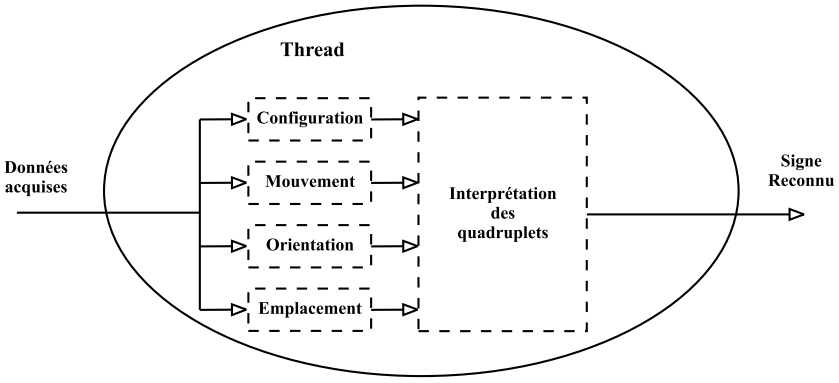

FIG. 6.9 – Exemple d'utilisation du système statistique dans le cas d'une implémentation multithread.

Configuration

Nous avons utilisé le système statistique pour ce paramètre. La simplicité du corpus d'expérimentation nous permet d'utiliser ce système sans trop de difficultés. Suivant le module de reconnaissance considéré (main dominante, main dominée, deux mains), le vocabulaire à reconnaître n'est pas le même. Pour le module de la main gauche (main dominée) les configurations suivantes peuvent être rencontrées : *plat* et *C*. Le vocabulaire des configurations de la main droite (main dominante) est lui constitué de *plat*, *X* et *C*. Pour le module de reconnaissance des signes bimanuels, les configurations *plat*, *C* et *S* sont utilisées.
Pour chacun des vocabulaires des différents modules, des configurations supplémentaires (correspondant à différents cas non couverts par le vocabulaire du corpus) sont rajoutées afin de pouvoir déterminer si une configuration reconnue est correcte ou non (voir 5.3.1).

Les évaluations ayant été effectuées avec des gants DataGlove 5™ de 5DT, le choix des primitives a été très simple : il s'agit de la valeur de flexion donnée pour chacun des doigts. Comme les configurations du corpus sont relativement simples (tous les doigts suivent une même conformation), cette description simpliste de la main n'a pas posé de problèmes et les différents modules de reconnaissance peuvent tous reconnaître leurs différentes configurations avec une grande fiabilité.
Il est évident qu'avec un nombre de signes plus importants pour lesquels les configurations seraient plus variées, l'utilisation de ce modèle de gant et de cet ensemble de primitives ne serait pas du tout appropriée.

[3]Il s'agit de celui de Dean Rubine [Rubine, 1991], que nous avons également utilisé pour la réalité virtuelle. Voir la section 5.2.1 pour plus de détail sur ce système

Mouvement

Pour ce paramètre aussi, nous avons choisi d'utiliser l'approche statistique. Le système de Dean Rubine ayant été à l'origine conçu pour effectuer la reconnaissance de tracés en deux dimensions, son utilisation pour le mouvement semble tout à fait justifiée.

Dans le cas de notre corpus, le mouvement n'est réellement utile que pour l'interprétation des signes bimanuels, et même pour ces signes, le vocabulaire se réduit à un simple mouvement linéaire. Pour les autres signes, le mouvement est soit nul soit non signifiant (cas des proformes). L'implémentation d'un module reconnaissant les mouvements peut paraître a priori "optionnel" dans le cadre de nos expérimentations.
Nous avons néanmoins réalisé différents essais à partir du système de reconnaissance statistique dans l'optique d'un vocabulaire plus complexe ou d'une utilisation en réalité virtuelle.

Deux solutions peuvent être utilisées dans le cadre d'une analyse du mouvement.
- Soit on effectue une analyse globale qui consiste à considérer le mouvement entre deux immobilisations de la main ; on peut alors reconnaître des trajectoires complexes comme par exemple le tracé de lettres.

- Soit on examine le mouvement à un niveau local, auquel cas on s'intéresse plutôt aux différents éléments qui composent une trajectoire. Par exemple, on va être capable de savoir qu'un mouvement complet est composé de deux trajectoires rectilignes et d'un arc.

Du fait de la simplicité des mouvements utilisés dans le corpus, de l'éventuel usage du travail effectué dans le domaine de la réalité virtuelle, et de notre approche iconique de la langue des signes, nous avons choisi la solution basée sur les mouvements locaux. En effet, une des perspectives de ce travail est de pouvoir faire le lien entre une entité et sa forme, ou du moins certains traits saillants, exprimés à l'aide de gestes. Le système de reconnaissance va alors travailler sur de petits historiques de données afin de reconnaître des primitives de mouvement.

Le premier élément que doit pouvoir détecter un système de reconnaissance de mouvements est la présence ou l'absence de mouvement. Pour cela, nous utilisons tout simplement, comme primitive, la dérivée de la position au cours du temps : $\frac{dx}{dt}$, cette dérivée est calculée sur les deux dernières positions reçues. Cette primitive permet également de distinguer différentes vitesses pour une même trajectoire, mais cet aspect ne nous intéresse pas dans le cadre de notre corpus. Cette information serait par contre très utile pour pouvoir interpréter des données de type aspectuelles dans les verbes de la langue des signes (action menée lentement , de manière saccadée, interrompue, etc). Lorsque le système reconnaît qu'il y a absence de mouvement, le symbole *aucun* est renvoyé.

Dans un deuxième temps, le système doit être capable de distinguer les mouvements linéaires des mouvements courbes. Pour cela, il suffit d'évaluer la courbure d'une trajectoire. Cette évaluation se fait généralement à partir de plusieurs points de la courbe, la figure 6.10 donne un exemple de trois points pouvant servir à différents calculs.

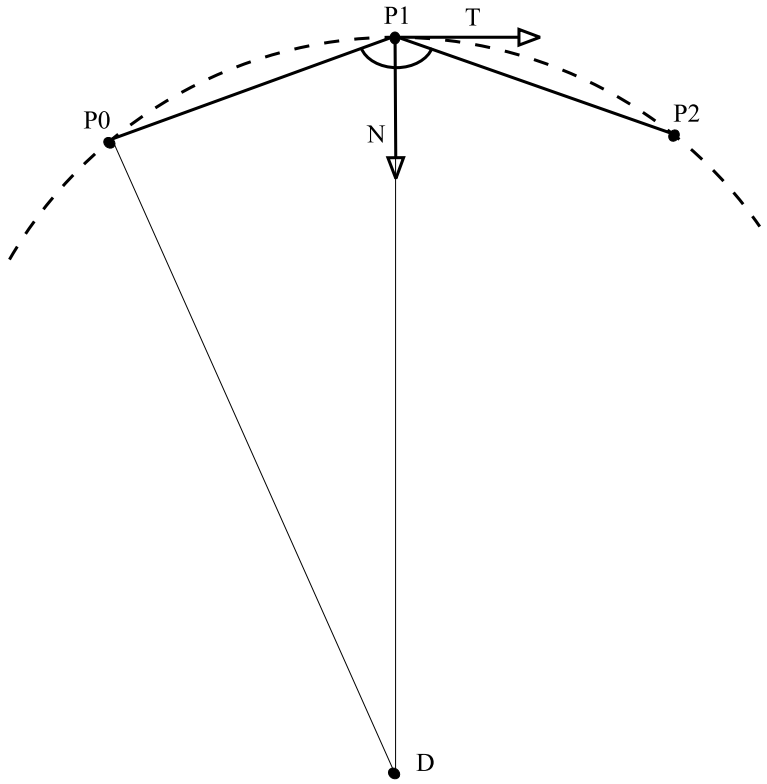

FIG. 6.10 – Trois points appartenant à une trajectoire courbe.

Evaluation de la courbure Une première solution consiste à calculer directement la courbure à partir des formules de Frenet. Il faut alors se ramener à un repère curviligne (on peut prendre P_0 comme origine) et calculer un repère de Frenet au point où l'on veut calculer la courbure (P_1 dans la figure 6.10). Pour calculer le repère, il faut calculer la tangente $\overrightarrow{T}$ (formule (1)) puis en déduire la normale $\overrightarrow{N}$. Ensuite, on peut déduire la courbure C depuis ces deux vecteurs $\overrightarrow{T}$ et $\overrightarrow{N}$ et de la formule (2) (on peut utiliser P_2 pour calculer la variation de $\overrightarrow{T}$ au cours de la courbe).

$$\overrightarrow{T} = \frac{d\left(\overrightarrow{P_0P_1}\right)}{ds} \quad (1)$$

$$\frac{d\overrightarrow{T}}{ds} = C\overrightarrow{T} \quad (2)$$

L'inconvénient de ce type de calcul est l'utilisation d'un repère curviligne. On est obligé de faire des approximations pour passer du repère cartésien où sont mesurés les points à ce repère curviligne où est effectué le calcul de la courbure.

Une autre solution consiste à calculer le rayon de courbure à partir de propriétés géométriques en considérant les trois points comme faisant parti d'un cercle et que $P_0P_1 \simeq P_1P_2$. On peut à partir du segment $[P_0P_2]$ déduire la médiatrice (P_1D), D étant le point diamétralement opposé à P_1. P_0P_1D étant un triangle inscrit dans le cercle on a $\widehat{P_1P_0}D$ qui est un angle droit, et comme on suppose que $P_0P_1 = P_1P_2$, on a également $\widehat{P_0P_1}D = \frac{\widehat{P_0P_1P_2}}{2}$. On peut alors déduire le rayon de

courbure avec la formule suivante :

$$R = \frac{1}{2} \frac{P_0P_1}{cos\left(\widehat{P_0P_1D}\right)} \quad (3)$$

L'inconvénient de cette méthode est qu'elle ne peut être utilisée qu'avec un mouvement relativement uniforme où les arcs P_0P_1 et P_1P_2 sont similaires.

Une troisième méthode consiste tout simplement à utiliser l'angle $\widehat{P_0P_1P_2}$ comme indicateur de courbure. Pour calculer cet angle on peut utiliser la formule suivante :

$$\widehat{P_0P_1P_2} = acos\left(\frac{\overrightarrow{V1}.\overrightarrow{V2}}{\|\overrightarrow{V1}\|\|\overrightarrow{V2}\|}\right) \quad (4)$$

Le défaut de cette solution est qu'elle ne permet pas d'avoir précisément la courbure d'une trajectoire, elle permet seulement d'avoir une "idée" de l'intensité de la courbure.

Pour des raisons de simplicité d'implémentation, nous avons testé uniquement les deux dernières solutions. Seule la dernière a été retenue car la méthode calculant le rayon de courbure ne marche pas très bien lors de mouvements où ont lieu des accélérations ou décélérations.

Problème de taille d'historique L'utilisation de petits historiques de données pour faire une analyse locale du mouvement n'est pas sans poser des problèmes pour l'évaluation de la courbure de la trajectoire. En effet, étant donné la fréquence d'acquisition des positions des mains (environ 60Hz), si on essaye d'évaluer la courbure à partir de trois mesures successives, les trois points P_0, P_1 et P_2 peuvent paraître comme faisant partie d'un mouvement linéaire (les tangentes sont alors quasi colinéaires et l'angle $\widehat{P_0P_1P_2}$ voisin de 180 degrés.

La solution la plus évidente consiste à prendre un historique de taille plus grande afin de choisir des points plus écartés. Dans certains cas, on ne peut pas se permettre de prendre un historique trop grand (problème de temps réel en réalité virtuelle par exemple), une deuxième solution consiste alors à effectuer un cumul des dernières évaluations calculées. Dans le cas d'une droite, le cumul n'augmente pas de manière significative, alors qu'avec une courbe l'addition des angles devient significative au bout d'un moment. Tout le problème, alors, est de décider du nombre de cumuls qu'il faut effectuer.

Si ces deux solutions permettent de détecter tout type de courbure, elles ne permettent pas en revanche de les différencier si on utilise un historique ou un cumul de taille fixe. En effet, suivant la vitesse du mouvement, les points ne vont pas être espacés de la même manière. Par conséquent, deux mouvements décrivant la même courbe à des vitesses différentes vont avoir deux évaluations différentes pour la courbure. La solution consiste alors à choisir dynamiquement son historique ou son cumul afin que ceux ci soient indépendants de la vitesse.

Au final, pour effectuer la différentiation entre droites et courbes, nous utilisons un historique correspondant à un mouvement d'une dizaine de centimètres (indépendant de la vitesse). Et sur cet historique, nous utilisons l'angle $\widehat{P_0P_1P_2}$ comme évaluation de la courbure. Le point P_0 est pris au début de l'historique, le point P_1 au milieu et le point P_2 à la fin.

Lorsque la main est immobile ou que l'on est en début de mouvement (historique de taille insuffisante), on considère que l'angle est nul et que l'on est en présence d'un mouvement rectiligne. De même en fin de mouvement, et que l'historique devient trop grand du fait de la faible vitesse, on renvoie un angle nul.
La primitive ainsi calculée permet de différencier les courbes des droites dans la plupart des cas. Lorsque la vitesse est trop basse ou la courbure trop faible, le système a tendance par contre à produire une réponse erronée. Des améliorations restent donc à apporter pour le calcul de cette primitive.

Orientation du mouvement Un troisième aspect qui peut être intégré à l'ensemble des caractéristiques du mouvement est l'orientation. Par exemple, dans le signe [TABLE], la main droite effectue un mouvement horizontal vers la droite, tandis que le mouvement du signe [VOITURE] est lui vertical.
Comme, dans le cas de notre corpus, la configuration de la main suffit largement à différencier les deux signes, nous n'avons pas testé cet aspect du mouvement. Toutefois, nous avons choisi le calcul suivant dans le cas où l'on devrait utiliser l'orientation du mouvement comme discriminant : à partir du premier et du dernier point de l'historique, on constitue un vecteur que l'on normalise puis que l'on projette sur chacun des axes du repère du signeur (axe z : hauteur, axe y : avant-arrière, axe x : gauche-droite). On obtient alors trois nombres qui permettent de discriminer les différentes situations. Par exemple, pour le mouvement de la main droite lors du signe [TABLE], la projection sur l'axe z va être voisine de 0 ce qui permet de désigner un mouvement horizontal ; le signe positif de la projection sur l'axe x va désigner, lui, le fait qu'on aille de gauche à droite.

Emplacement

Le cas du paramètre de l'emplacement est particulier. Il s'agit de déterminer si la valeur de l'emplacement fait partie d'une zone prédéfinie. C'est pourquoi nous avons choisi d'utiliser des calculs géométriques afin d'identifier l'appartenance de l'emplacement de la main à une de ces zones. La figure 6.11 montre différents emplacements au niveau du corps : ventre, torse, main, bras, front, etc.

D'autres emplacements autour de l'utilisateur sont également disponibles, en particulier la zone dite ***neutre*** où sont énoncés certains signes standard lorsque ceux ci ne sont pas positionnés de manière explicite dans la scène de narration.
Dans le contexte de notre corpus, seul l'emplacement du ventre et l'emplacement neutre sont utilisés. Pour déterminer ces zones, un troisième capteur de position/orientation est utilisé. Ce capteur est positionné au niveau de la ceinture et permet d'identifier la zone du *ventre*, celle du *torse* et la zone *neutre* qui est située juste devant le buste. Comme nous disposons de capteurs de position au niveau des mains, nous pouvons également identifier la zone de la *main droite* et celle de la *main gauche*.

Nous avons choisi des parallélépipèdes pour représenter les diverses zones. Ce choix a principalement été effectué sur des critères de simplicité d'implémentation car chaque zone ne possède pas vraiment de forme régulière. Chaque parallélépipède est calculé à la volée en fonction de la position et de l'orientation du capteur et de caractéristiques physiologiques prédéfinies (dimensions du torse par exemple).
Pour déterminer l'emplacement de la main, on regarde si la position de celle-ci se trouve dans une

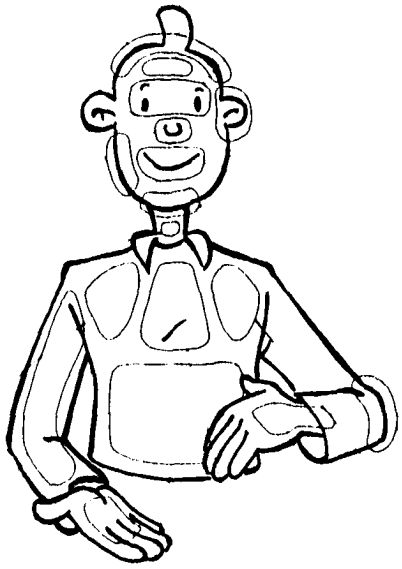

FIG. 6.11 – Différents emplacements prédéfinis au niveau du corps. (CISA - 2004)

des zones ou à proximité de celle-ci (environ la largeur d'un main, c'est à dire moins de 5cm). Lorsque l'emplacement ne peut être déterminé, il est jugé comme étant un emplacement *autre*. Le calcul de la proximité correspond à la distance minimale entre la main et chacune des faces du parallélépipède. Cette "distance de proximité" permet d'avoir une certaine variabilité de l'emplacement et rend possible le choix entre deux zones lorsque qu'une main se situe entre elles.

Orientation

L'orientation n'a pas du tout été utilisée dans nos tests du fait des caractéristiques du corpus qui permettent de distinguer les différents signes en utilisant les autres paramètres. Elle aurait pu être interprétée afin de distinguer l'orientation de certaines proformes ou pour ajouter certains critères de recherche de relations spatiales. Par exemple, une orientation horizontale de la proforme *main plate* indique que l'on va probablement avoir affaire à la relation spatiale selon l'axe vertical (*sur*, *sous*, *au-dessus*...).

Les primitives utilisées dans le cas d'un module de reconnaissance de l'orientation seraient similaires à celles utilisées pour l'orientation du mouvement. L'orientation étant définie par deux vecteurs (un orthogonal à la paume que nous appellerons V_o, l'autre colinéaire qui sera noté V_c), on peut normaliser ces vecteurs puis les projeter sur les axes du repère du signeur. On obtient alors six valeurs qui nous permettent d'identifier les différents cas : P_x^o, P_y^o et P_z^o pour les projections du vecteur orthogonal, P_x^c, P_y^c et P_z^c pour les projections du vecteur colinéaire. Les cas typiques pour chaque vecteur sont les suivants : si $P_z^c = 0$ la main est horizontale, si $P_x^c = P_y^c = 0$ la main est verticale, si $P_x^o = P_y^o = 0$ et $P_z^o = 1$ la paume est en haut, si $P_x^o = P_y^o = 0$ et $P_z^o = -1$ la paume est en bas, si $P_x^o = -1$ $P_y^o = 0$ et $P_z^o = 0$ la paume est vers la gauche, etc. On arrive ainsi à identifier les cas les plus typiques : signe *horizontal*, signe *vertical*, signe *horizontal paume en haut*, signe

horizontal paume en bas, etc. Si l'orientation ne peut être située dans aucun de ces cas typiques, elle est considérée comme *autre*.
Comme pour l'emplacement, on peut définir un certain seuil de tolérance autour des valeurs clefs (par exemple $-0,1 < P_z^c < 0,1$ au lieu de 0 quand la main est horizontale) afin d'avoir une certaine variabilité des valeurs clefs.

Symbole reconnu, score et valeurs caractéristiques

En plus du symbole reconnu, chaque système de reconnaissance va devoir fournir un certain nombre de valeurs aux modules suivants du système. Tout d'abord, chaque système va être obligé de fournir un score afin de pouvoir évaluer la qualité du signe reconnu lors de l'étape de comparaison. Le calcul du score étant effectué en fonction du processus de comparaison, nous reviendrons sur cet aspect un peu plus loin dans la section dédiée au module de comparaison.

Ensuite, nous voulons fournir un certain nombre de valeurs caractéristiques du symbole reconnu. Ces éléments peuvent être utilisés au niveau de l'interprétation du signe mais aussi plus loin lors de la phase de détection de relations spatiales et/ou temporelles, ou lors de la phase d'analyse sémantique. En effet, les paramètres peuvent être porteurs de sens à un niveau iconique (transferts situationnels, transferts de taille et de forme, etc) ; connaître leurs caractéristiques peut s'avérer être important au moment de la phase d'analyse sémantique.
Nous avons décidé d'utiliser les flexions des doigts pour la configuration, cela permet d'avoir une idée relativement précise de la forme de la main. Pour le mouvement, nous avons choisi la position de départ, la position d'arrivée et l'estimation de la courbure. On peut ainsi savoir à partir de plusieurs mouvements si l'on effectue un cercle, un carré, etc. Pour l'emplacement, nous gardons bien évidemment la position, nous avons également choisi de conserver les deux vecteurs de l'orientation comme caractéristiques. On peut ainsi, lors de signes monomanuels simultanés, comparer les vecteurs d'une main avec ceux de l'autre afin de tester si les mains sont parallèles, perpendiculaires, etc.

En résumé, chacun des modules décrits précédemment va envoyer à la suite du système : le symbole reconnu, un temps de début et un temps de fin, un score, et des valeurs correspondant aux caractéristiques du paramètre reconnu. Nous allons maintenant voir comment, à partir de ces données, le module de reconnaissance va pouvoir reconnaître un signe.

6.3.2 Reconnaissance du signe

Après que chaque paramètre ait été reconnu, un module de reconnaissance va devoir étiqueter ces paramètres avec un signe lorsque cela est possible. Cette tâche est effectuée par un module dédié qui est situé après les systèmes de reconnaissance (voir figure 6.12).

La tâche d'étiquetage peut être faite suivant diverses méthodes plus ou moins évoluées. Pour l'expérimentation de cette architecture, nous avons utilisé une méthode assez rudimentaire qui permet de traiter de manière ad hoc le vocabulaire de notre corpus.
Le module d'étiquetage va effectuer sa tâche en deux étapes : tout d'abord il va constituer un quadruplet [4] correspondant aux différents paramètres du signe à étiqueter ; ensuite il va comparer ce quadruplet à une base de signes constituée de quadruplets étiquetés. Si une correspondance est

[4] Dans nos expérimentations, nous avons utilisé uniquement trois paramètres, toutefois comme l'implémentation avec quatre paramètres est tout à fait similaire, c'est cette dernière que nous explicitons.

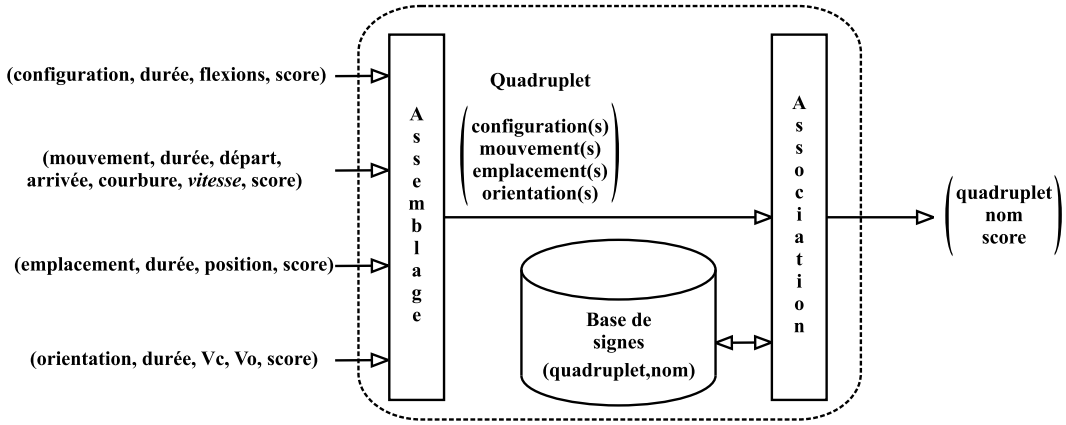

FIG. 6.12 – Partie étiquetage du module de reconnaissance

trouvée dans la base de signes, le module renvoie un quadruplet avec une étiquette, sinon il renvoie le quadruplet tel quel.
Nous allons maintenant voir en détail le fonctionnement de ce module d'étiquetage.

Constitution d'un quadruplet

Un quadruplet va correspondre à la totalité d'un signe, ce qui implique que chaque élément du quadruplet peut comprendre une suite de valeurs lorsque le paramètre correspondant est dynamique. La structure adoptée pour un quadruplet est alors un tableau de listes. Dans le cas d'un signe monomanuel, on manipule quatre listes, chacune correspondant à un paramètre. Dans le cas d'un signe bimanuel, il y a quatre couples de listes, chaque couple correspondant à un paramètre pour la main gauche et la main droite. Chaque élément d'une liste est un symbole associé à un temps de début et à un temps de fin. Ce symbole est également doté d'un score (lorsque le système de reconnaissance en donne un) et est associé à certaines valeurs : flexions des doigts pour une configuration, courbure et position de début et de fin pour un mouvement, position(s) pour un emplacement, V_o et V_c pour une orientation.
Dans la suite nous noterons un quadruplet monomanuel de la manière suivante :

```
<[liste config.],
 [liste mouv.],
 [list empl.],
 [liste orient.]>
```

et un quadruplet bimanuel de la manière suivante :

```
<([liste config. droite],[liste config. gauche]),
 ([liste mouv. droite],[liste mouv. gauche]),
 ([list empl. droite],[list empl. gauche]),
 ([liste orient. droite],[liste orient. gauche])>
```

Par exemple pour le signe [VOITURE] qui est un signe bimanuel où le mouvement est composé de plusieurs mouvements rectilignes, on aura le quadruplet suivant :

```
<([S],[S]),
 ([ligne,aucun,ligne,aucun,ligne],[ligne,aucun,ligne,aucun,ligne]),
 ([neutre],[neutre]),
 ([horiz. paume bas],[horiz. paume bas])>.
```

Pour constituer les quadruplets, le module ne peut pas utiliser directement les valeurs envoyées par les systèmes de reconnaissance. En effet, du fait que les systèmes de reconnaissance travaillent sur des historiques de taille limitée (plusieurs trames pour le mouvement, une ou plusieurs trames pour la configuration, une trame pour l'emplacement et l'orientation), un paramètre peut être reconnu en plusieurs fois. Par exemple, si le signeur maintient la configuration C pendant une douzaine de trames, mais que la taille de l'historique n'est que de trois trames au niveau du système de reconnaissance, on va recevoir quatre fois la même configuration.
Avant d'effectuer la constitution du quadruplet, il va falloir fusionner les répétitions au niveau de chaque paramètre. A partir de toutes les occurrences ayant le même symbole, on va alors constituer un unique élément pour lequel le temps de début et de fin, le score et les valeurs sont calculés en fonction de ceux des répétitions.

Ensuite, il va falloir effectuer la ***segmentation*** du quadruplet, c'est à dire que l'on va devoir décider du moment de début et de fin du signe correspondant au quadruplet. Du fait que certains paramètres sont dynamiques, la tâche de segmentation n'est pas forcément un processus aisé. Toutefois, le vocabulaire de notre corpus ne comprend que des configurations statiques, ce qui permet de se baser sur la variation de la configuration pour effectuer la segmentation des signes.
Lorsqu'un autre paramètre est à cheval sur deux configurations successives, nous avons décidé de séparer ce paramètre en deux éléments. Lorsqu'il est légèrement décalé nous avons choisi de le garder tel quel. La figure 6.13 donne une illustration des différentes segmentations possibles pour un quadruplet.

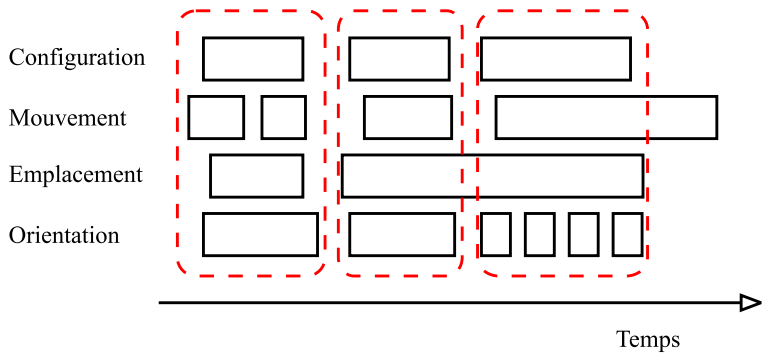

FIG. 6.13 – Exemples de segmentation des quadruplets.

Comparaison avec une base de signes

Une fois un quadruplet constitué, le module va devoir l'étiqueter s'il correspond à un signe connu. Pour cela, on compare le quadruplet avec ceux contenus dans une base de signes. Pour le module de reconnaissance de la main gauche, on va avoir la base de données constituée des couples quadruplet-nom suivants :

```
( <[C],[libre],[libre],[verticale]>                    , C          )
( <[plat],[libre],[libre],[horizontale paume en bas]> , main plate )
```

La valeur *libre* pour un paramètre indique que n'importe quelle valeur du paramètre peut être utilisée. Si par exemple le quadruplet <[C],[aucun],[autre],[verticale]> est identifié, on sait alors qu'il s'agit de la proforme *C*.
Dans notre cas, on utilise uniquement les symboles contenus dans les quadruplets, mais pour un vocabulaire contenant des signes ayant des paramètres similaires, on peut très bien imaginer l'utilisation de la durée et des caractéristiques des paramètres afin de discriminer plus précisément les signes.

Une fois que le module d'étiquetage a fini sa tâche, il renvoie un triplet constitué du quadruplet, du nom du signe reconnu (on renvoie *inconnu* si aucune correspondance n'a été trouvée) et d'un score. Le score correspond à l'évaluation de l'étiquetage effectué. Pour cette expérimentation, nous avons choisi de calculer ce score en fonction du nombre de paramètres mis en correspondance.
Par exemple, si on ne fait aucune correspondance et que l'on renvoie le signe *inconnu*, le score est 0. Si on arrive à faire une correspondance avec deux paramètres (cas de la proforme *C* ci dessus), on renvoie 2, si tous les paramètres sont en correspondance on renvoie 4.

6.4 Module de comparaison

Une fois les signes reconnus au niveau des différents modules de reconnaissance (main gauche, main droite, deux mains), c'est le module de comparaison (figure 6.14) qui prend le relais. A partir des triplets émis sur chaque canal, il va devoir déterminer si l'on a affaire à un signe bimanuel, un signe monomanuel, ou deux signes monomanuels simultanés.

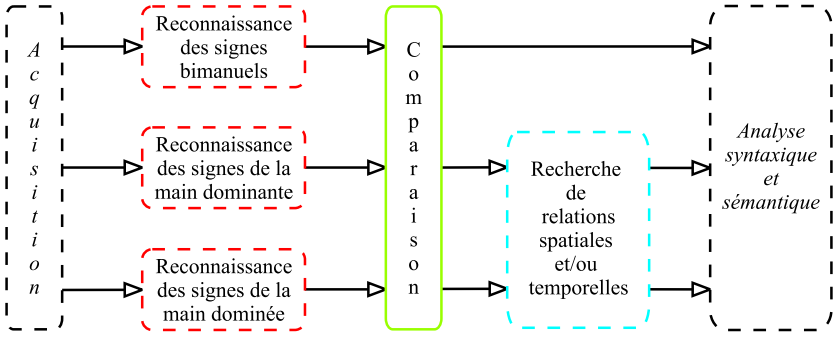

FIG. 6.14 – Place du module de comparaison dans l'architecture.

Pour pouvoir effectuer la comparaison des signes, le module va devoir se baser principalement sur une évaluation qualitative des reconnaissances effectuées. Pour cela, il va utiliser les scores contenus dans les triplets afin de déterminer le ou les signes qui sont corrects. On voit donc que les modules de comparaison et de reconnaissance sont complètement liés sur ce problème d'évaluation des signes.

Le premier problème à résoudre est la détermination des différents scores au niveau de chaque signe. Nous avons vu précédemment la méthode qui a été choisie pour effectuer l'évaluation de l'étiquetage. Nous allons maintenant voir les choix et expérimentations qui ont été effectués pour les paramètres.

6.4.1 Choix et évaluation de divers scores pour les paramètres

Au sein des modules de reconnaissance, deux types de systèmes existent : un système basé sur des calculs statistiques, et un système basé sur des calculs géométriques.

Paramètres évalués de manière géométrique

Pour l'emplacement et l'orientation, ce sont des calculs géométriques qui servent à déterminer le symbole reconnu. On utilise l'appartenance d'une position à une zone référence ou l'alignement de V_o et V_c par rapport à des axes. Pour permettre une certaine flexibilité, on utilise un certain intervalle autour de la zone ou des axes références. Au sein de ces intervalles, on peut calculer une distance qui est en mesure de servir d'évaluation de la qualité de la reconnaissance.

Pour l'emplacement, nous utilisons la distance entre la position courante et le parallélépipède qui représente la zone reconnue. Ainsi, si la main est à quatre centimètres au dessus de la face supérieure de la zone, le score va être de 4.

Pour l'orientation, la situation est un peu plus compliquée. D'une part, selon les cas, un ou deux vecteurs sont utilisés. Par exemple, pour l'orientation *verticale*, seul V_c est utilisé ; pour l'orientation *horizontale paume en haut*, les deux vecteurs sont utilisés. D'autre part, on ne fait pas toujours la même comparaison suivant les orientations, on peut tester si un des vecteurs est colinéaire, orthogonal, opposé, etc.
Nous avons choisi d'utiliser comme distance l'écart angulaire entre le vecteur courant et la position souhaitée pour ce vecteur. Par exemple, si l'on teste la colinéarité entre V_c et l'axe vertical (orientation verticale), on va calculer l'angle entre V_c et le vecteur de l'axe vertical. Dans le cas où les deux vecteurs V_c et V_o interviennent, le score correspond à l'addition des deux angles calculés, dans le cas où un seul vecteur intervient (V_c a priori), le score correspond à deux fois l'angle calculé.
Du fait que nous n'avons pas utilisé l'orientation lors de nos tests, ces scores n'ont pas pu être évalués. Les calculs de scores donnés ici sont purement indicatifs de ce que nous souhaiterions avoir comme type de score.

Paramètres reconnus de manière statistique

Le système de reconnaissance statistique utilisé revient à comparer un vecteur fourni en entrée (primitives calculées pour l'historique courant) avec des vecteurs représentant chaque élément du vocabulaire. Effectuer une évaluation de l'élément reconnu revient alors à calculer l'écart entre le vecteur donné et le vecteur représentant le paramètre reconnu.
Différents types de distances existent pour mesurer cet écart entre vecteurs. Pour déterminer la distance la plus appropriée à notre système, nous avons choisi d'effectuer un certain nombre de

tests avec différentes distances sur le paramètre de la configuration.

Parmi toutes les distances existantes, nous avons choisi de tester les suivantes : Mahalanobis, Hamming (également appelée distance de Manhattan ou bloc), Euclidienne (distance géométrique), Chebychev et Canberra.
Pour tester ces distances, nous avons effectué deux expérimentations. Dans la première, nous effectuons l'apprentissage de la configuration C pour la main gauche et la main droite et l'apprentissage globale des deux configurations C au niveau des deux mains. Cela permet d'une part de tester la variation de la distance entre une posture correcte (C) et une posture incorrecte (n'importe quelle configuration), et d'autre part d'évaluer la différence entre la distance d'un signe monomanuel et celle d'un signe bimanuel.
Dans la deuxième expérimentation, on utilise au niveau de chaque canal un vocabulaire constitué de plusieurs configurations (C, *plat*, S ...) afin d'évaluer la variation des distances sur des signes corrects mais employant des configurations différentes.

La distance de Mahalanobis effectue son calcul en fonction du vecteur fourni en entrée (V), du vecteur moyen représentant le symbole reconnu (P^c) et de la matrice de covariance du système (S).

$$d = (V - P^c)^T.S^{-1}.(V - P^c) \quad Mahalanobis$$

Du fait de l'utilisation de la matrice de covariance, la distance calculée dépend grandement de l'apprentissage effectué. Du coup, nous avons eu quelques problèmes avec cette distance, en particulier lors de la première expérimentation, probablement du fait de l'apprentissage réduit effectué pour les configurations (de 4 à 10 exemples suivant les tests). Mis à part ces problèmes, cette distance est la plus efficace pour distinguer une configuration correcte d'une configuration incorrecte.

Les autres distances sont moins dépendantes de l'apprentissage du fait qu'elles calculent leur évaluation uniquement à partir du vecteur moyen. Les calculs effectués sont similaires du fait que chacun utilise la distance au niveau de chaque primitive ($V_i - P_i^c, i \in [1..N]$ avec N le nombre de primitives) pour évaluer la distance totale entre les vecteurs.

$$d = \sqrt{\sum_{i=1}^{N} (V_i - P_i^c)^2} \quad Euclidienne$$

$$d = \sum_{i=1}^{N} |V_i - P_i^c| \quad Hamming$$

$$d = \max_{i \in [1..N]} |V_i - P_i^c| \quad Chebychev$$

$$d = \sum_{i=1}^{N} \left| \frac{V_i - P_i^c}{V_i + P_i^c} \right| \quad Canberra$$

La distance de Canberra a été très vite écartée car elle ne permet pas de discriminer correctement les bonnes configurations des mauvaises. Cette distance est plutôt adaptée pour des vecteurs dont les éléments sont répartis sur des échelles de valeurs très différentes. Or, pour la reconnaissance des configurations, les données proviennent de capteurs de flexion qui ont tous la même échelle de

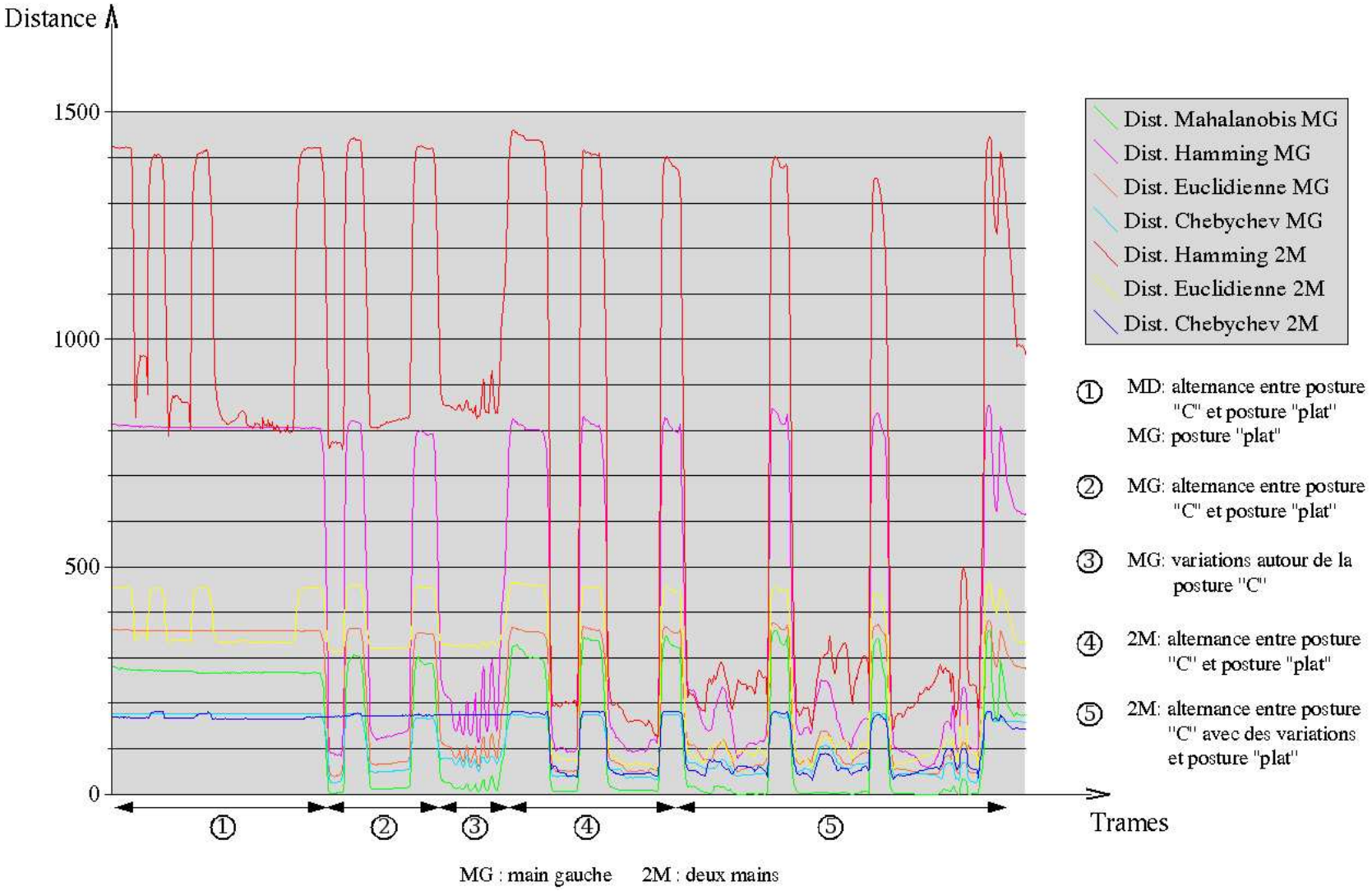

FIG. 6.15 – Les distances, dans le cas de la première expérimentation, pour les systèmes de reconnaissance de la main gauche et des deux mains. La distance de Mahalanobis est absente pour les deux mains car les valeurs sont incohérentes. La distance de Canberra n'est pas représentée pour des raisons d'échelles. Quand aux distances de la main droite, elles sont similaires à celles de la main gauche.

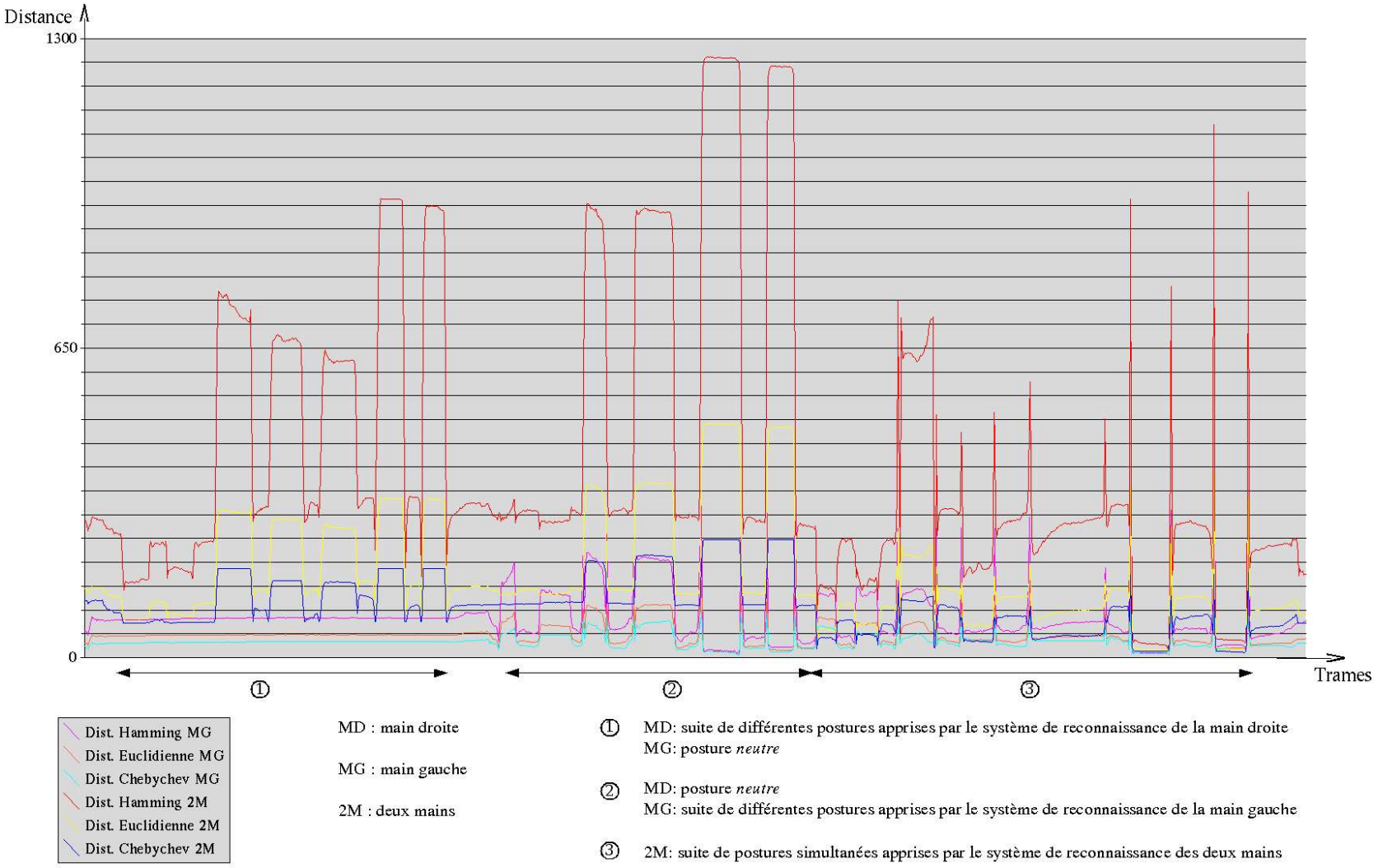

FIG. 6.16 – Les distances, dans le cas de la deuxième expérimentation, pour les systèmes de reconnaissance de la main gauche et des deux mains. Les distances de la figure 6.15 ont été reprises pour permettre la comparaison.

valeur.
Les distances d'Euclide et de Hamming ont un comportement similaire dans les deux tests (voir figures 6.15 6.16), toutefois la distance de Hamming effectue une meilleur discrimination entre les cas corrects et les cas incorrects. D'autre part, la distance de Hamming est plus robuste vis à vis des grands écarts, contrairement à la distance géométrique. Par rapport à la distance de Hamming, la distance de Chebychev a tendance à moins varier lorsque les valeurs du vecteur V oscillent autour de celles du vecteur P^c (voir figure 6.15).
Au niveau du module de la main droite et du module de la main gauche, lorsque l'on produit une configuration à deux mains (par exemple deux C), la distance de Chebychev est la seule à faire, parfois, une différence par rapport à une configuration monomanuelle (C dans le cas de notre exemple). Par exemple dans le cas de la première expérience (figure 6.15), la distance de Chebychev, contrairement aux autres, varie très peu durant les périodes 1, 2 et 3. Toutefois, cette différence n'est pas toujours exploitable pour effectuer une discrimination signes monomanuels/signe bimanuel au niveau du paramètre (la distance de Chebychev se comporte de manière similaire aux autres dans la figure 6.16).

Parmi toutes les distances testées, seules celles de Hamming et de Chebychev nous semblent intéressantes. Nous avons choisi de retenir celle de Hamming car elle permet une bonne discrimination des paramètres corrects de ceux qui ne le sont pas. D'autre part, elle nous semble assez proche des calculs effectués dans le cas des systèmes de reconnaissance de l'emplacement et de l'orientation.

Passage d'une distance à un score d'évaluation

Pour tous les paramètres, ce sont des "distances" qui sont utilisées : distance géométrique pour l'emplacement, différence d'angles pour l'orientation et distance de Hamming pour la configuration et le mouvement. Pour une distance, plus la valeur est petite, plus la reconnaissance est fiable. Par contre pour l'étiquetage, c'est un score croissant qui est utilisé pour évaluer la qualité du processus. Nous avons fait le choix de transformer les distances en scores croissants afin d'avoir une homogénéité entre les paramètres et l'étiquetage. Cette transformation se fait tout simplement de la manière suivante :

```
score = distance_max - distance
```

Du fait que les vecteurs sont normalisés, le calcul de la distance maximale correspond tout simplement au nombre de primitives pour la configuration et le mouvement. Pour l'emplacement et l'orientation, la distance maximale est la taille de la zone d'amplitude laissée autour de la position de référence.

6.4.2 Comparaison entre signes

Normalisation des scores

Une fois qu'une méthode d'évaluation a été choisie pour chaque paramètre, il va falloir comparer les différents signes à partir des différents scores calculés pour chacun des signes. Pour effectuer cette comparaison, il faut que les scores soient normalisés entre les trois modules de reconnaissance.

Dans notre situation, les scores sont presque normalisés du fait des choix effectués.

En effet, nous avons choisi les mêmes primitives au niveau des différents paramètres reconnus par les modules de reconnaissance, et nous avons opté pour les mêmes calculs de scores pour ces paramètres. La seule différence entre un module de reconnaissance de signes monomanuels et le module de reconnaissance de signes bimanuels est le doublement du nombre de paramètres.
Pour un signe monomanuel, on va avoir un quadruplet constitué de cinq scores (lorsque un paramètre varie au cours du signe, on fait la moyenne des scores) : celui de la configuration varie entre 0 et 5 (5 primitives normalisées), celui du mouvement varie entre 0 et 2 (2 primitives normalisées), celui de l'emplacement entre 0 et 5, celui de l'orientation entre 0 et 5 et le score de l'étiquetage varie entre 0 et 4. Pour un signe bimanuel, on va avoir 9 scores (deux fois plus de paramètres), avec la même variation pour le score des paramètres et une variation entre 0 et 8 pour le score de l'étiquetage.
Il suffit alors d'introduire un coefficient 2 lors de la comparaison entre signes monomanuels et signe bimanuel pour avoir une complète équivalence des scores entre monomanuel et bimanuel.

Par contre, pour évaluer la validité d'un signe indépendamment des autres, nous avons besoin de normaliser les scores au niveau des paramètres afin que chacun de ceux-ci ait le même poids. En conséquence, on verra apparaître dans la fonction de comparaison un coefficient 2,5 au niveau du mouvement. Le score de l'étiquetage est, lui, un peu particulier et n'est pas normalisé.

Fonction de comparaison

Nous présentons ici une fonction de comparaison minimale. Elle calcule un score global à partir des paramètres et de l'étiquetage puis compare ce score par rapport à un seuil et aux autres scores afin de décider quel est le(s) meilleur(s) signe(s). Les seuils sont pour l'instant déterminés de façon expérimentale, mais pourraient à plus long terme être calculés en fonction de diverses caractéristiques (amplitude des primitives, caractéristiques du système de reconnaissance, poids de l'étiquetage, etc).

```
score_main_gauche = (score_config + score_mouvt*2,5 + score_empl +
                     score_orient)/4 + score_etiq
score_main_droite = (score_config + score_mouvt*2,5 + score_empl +
                     score_orient)/4 + score_etiq
score_deux_mains = (score_config_g + score_mouvt_g*2,5 + score_empl_g +
                    score_orient_g)/4 + (score_config_d + score_mouvt_d*2,5 +
                    score_empl_d + score_orient_d)/4 + score_etiq

si (score_main_gauche > score_deux_mains et
    score_main_gauche > seuil_main_gauche)
 alors

   si (score_main_droite > score_deux_mains et
       score_main_droite > seuil_main_droite)
     alors cas deux signes monomanuels simultanés
```

```
    sinon cas signe main gauche seul

sinon

  si (score_main_droite > score_deux_mains et
      score_main_droite > seuil_main_droite)
    alors cas signe main droite seul

  sinon

    si (score_deux_mains > seuil_deux_mains)
      alors cas signe bimanuel

    sinon absence de signe ou signe non interprétable
```

Le calcul de score global au niveau des trois canaux gestuels permet de favoriser d'une part les signes étiquetés et d'autre part d'avantager les signes standards parmi les signes étiquetés. Cela permet d'éviter la reconnaissance de signes faisant partie de la grande iconicité au lieu de signes standards. En effet, du fait de la grande flexibilité des paramètres en grande iconicité, on peut très bien substituer une proforme (entre autres) à un signe standard lors de l'étape d'étiquetage. Par exemple, si on prend le signe [TABLE] dans notre corpus, on voit que l'on pourrait reconnaître deux proformes *main plate* à la place au niveau de la main droite et de la main gauche.

La fonction minimale qui a été présentée ici peut ensuite être enrichie avec d'autres critères (comme par exemple la direction du regard) pour effectuer la comparaison. Nous nous penchons en particulier sur des critères faisant intervenir la syntaxe de la phrase et la scène de narration. Cela nécessite une architecture ascendante-descendante qui intègre un module d'analyse syntaxique et sémantique. Notre architecture ne possédant pas actuellement ces caractéristiques, nous n'avons pu expérimenter ce type de comparaison qui nous semble beaucoup plus pertinent qu'une simple comparaison à base de scores.

6.4.3 Problèmes d'hétérogénéité

Hétérogénéité des valeurs

Dans nos expérimentations, peu de problèmes se posent pour l'estimation des scores et la comparaison de ces scores du fait de la relative homogénéité des données au niveau des différents modules de reconnaissance (même primitives, données continues, etc). Cependant, on peut très bien imaginer que l'on soit un jour confronté à une situation où l'on soit obligé d'avoir trois systèmes de reconnaissance ayant des caractéristiques totalement différentes.
Des études existent dans le domaine de la reconnaissance automatique afin d'effectuer des calculs de distances sur des données hétérogènes. Par exemple, dans [Wilson et Martinez, 1997] différentes techniques permettant de normaliser des données hétérogènes afin de calculer une distance sont présentées. Les algorithmes proposés permettent en particulier de mélanger à la fois des données continues et des données discrètes. Cela pourrait être intéressant pour des paramètres comme

l'emplacement ou l'orientation qui peuvent correspondre à des valeurs clefs (ensemble discret de valeurs) ou des valeurs non remarquables (ensemble "continu" de valeurs).

Hétérogénéité des systèmes de reconnaissance

Les méthodes statistiques et géométriques présentées dans la section précédente pour reconnaître les paramètres sont plus ou moins similaires. Elles reviennent à effectuer un appariement : on compare un vecteur de valeurs courant avec d'autres vecteurs de valeurs qui représentent les classes. La comparaison de systèmes utilisant ces méthodes n'est donc pas trop difficile.
Le choix de ces systèmes a été effectué en fonction du vocabulaire constituant le corpus, il est évident que la généralisation de ce corpus à d'autres signes et structures de phrases va poser des problèmes pour ces systèmes de reconnaissance. Il est plus que probable que d'autres systèmes comme les modèles de Markov cachés ou les réseaux de neurones seront alors employés.
Or la manière d'évaluer un élément reconnu n'est pas du tout la même pour ces systèmes. Par exemple un modèle de Markov caché va calculer la probabilité de production d'un paramètre sachant ce qui a été effectué auparavant. La combinaison de systèmes de diverses natures compliquera, par conséquent, fortement la tâche du module de comparaison.

La solution consisterait à supprimer le processus de comparaison pour effectuer la reconnaissance de signes. Nous allons maintenant voir une proposition d'une architecture légèrement différente qui permettrait de s'affranchir des problèmes d'hétérogénéité au niveau du module de comparaison.

6.4.4 Proposition d'une modification de l'architecture

Cette autre architecture se base sur le fait qu'il n'y a aucun lien entre le vocabulaire des signes bimanuels et ceux des signes monomanuels. Dans le cas contraire, on est obligé d'utiliser un processus de comparaison afin d'effectuer un choix.

La reconnaissance des signes va s'effectuer en un maximum de trois étapes (voir figure 6.17). Tout d'abord, un premier module va effectuer la reconnaissance des différents paramètres des deux mains. Ce module va comporter huit systèmes de reconnaissance, puis un processus chargé de créer un quadruplet (même structure que dans l'architecture originale).

Ensuite, à partir du quadruplet fourni par les systèmes de reconnaissance, on va chercher dans un deuxième module à effectuer un étiquetage du quadruplet à l'aide d'une base de signes bimanuels. Si l'étiquetage réussit, alors on considère que l'on a reconnu un signe bimanuel et l'on émet le couple *(quadruplet/nom)* vers la suite du système (interprétation syntaxique et sémantique). Si l'étiquetage échoue, alors on est probablement en présence d'un ou deux signes monomanuels. On divise le quadruplet en deux quadruplets, chacun contenant uniquement les données concernant la main gauche ou la main droite.

Les deux derniers modules se chargent à partir des deux nouveaux quadruplets d'effectuer un étiquetage pour les signes de la main gauche et de la main droite. Si l'étiquetage réussit pour l'un des deux modules, celui ci envoie alors le signe reconnu vers le module de détection de relations spatiales. Dans le cas où un module n'arrive pas à étiqueter un signe, on renvoie le symbole *inconnu*

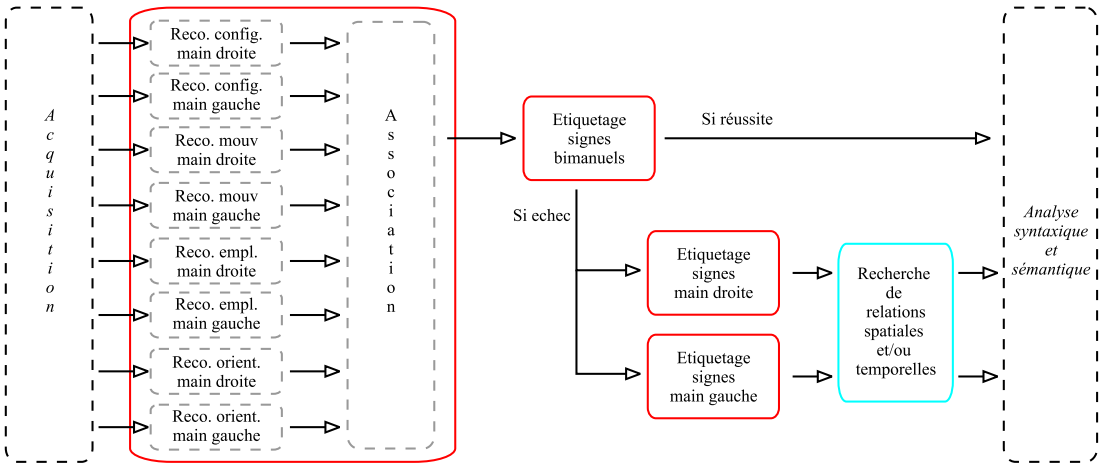

FIG. 6.17 – Modification de l'architecture originale afin de ne pas avoir de module de comparaison.

avec le quadruplet.

Si cette architecture permet de s'affranchir de l'étape de comparaison, elle comporte en revanche un gros risque au niveau du deuxième module. Il suffit d'effectuer un étiquetage erroné à ce niveau pour que toute l'interprétation de la phrase soit incorrecte.
Nous n'avons actuellement pas testé cette architecture et n'envisageons de le faire que lorsque le processus d'étiquetage actuel aura été validé sur des vocabulaires plus conséquents.

6.5 Module de détection de relations spatiales

Du fait que l'expérimentation de la partie comparaison se poursuit, l'implémentation du module de détection des relations spatiales n'a pas été évaluée dans le cadre de la langue des signes. Seule la partie effectuant la resynchronisation des signes a été testée.

6.5.1 Resynchronisation des signes

Cette partie est chargée de trouver les synchronisations temporelles en signes monomanuels. En effet, à la sortie des systèmes de reconnaissance, les signes sont émis de manière désynchronisée du fait qu'ils n'ont pas forcément la même durée.

Pour effectuer cette resynchronisation, nous utilisons deux files de type ***FIFO***[5], une pour la main gauche et une pour la main droite. Au fur et à mesure que les signes monomanuels sont émis par le module de comparaison, ils sont ajoutés à la file correspondante. On va ainsi retrouver dans chacune des files les signes dans l'ordre chronologique.
Dès que chaque file contient chacune au minimum un signe, on peut commencer le processus de resynchronisation. On va alors examiner chaque signe qui est en tête de file. Ces signes sont composés de deux éléments : un nom et un quadruplet. A partir du quadruplet, nous allons pouvoir

[5]First In, First Out. Ce type de file est l'équivalent d'une queue à une caisse de magasin.

calculer le temps de début et de fin du signe. Avec les deux temps calculés pour chacun des signes, on va pouvoir constituer les couples indiquant les périodes partagées par les signes (voir figure 6.18 pour un exemple). L'annexe B donne le détail de l'algorithme utilisé pour effectuer la constitution de ces couples à partir des temps de début et de fin.

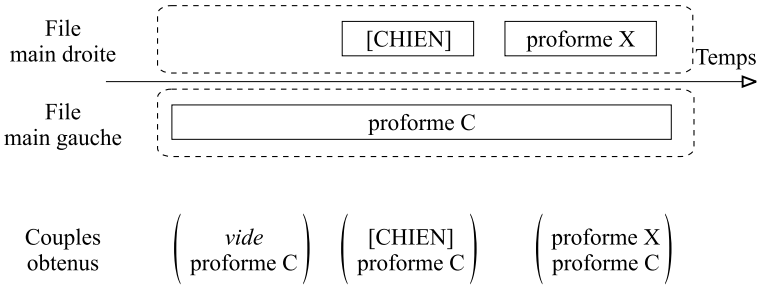

FIG. 6.18 – Exemple des couples créés à partir des signes de la phrase "le chien est dans la voiture". [VOITURE] étant un signe effectué avec les deux mains il n'apparaît bien évidemment pas ici.

Une fois qu'une liste de couples a été constituée à partir des files de signes monomanuels, on va supprimer de cette liste tous les couples dont une des composantes contient *vide*. Il va rester tous les signes monomanuels qui ont été effectués simultanément et qui sont susceptibles d'exprimer une relation spatiale. Il va falloir maintenant rechercher la présence d'une telle relation au sein des quadruplets contenus dans le couple.

6.5.2 Recherche et interprétation de relations spatiales

Pendant la période temporelle que partagent les deux signes, nous allons sélectionner uniquement les moments où ils partagent le même emplacement ou lorsqu'une des deux mains est située dans l'emplacement de l'autre main. Nous allons ensuite chercher à interpréter des relations spatiales entre les deux mains pendant ces instants.
Pour les relations spatiales présentes dans notre corpus (*dans*, *sur* et *sous*), nous utilisons la même approche que celle présentée dans le cadre de l'expérimentation en réalité virtuelle : le système examine la position de la main dominante par rapport à un référentiel centré sur la main dominée (voir section 5.5.3).

Toutefois un tel système manque de finesse pour toutes les relations spatiales où une entité est à la frontière d'une autre. Par exemple, pour une phrase comme "une feuille est sur la table", où l'on aurait une proforme **main plate** sur une autre, le système risque d'interpréter une relation *dans* ou *au dessus*.
D'autre part, comme en réalité virtuelle, les quelques relations spatiales pouvant être traitées par le système présenté dans la section 5.5.3 sont bien insuffisantes. En langue des signes, on a également besoin de pouvoir reconnaître des relations exprimant le fait que deux entités sont parallèles, orthogonales, sécantes, etc. Des travaux restent donc à faire dans ce domaine afin de pouvoir pleinement exploiter les relations spatiales entre les mains.

6.6 Syntaxe et sémantique

Dans cette section, nous abordons des aspects qui ne sont pas traités par le prototype élaboré durant cette thèse. Toutefois dans le cadre d'une application de reconnaissance de la langue des signes, l'architecture doit intégrer, à plus ou moins court terme, les aspects syntaxiques et sémantiques afin d'être capable d'interpréter des énoncés autres que ceux établis pour le corpus du prototype. C'est pourquoi nous discutons, ici, des diverses extensions qui devraient être ajoutées au système et nous indiquons comment elles vont interagir avec le prototype actuel.

6.6.1 Interprétation d'un énoncé

Dans le cadre de notre expérimentation, l'interprétation des énoncés ne pose aucun problème car la structure syntaxique de ces énoncés est figée. Nous n'avons alors aucun problème à mettre en relation les signes standard avec leur proforme correspondant (l'un succède forcément à l'autre) et nous pouvons identifier quelle entité joue le rôle de référence spatiale et quelle est celle qui joue le rôle d'actant (la première entité produite est forcément la référence spatiale).
Pour des énoncés naturels en langue des signes, il est évident que notre structure figée est totalement inopérante. Il faut alors ajouter à notre architecture un module capable de gérer la structure syntaxique d'une phrase. Plusieurs modélisations de la syntaxe existent dans le domaine du traitement informatique de la langue des signes ; nous allons maintenant en présenter quelques unes.

Syntaxe temporelle

Dans un premier temps, les informaticiens ont repris les outils utilisés dans le traitement des langues orales et écrites, car on retrouve en langue des signes des structurations similaires (comme par exemple la décomposition d'une phrase en syntagmes). Par exemple, dans [Starner et al., 1996], les auteurs utilisent une grammaire structurale pour interpréter des phrases basées sur un vocabulaire contenant une quarantaines de signes ; et on retrouve bien évidemment la grammaire générative dans le domaine de la synthèse de phrases en langue des signes.

Dans le cas de notre corpus, la grammaire utilisée aurait pu être la suivante[6] :

$$P \rightarrow \quad ss \begin{pmatrix} pr_g \\ \epsilon \; ss_d \; pr_d \end{pmatrix}$$

$$\Big| \; ss \begin{pmatrix} pr_g \\ \epsilon \end{pmatrix} ss \begin{pmatrix} pr_g \\ pr_d \end{pmatrix}$$

Du fait de cette réutilisation de techniques conçues pour les langues orales et écrites, la modélisation que l'on obtient pour la langue des signes suit uniquement l'évolution temporelle des signes. Or la syntaxe de la langue des signes peut également utiliser des critères spatiaux.

[6] ss : signe standard, pr : proforme, ϵ dénote l'absence de signe, la notation en vecteur indique les signes produits simultanément par la main gauche et la main droite. Note : si jamais on traitait le regard, il faudrait le faire figurer dans la grammaire.

Syntaxe spatiale

Actuellement, très peu de systèmes informatiques ont essayé de modéliser cet aspect de la langue des signes. Dans [Sagawa et Takeuchi, 1999] et [Braffort, 1996a] les auteurs proposent des architectures permettant d'identifier les sujets et objets pour les verbes directionnels, cette identification étant basée sur les emplacements de départ et d'arrivée des verbes. Plus récemment, une modélisation de l'espace de signation a été proposée dans le cadre d'un système d'analyse d'images [Lenseigne, 2004]. On assigne ainsi à chaque entité présente dans le discours une position dans l'espace (une entité pouvant être un lieu, une date, une personne, un objet, etc).
Si ces modélisations de l'espace constituent une première étape, on est encore loin d'avoir une grammaire opérationnelle pour gérer, de manière générale, l'aspect spatial de la langue des signes.

Grammaire cognitive

Les modélisations formelles des aspects temporels, dont nous avons parlé précédemment, sont contraintes à traiter des phrases correctement formées (i.e : qui respectent les règles de la grammaire) ; or une langue - surtout dans le cadre de dialogues - ne suit pas forcément des règles rigides. Par exemple, en langue des signes, le signeur peut très bien décider qu'il énoncera d'abord la proforme et plus tard l'entité liée à cette proforme, ce qui remettrait en cause la grammaire que nous avons donnée ci-dessus.

Une solution consiste à se positionner à un niveau supérieur à la syntaxe : c'est le cas des grammaires cognitives. Ces grammaires partent du principe qu'une phrase est liée au processus cognitif de la personne qui l'émet. On va alors chercher à modéliser une structuration sémantique plutôt que syntaxique.
De nombreux travaux ont été menés dans le domaines de ces grammaires, tant pour les langues orales et écrites que pour la langue des signes ([Langacker, 1986], [Liddell, 1995], [Talmy, 2000], etc). Dans le cadre de notre travail, nous allons nous intéresser à une grammaire bien particulière : la Grammaire Applicative et Cognitive [Desclés, 1990].

Cette grammaire, à l'origine prévue pour les langues orales, peut être appliquée à la langue des signes [Risler, 2000], et plus particulièrement à notre cas des relations spatiales comme on peut le voir dans [Lejeune, 2004] où l'auteur traite, entre autres, les relations topologiques.

La modélisation d'une phrase se fait au travers de ce que l'on appelle un **schème sémantico-cognitif**. Dans ce schème, on va appliquer à un ou plusieurs atomes (i.e. les entités de la phrase) des opérateurs qui vont caractériser la structure sémantique de la phrase. Différentes catégories d'opérateurs existent :

- typage. Par exemple : LOC donne un type lieu[7], ce qui permet d'utiliser le résultat comme référence spatiale.
- détermination. L'opérateur DET sert à caractériser une entité, ce qui se traduit par l'utilisation d'une proforme qui est choisie en fonction des traits saillants de l'entité.

[7] Dans cette modélisation sous forme de schèmes sémantico-cognitifs, chaque entité est typée. Un type sémantico-cognitif correspond à la représentation cognitive que l'on a d'une entité qui peut être perçue comme étant un lieu, un tout massif, en ensemble, etc. Ici l'opérateur LOC permet d'indiquer que l'on perçoit une entité comme un lieu.

- topologie. Pour caractériser une frontière, on dispose de l'opérateur FR; pour un intérieur, de l'opérateur IN, etc.
- relationel. Ce type d'opérateur permet de mettre en relation deux éléments. Pour les relations spatiales, il s'agit de REP.

Le schème de base d'une relation spatiale entre deux entités X et Y est décrit par la notation suivante : $X\ REP\ Y$ (X est repéré par rapport à Y). Suivant la relation spatiale que l'on va vouloir ensuite exprimer entre les deux entités (*dans*, *sur*, *devant*, etc), on va appliquer différents opérateurs topologiques à l'entité référence.

FIG. 6.19 – "Le chat est dans la voiture". (LS-DANCE, LIMSI-CNRS)

Par exemple dans le cas de la phrase "Le chat est dans la voiture" indiquée dans la figure 6.19), on va obtenir le schème suivant(sa construction est détaillée dans [Braffort et Lejeune, 2006]) :

$$DET(chat)\ REP\ IN(DET(LOC(voiture)))$$

$LOC(voiture)$ type l'entité *voiture* comme pouvant servir de lieu, $IN(DET(...))$ indique que l'on s'intéresse à l'intérieur de la détermination (dans ce cas, l'intérieur de la proforme C).
$DET(chat)\ REP\ Y$ exprime le fait que l'on situe la détermination du *chat* par rapport au lieu Y. Autrement dit, on ne modélise pas comment est organisée la phrase, mais bien l'idée que l'entité *chat* se situe à l'intérieur de l'entité *voiture*.
Toutefois ce schème, malgré le fait qu'il soit une unique représentation pour différentes formulations possibles, reste propre à la langue des signes où l'on utilise des proformes (opérateur DET). Pour passer à une autre langue (le français oral par exemple), il faudrait passer par un schème plus "généraliste" pour ensuite générer un schème propre à la langue ciblée.

Implémentation dans le cas de notre architecture

A la sortie de notre système de reconnaissance, nous allons avoir une liste contenant les différents éléments successivement reconnus. Par exemple pour la phrase "le verre est sur la table", on peut obtenir une liste du type :

$$< [TABLE], \{Mainplate\}, [VERRE], \{Mainplate\} + \{C\}, (SUR) >\ ^{8}$$

Du fait qu'une relation spatiale a été reconnue on sait que cela va se traduire par un schème $X\ REP\ Y$. Ensuite, à partir des éléments reçus en entrée, on va chercher à instancier ce schème.

[8] La notation que nous avons adopté ici est : [SIGNE STANDARD], {Proforme}, (RELATION SPATIALE) et SigneMainDominée+SigneMainDominante

Par exemple, on sait que cette relation spatiale (SUR) concerne la surface d'une entité. On va donc pouvoir en déduire le schème suivant : $X\ REP\ FR(Y)$. Du fait qu'il y a deux proformes qui interviennent dans la relation, on sait que : $DET(X)\ REP\ FR(DET(Y)$. Comme dans une relation spatiale, le repère est forcément un locatif, on force la partie Y à être un locatif : $DET(X)\ REP\ FR(DET(LOC(Y))$. Ensuite, en fonction des deux entités que l'on a (une table est un verre) et des contraintes que l'on a sur les proformes (la proforme pour Y doit pouvoir "porter" X, les proformes de X et Y doivent effectuer une reprise partielle ou globale de X et Y), on va en déduire que Main plate peut être relié à [TABLE] et C à verre, et comme C (ici utilisé selon un axe vertical) ne correspond pas à un support on peut en déduire que X est le verre et Y la table. On obtient au final : $DET(Verre)\ REP\ FR(DET(Table)$.
On a alors obtenu le schème instancié de l'énoncé en langue des signes française. On peut ensuite, à partir de cet énoncé, soit passer vers une autre langue dans le cas d'une traduction, soit effectuer les opérations appropriées dans le cas d'un dialogue homme-machine.

Un système capable d'effectuer l'instanciation d'un schème va avoir besoin de nombreuses connaissances : traits iconiques des entités correspondant aux signes pouvant être reconnus, fonctionnalités et traits iconiques des proformes, patrons de schèmes pour les différentes relations spatiales, etc (figure 6.20).

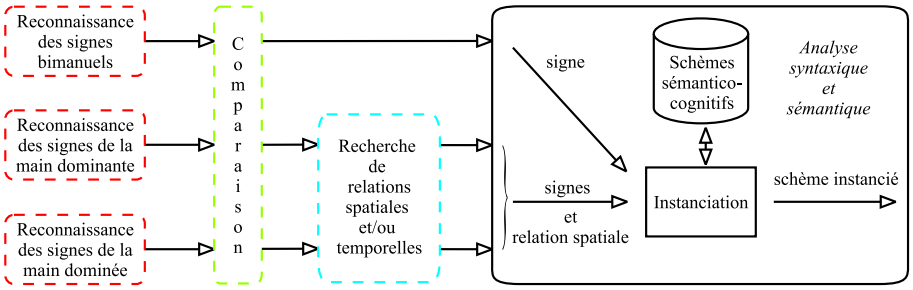

FIG. 6.20 – Architecture permettant l'instanciation d'un schème sémantico-cognitif.

Si un tel système n'a pas été expérimenté dans le cadre de cette thèse, un prototype similaire a toutefois été implémenté dans le cadre d'un atelier sur la langue des signes [Braffort et al., 2005]. Ce prototype s'inscrit dans le développement d'un système permettant de faire signer, par un avatar, une phrase générée à partir du français. Pour cet atelier, le système a été restreint à des phrases décrivant des dispositions géographiques (par exemple "Paris est au nord de Toulouse") et les données fournies en entrée n'étaient pas une phrase de français écrit mais un triplet donnant les entités impliquées et la relation qui les relie (dans le cas de notre exemple $< Paris, nord, Toulouse >$).

La génération de la phrase en langue des signes se fait en plusieurs étapes :

1. Génération d'un schème sémantico-cognitif à partir du triplet fourni,
2. Elaboration d'un énoncé à partir de ce schème,
3. Génération d'un fichier d'animation depuis l'énoncé,
4. Interprétation de ce fichier par un avatar.

Dans le cas de notre architecture de reconnaissance, seule la première étape nous intéresse (on

remplacerait le triplet par les éléments reconnus).

Sachant qu'un schème n'est ni plus ni moins qu'une arborescence (où les feuilles sont des entités et les noeuds internes des opérateurs), générer un schème consiste à trouver une stratégie permettant de construire l'arbre correspondant. Dans le cadre de notre triplet qui décrit une relation géographique entre deux entités, on a déjà un a priori sur la forme de l'arbre et sur l'emplacement de chacune des entités (on sait qui est la référence et qui est le repéré) ; par contre on ne connaît pas les opérateurs qui vont intervenir entre les deux feuilles et la racine. De ce fait, la stratégie adoptée dans l'étape 1 consiste à partir des feuilles (que l'on considère comme deux schèmes séparés au départ), puis d'enrichir au fur et à mesure ces arbres en fonction des informations que l'on peut déduire du triplet (voir l'exemple dans la figure 6.21).

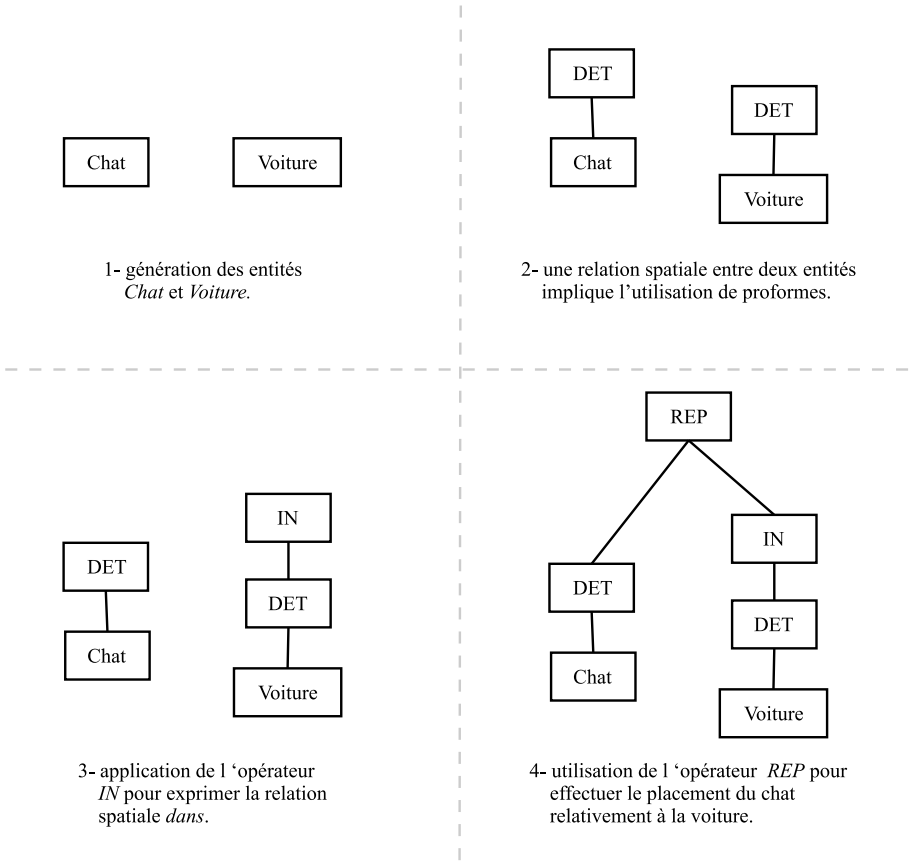

FIG. 6.21 – Exemple de génération de schème à partir du triplet $< chat, dans, voiture >$.

A contrario dans le cas de cette thèse, à la sortie des modules de reconnaissance, on possède peu ou pas d'hypothèses sur la structure de l'arbre et on possède quelques informations sur les noeuds et feuilles possibles (on a des signes standards qui vont donner des feuilles, des proformes qui vont correspondre à des opérateurs DET, etc). Il semble alors logique d'utiliser une stratégie à contre-pied : construire l'arbre à partir de la racine.

6.6.2 Aide à la reconnaissance

Une fois que les premiers signes et/ou relations spatio-temporelles d'une phrase ont été reconnus, du fait que la langue des signes possède une structure syntaxique et que ces éléments reconnus contiennent des informations sémantiques et cognitives, on peut éventuellement en déduire un certain nombre d'informations. Ces informations peuvent servir soit à faire des hypothèses sur les éléments que le signeur va produire dans le futur, soit à vérifier que l'interprétation que l'on a faite de certains éléments est correcte.
Par exemple, si à travers les signes déjà effectués deux entités ont été identifiées, du fait de certaines propriétés syntaxiques (le contexte est décrit avant l'action), on peut faire l'hypothèse que le ou les signes suivants vont servir à mettre en relation ces deux entités. Cette mise en relation peut s'exprimer au travers d'un verbe décrivant une action impliquant les deux entités (par exemple le verbe directionnel [DONNER]), d'une relation spatio-temporelle (voir l'exemple "Le chat est dans la voiture" dans la figure 6.19), d'une interrogation (pour la phrase "Quel pourcentage d'enfants nés sourds ont des parents entendants", le signeur va tout d'abord énoncer les signes pour les parents et les enfants avant de finir par le signe [%] et l'interrogation [COMBIEN ?]), etc.
Si l'on réussit à identifier dans une phrase deux entités et une relation spatiale, la nature même des entités va imposer des contraintes sur la relation spatiale. Par exemple, si l'on a interprété une séquence de signes contenant une entité *table*, une entité *verre* et la relation spatiale *dans* comme étant la phrase "le verre est dans la table", on peut, avec quasi-certitude, indiquer que l'interprétation d'un des éléments est erronée du fait des propriétés physiques d'un verre et d'une table.

Bien entendu, le fait que l'on puisse déduire des informations à partir des éléments précédemment interprétés nous intéresse dans le cadre de notre architecture. Par exemple, les propriétés de la syntaxe pourraient nous indiquer à quels moments des points de synchronisation sont susceptibles de se produire, ou à quelle(s) catégorie(s) grammaticale(s) doit appartenir le signe suivant. De même, les propriétés sémantico-cognitives des éléments déjà reconnus peuvent apporter des contraintes sur les futurs éléments à reconnaître. Cela se traduirait dans notre architecture par des retours d'information depuis le module d'analyse syntaxique et sémantique vers le module de recherche de relations spatio-temporelles, le module de comparaison, voir aussi les modules de reconnaissance (voir figure 6.22).

Etant donné qu'aucun module d'analyse syntaxique et sémantique n'a été implémenté dans notre prototype, de tels retours d'information n'ont pu être expérimentés. Malgré la complexité qu'il y aurait à mettre en place ce contrôle de la reconnaissance par la syntaxe et la sémantique, un tel mécanisme nous semble toutefois préférable plutôt qu'utiliser uniquement les capacités des systèmes de reconnaissance qui sont limités à un cadre "rigide".

6.7 Bilan et perspectives

Bilan

Le développement du prototype que nous avons détaillé dans ce chapitre n'a pu être achevé durant cette thèse. De ce fait, il n'a pu être complètement testé sur le corpus présenté en début de ce chapitre, ni être éprouvé avec un vocabulaire plus important. Toutefois un certain nombre de faits peuvent être tirés des implémentations et expérimentations qui ont pu être réalisées.

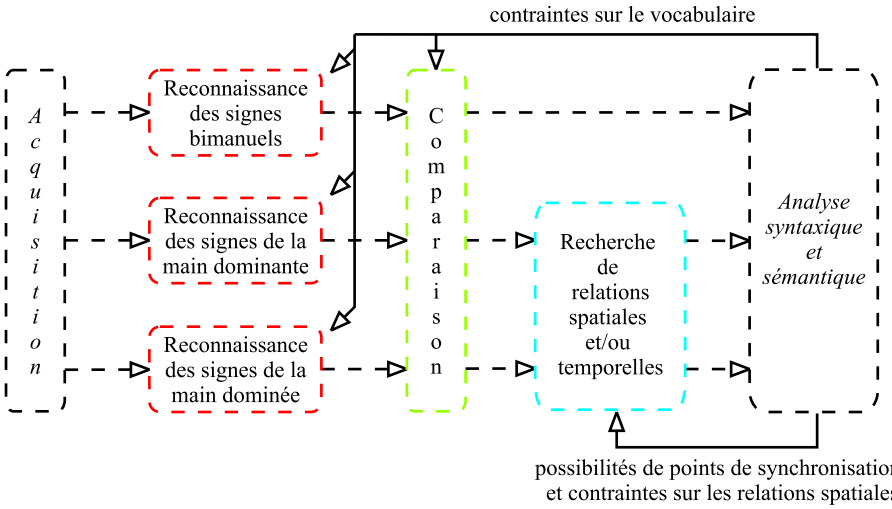

FIG. 6.22 – Influence de l'analyse syntaxique et sémantique sur les autres modules de l'architecture.

Pour ce qui concerne la synchronisation des données, le bilan reste mitigé. Si l'on a pour cible d'effectuer de la reconnaissance en temps réel (ce qui est notre cas), on est obligé de gérer le fait qu'il faut traiter les données en même temps qu'on les acquiert au fur et à mesure depuis les périphériques. Pour cela, les implémentations utilisées ici (prototype multithread ou basé sur l'architecture EVI3d) sont tout à fait adaptées. Par contre, si l'on n'a pas cet impératif du temps réel, il est clair que les solutions proposées ici sont beaucoup trop complexes.
Quant aux essais pour avoir les données les plus synchrones possibles (que ce soit entre systèmes de reconnaissance ou entre périphériques), ils s'avèrent peu concluants. Le fait que l'on ne perçoive pas de différence avec une simple synchronisation est probablement dû à la fréquence élevée des périphériques et à la capacité des systèmes de reconnaissance à traiter la variabilité des données en entrée. Il est possible que dans le cas de périphériques ayant une fréquence beaucoup plus lente (donc avec un écart maximal entre les trames de deux périphériques plus importants), une synchronisation simple ne soit plus suffisante, et l'implémentation présentée dans ce chapitre serait alors plus intéressante.

Du côté des modules de reconnaissance, le fait de décomposer le signe en plusieurs éléments n'est pas nouveau dans le domaine de la reconnaissance de la langue des signes. Par contre la manière de le mettre en oeuvre est originale et prometteuse.
En plus de cette partition en plusieurs paramètres, nous décomposons, au sein d'un signe, ces paramètres en plusieurs éléments. Cela permet de prendre en compte les variations d'un paramètre au cours d'un signe. En particulier, nous ne considérons pas le mouvement comme un tout entre deux moments où la main est immobile, mais comme une succession d'éléments composés d'une courbure, d'une vitesse et d'une direction.
D'autre part, une fois qu'un signe est étiqueté, nous préservons la structure qui contient les valeurs des paramètres afin qu'elles puissent servir pour les traitements qui suivent (interprétation de relations spatiales, analyse sémantique, etc), alors que traditionnellement un système de reconnaissance se contente d'émettre le symbole reconnu, ce qui peut se révéler limitant en terme d'informations lors d'une étape d'interprétation sémantique assez poussée.

Enfin, nous mêlons ici différentes manières d'interpréter les paramètres au sein d'un même système (pour les configurations et le mouvement nous utilisons un modèle statistique, tandis que pour l'emplacement et l'orientation nous utilisons de simples calculs géométriques). Ce choix est motivé pas la différence de nature des paramètres. On ne peut pas reconnaître de manière correcte les quatre paramètres, avec un seul et unique type de système de reconnaissance, du fait que certains de ces paramètres sont de nature discrétisable et d'autres sont de caractère continu.
Au final, il est possible que notre système soit moins performant et abouti que certains autres en ce qui concerne la taille du vocabulaire, l'apprentissage et la mise en oeuvre. Mais de par ses caractéristiques, il nous semble plus à même de traiter un vocabulaire non figé et de donner la possibilité de traiter des phrases complexes, sans pour autant figer la structure des phrases.

Les quelques expérimentations qui ont eu lieu au niveau de la comparaison de signes montrent que de nombreux paramètres peuvent jouer : apprentissage, choix des calculs de scores, détermination de seuils. L'implémentation présentée ici, permet une première approche du problème. Toutefois nous avons conscience que le module de comparaison obtenu est rudimentaire et insuffisant pour une utilisation avec un vocabulaire plus consistant. C'est pourquoi nous avons également proposé l'alternative de la section 6.4.4 qui reporte le problème vers le module de reconnaissance des signes bimanuels.

Perspectives

Dans le futur un certain nombre de travaux restent donc à faire afin de valider notre architecture dans le cadre de la langue des signes : finaliser et tester le module de comparaison, expérimenter la recherche de points de synchronisation, etc. Si notre implémentation s'avère valide, nous pourrons alors tenter d'étendre l'usage de notre architecture à d'autres relations spatiales et à des phrases ayant une structure un peu plus complexe.

Hormis la perspective de savoir exploiter les signes bimanuels, l'implémentation présentée ici ouvre la porte à d'autres aspects de la langue des signes. Cela est rendu possible par la modélisation que nous avons adoptée pour les signes. En effet, comme nous donnons la priorité à la reconnaissance des paramètres et non pas à l'étiquetage du signe, on se donne la possibilité de traiter des signes non appris par le système, ces signes pouvant être par exemple un spécificateur de taille et/ou de forme, un signe standard modifié pour les besoins du contexte, etc.
D'autre part, notre décomposition poussée des signes permet de stocker des informations sur la manière dont a été réalisée un signe (vitesse du mouvement, configuration exacte utilisée pour un signe...). Or, en plus des mimiques faciales, un signeur peut utiliser la dynamique du geste, les répétitions d'un motif au sein d'un signe, ou d'autres techniques afin de nuancer ou modifier la signification d'un signe. Le fait de garder toutes les informations correspondant aux paramètres permet à une étape d'analyse sémantique d'analyser ces informations transmises par le signeur via les paramètres.
Enfin, la nature de type "système réparti" de l'architecture proposée permet de pouvoir facilement intégrer un module d'analyse de plus haut niveau (syntaxe, sémantique, etc). On a alors la possibilité de concevoir un système de reconnaissance mixte ascendant-descendant.

Conclusion et Perspectives

Au travers de ce manuscrit nous avons présenté l'architecture que nous proposons afin de traiter les gestes bimanuels. Cette architecture a été complètement ou partiellement implémentée dans les domaines des la réalité virtuelle ou de la langue des signes. Lorsque l'on regarde dans le détail ces deux implémentations, elles possèdent beaucoup de différences : modules de reconnaissance différents ; présence d'un module de comparaison d'un côté, absence de l'autre ; etc. Malgré ces dissimilitudes, les deux prototypes suivent la même trame proposée dans le chapitre 3.
C'est ce fait, voulu dès le départ, qu'il nous semble important de retenir. L'architecture permet une grande modularité nécessaire pour être utilisée dans divers cadres d'application. Il ne faut pas la voir comme un tout dont la mise en oeuvre serait ardue mais plutôt comme un système multi-agent dont chaque élément peut être mis au point séparément (modulo le fait qu'ils puissent dialoguer entre eux bien évidemment).
Ce n'est donc pas tellement les implémentations présentées ici pour la réalité virtuelle et la langue des signes qu'il faut retenir, mais le processus réalisé par l'ensemble de l'architecture :

1. reconnaître les gestes selon la catégorie à laquelle ils appartiennent (geste main gauche, geste main droite, geste à deux mains),
2. sélectionner le "bon" geste parmi ceux reconnus (si cela est nécessaire),
3. si un geste de la main droite et un geste de la main gauche sont présents simultanément, rechercher un point de synchronisation entre eux (généralement synchronisation spatio-temporelle dans le cas de gestes sémiotiques, synchronisation sur un objet d'intérêt commun pour des gestes ergotiques), puis interpréter l'information ou action exprimée par les deux mains,
4. traiter les éléments reconnus (cela peut être une analyse syntaxique/sémantique, la modification d'objets dans une scène virtuelle, etc).

Avec les étapes 1 et 2, on traite le problème de la confusion entre signes monomanuels et signe bimanuel. L'étape 3 permet, elle, d'exploiter les informations contenus par une éventuelle conjonction des deux mains. Au travers du processus réalisé par l'architecture, quelle que soit l'implémentation choisie pour chaque module, on est normalement assuré de résoudre les problèmes liés à l'utilisation des mains.

On peut relever le fait que notre architecture n'a été testée que pour des gestes de la langue des signes (l'expérimentation en réalité virtuelle étant faite sur des gestes tirés de la langue des signes), et qu'il n'est pas forcément aussi générique que nous le pensons.
Toutefois, du fait que la langue des signes intègre la plupart des mécanismes gestuels présents dans d'autres domaines, si l'on arrive à traiter tous les aspects de l'interaction à deux mains pour ce domaine, il est probable que notre approche soit valide pour d'autres cas (comme par exemple les

gestes coverbaux). Actuellement nous savons gérer, pour la langue des signes, les signes bimanuels et les relation spatiales. L'objectif suivant est alors de savoir exploiter les relations temporelles qui peuvent exister entre les mains.
Par contre, pour ce qui est du domaine sensorimoteur, il n'y a pas d'équivalent en langue des signes. Nous pensons tout de même que le principe de fonctionnement de notre architecture reste valable pour ces gestes, la différence se faisant surtout au niveau du contenu des modules.

La deuxième difficulté concerne la deuxième étape réalisée par notre architecture. En effet, sélectionner le "bon" geste (i.e savoir si l'on a affaire à deux gestes monomanuels ou un geste bimanuel) peut s'avérer très difficile. Nous donnons une première piste d'alternative dans la section 6.4.4, toutefois cette solution ne fait que reporter le problème vers le système de reconnaissance qui doit être alors capable de déterminer s'il a correctement reconnu un geste ou s'il a fait une erreur. Une deuxième solution consiste à utiliser un système de reconnaissance qui soit capable de reconnaître à la fois des gestes monomanuels simultanés et des gestes bimanuels. C'est le cas, par exemple, des modèles de Markov parallèles (voir la partie fusion tardive dans la section 3.2.3) ; par contre cela impose du coup un type de système de reconnaissance bien particulier (qui ne sera peut être pas à même de traiter un paramètre de nature continue comme l'orientation).
C'est pourquoi, selon les contraintes que l'on a ou les objectifs que l'on se fixe, cette étape de comparaison peut s'avérer obligatoire afin de traiter correctement les différents cas d'interaction entre les deux mains.

En conclusion, on peut dire que malgré les défauts énoncés précédemment, le schéma d'architecture que nous proposons dans cette thèse contribue à faire progresser le traitement des interactions à deux mains car elle permet à la fois de différencier les gestes monomanuels des gestes bimanuels et de traiter les relations spatiales voir temporelles entre les deux mains.
C'est sur ce point que notre système innove par rapport à la plupart des systèmes de reconnaissance qui considèrent les deux mains d'un point de vue global, ou par rapport aux quelques rares systèmes qui sont capables de discriminer gestes monomanuels et bimanuels mais qui n'exploitent pas les informations exprimées par la synchronisation de deux gestes simultanés. En particulier, à notre connaissance, il n'y a actuellement aucun autre système qui est capable de traiter des informations de type spatio-temporel entre les mains

Dans les futurs développements à réaliser il nous faudra chercher à réduire ou éliminer ces défauts. Dans les perspectives des chapitres 5 et 6, nous avons déjà donné des objectifs qui répondent à cette demande. Tout d'abord l'expérimentation des gestes de manipulation et de modification d'objets en réalité virtuelle permettra de vérifier la possibilité de traiter des gestes sensorimoteurs avec notre architecture. Ensuite, nous avons prévu, en langue des signes, de continuer à explorer les relations entre signes monomanuels, et entre autres les relations temporelles.
Une fois ces travaux aboutis et s'ils s'avèrent concluants, nous pensons que l'on pourra considérer l'architecture comme étant opérationnelle quel que soit le domaine d'application. Il ne restera plus que le problème de la différenciation entre gestes monomanuels et geste bimanuel à résoudre. Il nous semble que ce point dur a un aspect propre à chaque situation. Par exemple, dans le cas de notre prototype en réalité virtuelle il n'apparaît pas. Par contre, en langue des signes, le prototype ne peut utiliser des modèles de Markov parallèles du fait de la décomposition que nous voulons avoir pour la modélisation des signes. Ce sera à chacun de chercher une solution, probablement en

adaptant les solutions existantes (modèles de Markov parallèles, report du choix sur le module de reconnaissance des gestes à deux mains, etc) à son implémentation des modules de reconnaissance.

Bibliographie

[Battison, 1978] Battison, R. (1978). *Lexical borrowing in American Sign Language.* Silver Spring, Linstok Press.

[Bauer et Kraiss, 2001] Bauer, B. et Kraiss, K. F. (2001). Towards an automatic sign language recognition system using subunits. In *Gesture and Sign Language in Human-Computer Interaction*, LNAI 2298, pages 64–75. Springer-Verlag.

[Belaïd et Belaïd, 1992] Belaïd, A. et Belaïd, Y. (1992). *Reconnaissance des formes, Méthodes et applications.* InterEditions.

[Bier et al., 1994] Bier, E., Stone, M., Fishkin, K., Buxton, W., et Baudel, T. (1994). A Taxonomy of See-Through Tools. In *BIBCHI*, pages 358–364. Addison-Wesley.

[Bimber, 1999] Bimber, O. (1999). *Gesture-Based Interaction in Virtual Environments.* PhD thesis, Université de Darmstad.

[Bossard, 2002] Bossard, B. (2002). Problèmes posés par la reconnaissance de gestes en langue des signes. In *TALN & RECITAL 2002*, pages 445–454. ATALA.

[Bossard, 2005] Bossard, B. (2005). Gestural interaction for virtual scene description. In *Gesture Workshop'05.*

[Bossard et al., 2003] Bossard, B., Braffort, A., et Jardino, M. (2003). Some issues in sign language processing. In *Gesture-Based Communication in Human-Computer Interaction. Gesture Workshop'03*, LNAI 2915. Springer-Verlag.

[Bossard et al., 2004] Bossard, B., Convard, T., Braffort, A., Touraine, D., Bourdot, P., et Jardino, M. (2004). Un système de reconnaissance de gestes générique pour la réalité virtuelle. In *Actes de la Conférence Reconnaissance des Formes et Intelligence Artificielle (RFIA 2004).*

[Bourdot, 2001] Bourdot, P. (2001). Projet venise : Virtualité et environnement immersif pour les sciences experimentales. Comité scientifique venise, LIMSI-CNRS.

[Bourdot et Touraine, 2002a] Bourdot, P. et Touraine, D. (2002a). A hierarchical server to manage multimodal and collaborative interactions within virtual environments. In *Proceedings of International Conference on Applied Informatics.*

[Bourdot et Touraine, 2002b] Bourdot, P. et Touraine, D. (2002b). Polyvalent display framework to control virtual navigation with 6dof tracking. In *Proceedings of Virtual Reality Conference.*

[Boutet, 2001] Boutet, D. (2001). *Approche morpho-dynamique du sens dans la gestuelle conversationelle.* PhD thesis, Université Paris 8.

[Braffort, 1996a] Braffort, A. (1996a). Argo : An architecture for sign language recognition and interpretation. In Harling, P. et Edwards, A., editors, *Progress in Gestural Interaction*, Lectures notes in Artificial Intelligence, pages 17–30. Springer.

[Braffort, 1996b] Braffort, A. (1996b). *Reconnaissance et compréhension de gestes, application à la langue des signes*. PhD thesis, Université de Paris XI Orsay.

[Braffort et Lejeune, 2006] Braffort, A. et Lejeune, F. (2006). Spatialised semantic relations in french sign language : Toward a computational modelling. In *Gesture and Sign Language in Human-Computer Interaction. Gesture Workshop'05*. Springer-Verlag.

[Braffort et al., 2005] Braffort, A., Segouat, J., Bossard, B., Bolot, L., et Lejeune, F. (2005). Modélisation des relations spatiales en langue des signes française. In *TALN & RECITAL 2005*, pages 333–337.

[Brand, 1997] Brand, M. (1997). Coupled hidden markov models for modeling interacting processes. Learning and Common Sense Tech. Report 405, MIT Media Lab Perceptual Computing.

[Cadoz, 1994] Cadoz, C. (1994). Le geste canal de communication homme-machine - la communication "instrumentale". *Sciences Informatiques*, 13(1) :31–61. Numéro Spécial : Interface Homme-Machine.

[Cassel et Collet, 2003] Cassel, R. et Collet, C. (2003). Tracking of real time acrobatic movements by image processing. In *Gesture-Based Communication in Human-Computer Interaction. Gesture Workshop'03*, LNAI 2915, pages 164–171. Springer-Verlag.

[Colin, 2004] Colin, B. (2004). Langage gestuel et contexte spatial pour l'interaction 3d en réalité virtuelle. Master's thesis, Université Paris 11, Orsay.

[Collet, 1993] Collet, C. (1993). Reconnaissance de gestes par réseaux neuromimétiques. Master's thesis, Université Paris XI Orsay.

[Convard et Bourdot, 2003] Convard, T. et Bourdot, P. (2003). A multimodal approach for computer aided design in immersive environments. In *Proceedings of Virtual Concept*.

[Cruz-Neira et al., 1993] Cruz-Neira, C., Sandin, D., et DeFanti, T. (1993). Surround-screen projection-based virtual reality : The design and implementation of the cave. In *Proceedings of SIGGRAPH'93*.

[Cutler et al., 1997] Cutler, L. D., Frohlich, B., et Hanrahan, P. (1997). Two-handed direct manipulation on the responsive workbench. In *Symposium on Interacting 3D Graphics*, pages 107–114.

[Cuxac, 2000] Cuxac, C. (2000). *La langue des Signes Française (LSF), les voies de l'iconicité*, chapter 3. Faits de Langues. OPHRYS.

[Cuxac, 2003] Cuxac, C. (2003). *La langue des signes, statuts linguistiques et institutionnels*. Langue Française. Larousse.

[Delamarre et Faugeras, 1998] Delamarre, Q. et Faugeras, O. (1998). Finding pose of hand in video images : a stereo-based approach. In *Proceedings of Face and Gesture'98*.

[Desclés, 1990] Desclés, J. P. (1990). *Langages Applicatifs, Langues Naturelles et Cognition*. Hermès.

[Fang et al., 2004] Fang, G., Gao, W., et Zhao, D. (2004). Large vocabulary sign language recognition based on fuzzy decision trees. In *IEEE Transactions on System Man and Cybernetics*, number 3 in Volume 34, pages 305–314.

[Frohlich et Wachsmuth, 1997] Frohlich, M. et Wachsmuth, I. (1997). Gesture recognition of the upper limbs - from signal to symbol. In Wachsmuth, I. et Fröhlich, M., editors, *Gesture and*

Sigm Language in Human-Computer Interaction, Lectures notes in Artificial Intelligence, pages 173–184. Springer.

[Fuchs et al., 2001] Fuchs, P., Moreau, G., et Papin, J.-P. (2001). *Le traité de la Réalité Virtuelle*, volume 1 of *Sciences Mathématiques et Informatiques*. Ecole des Mines de Paris.

[Gauvain, 2000] Gauvain, J. L. (2000). Systèmes de reconnaissance à grands vocabulaires. In *XXIIIèmes Journées d'Etude sur la Parole*.

[Ghahramani et Jordan, 1996] Ghahramani, Z. et Jordan, M. I. (1996). Factorial hidden markov models. In Touretzky, D. S., Mozer, M. C., et Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 472–478. The MIT Press.

[Girod et Vourc'h, 1981] Girod, M. et Vourc'h, A. (1981). *Dictionnaire bilingue élémentaire*, volume 3 of *La langue des signes*. Ellipse.

[Gourley, 1994] Gourley, C. (1994). Neural networks utilizing posture input for sign language recognition. Technical report computer vision and robotics research laboratory, University of Tenessee Knoxville.

[Guitteny et al., 2004] Guitteny, P., Legouis, P., et Verlaine, L. (2004). La langue des signes. Centre d'Information sur la Surdité d'Aquitaine.

[Harling, 1993] Harling, P. A. (1993). Gesture input using neural networks. Master's thesis, University of York.

[Harling et Edwards, 1996] Harling, P. A. et Edwards, A. D. (1996). Hand tension as a gesture segmentation cue. In *Progree in Gestural Interaction. Gesture Workshop'96*, pages 75–88. Springer-Verlag.

[Hienz et al., 1999] Hienz, H., Bauer, B., et Kraiss, K.-F. (1999). HMM-based continuous sign language recognition using stochastic grammars. In *Gesture and Sign Language in Human-Computer Interaction. Gesture Workshop'99*, LNAI 1739, pages 185–196. Springer-Verlag.

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

[Kadous, 1996] Kadous, M. W. (1996). Machine recognition of auslan signs using powergloves : Towards large-lexicon recognition of sign language. In *Proceedings of the Workshop on the Integration of Gesture in Language and Speech*, pages 165–174, Wilmington.

[Kendon, 1988] Kendon, A. (1988). How gestures can become like words. In *Crosscultural perspectives in nonverbal communication*, pages 131–141. F. Poyatos.

[Kruger et al., 1995] Kruger, W., Bohn, C., Frohlich, B., Schuth, H., Strauss, W., et Wesche, G. (1995). The responsive workbench : A virtual work environment. In *IEEE Computer*, number 7 in Volume 28.

[Langacker, 1986] Langacker, R. W. (1986). *Foundation of Cognitive Grammar*, volume 1 et 2. Stanford University Press.

[Latoschik, 2001] Latoschik, M. E. (2001). A general framework for multimodal interaction in VR : PrOSA. In *Proceedings of IEE Virtual Reality*.

[Laviola, 1999] Laviola, J. J. (1999). Msvt : A virtual reality-based multimodal scientific vizualisation tool.

[Leganchuk et al., 1998] Leganchuk, A., Zhai, S., et Buxton, W. (1998). Manual and cognitive benefits of two-handed input : An experimental study. In *Transactions on Computer-Human Interaction*, number Vol. 5, Issue 4, pages 326–359. ACM Press.

[Lejeune, 2004] Lejeune, F. (2004). *Analyse sémantico-cognitive d'énoncés en Langue des Signes Française pour une génération automatique de séquences gestuelles.* PhD thesis, Université de Paris XI Orsay.

[Lejeune et al., 2002] Lejeune, F., Braffort, A., et Descles, J.-P. (2002). Study on semantic representations of french sign language sentences. In *Gesture and Sign Language in Human-Computer Interaction. Gesture Workshop'01*, LNAI 2298, pages 197–201. Springer-Verlag.

[Lenseigne, 2004] Lenseigne, B. (2004). *Intégration de connaissances linguistiques dans un système de vision, application à l'étude de la Langue des Signes.* PhD thesis, Université Paul Sabatier - Toulouse 3.

[Liang et Ouhyoung, 1998] Liang, R. H. et Ouhyoung, M. (1998). A real-time continuous gesture interface for taiwanese sign language. In *Submitted to UIST'97.*

[Liddell, 1995] Liddell, S. (1995). Real, surrogate and token space : Grammatical consequences in asl. In Emmorey, K. et Reilly, J., editors, *Language, gesture and space*, pages 19–41. Lawrence Erlbaum.

[Liddell et Johnson, 1989] Liddell, S. K. et Johnson, R. E. (1989). *American Sign Language : The phonological base.* Sign Language Studies.

[McNeill, 1992] McNeill, D. (1992). *Hand and Mind : What gestures reveal about thought.* University of Chicago Press.

[Mine, 1995] Mine, M. R. (1995). Virtual environment interaction techniques. Technical report, University of North Carolina.

[Moody, 1983] Moody, B. (1983). *Histoire et Grammaire*, volume 1 of *La langue des signes.* Ellipse.

[Moody, 1986] Moody, B. (1986). *Dictionnaire bilingue élémentaire*, volume 2 of *La langue des signes.* Ellipse.

[Murakami et Taguchi, 1992] Murakami, K. et Taguchi, H. (1992). Gesture recognition using recurrent neural networks. *Proceedings of SIGCHI*, pages 237–242.

[Newby, 1993] Newby, G. B. (1993). Gesture recognition using statistical similarity. In *Proceedings of Virtual Reality Conference.*

[Pratini, 2001] Pratini, E. (2001). New approaches to 3d gestural modeling - the 3d sketchmaker project. In *Proc. of ECAADE.*

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–285.

[Risler, 2000] Risler, A. (2000). *Ancrage perceptivo-pratique des catégories du langage et localisme cognitif à travers l'étude de la motivation des signes et de la spatialisation des relations sémantiques.* PhD thesis, Université Toulouse-le-Mirail.

[Rubine, 1991] Rubine, D. H. (1991). *The Automatic Recognition of Gestures.* PhD thesis, Carnegie-Mellon.

[Sagawa et Takeuchi, 1999] Sagawa, H. et Takeuchi, M. (1999). A method for analyzing spatial relationships between words in sign language recognition. In *Gesture and Sign Language in Human-Computer Interaction. Gesture Workshop'99*, LNAI 1739, pages 197–209. Springer-Verlag.

[Sowa et Wachsmuth, 2001] Sowa, T. et Wachsmuth, I. (2001). Interpretation of shape-related iconic gestures in virtual environments. In *Gesture and Sign Language in Human-Computer Interaction*, LNAI 2298, pages 21–33. Springer-Verlag.

[Starner et Pentland, 1995] Starner, T. et Pentland, A. (1995). Visual recognition of american sign language using hidden markov models. In *Proceedings of ISCV '95*.

[Starner et al., 1996] Starner, T., Weaver, J., et Pentland, A. (1996). Real-time american sign language recognition using desk and wearable computer based video. Perceptual Computing 466, MIT Media Lab. appeared in IEEE PAMI'98.

[Stockoe, 1960] Stockoe, W. (1960). *Sign Language Structure An Outline of the Visual Communication System of the American Deaf.* Studies in Linguistics. University of Buffalo Press.

[Talmy, 2000] Talmy, L. (2000). *Toward a Cognitive Semantics*, volume 1 et 2. MIT Press.

[Touraine, 2003] Touraine, D. (2003). *Interaction naturelle en environnements immersifs - Démonstrateur multimodal et validation sur des applications scientifiques.* PhD thesis, Université de Paris XI Orsay.

[Touraine et al., 2002] Touraine, D., Bourdot, P., Bellik, Y., et Bolot, L. (2002). A framework to manage multimodal fusion of events for advanced interactions within virtual environments. In *Proceedings of Eighth Eurographics Workshop on Virtual Environments.*

[Vercher, 2003] Vercher, J.-L. (2003). Effets de la latence en réalité virtuelle et en téléopération. In *Actes des Journées Nationales de Recherche en Robotique.*

[Vogler et Metaxas, 1997] Vogler, C. et Metaxas, D. (1997). Adapting hidden markov models for asl recognition by using three-dimensional computer vision methods. In *ICSMC'97.*

[Vogler et Metaxas, 1999a] Vogler, C. et Metaxas, D. N. (1999a). Parallel hiden markov models for american sign language recognition. In *Proc. of the International Conference on Computer Vision.*

[Vogler et Metaxas, 1999b] Vogler, C. et Metaxas, D. N. (1999b). Toward scalability in ASL recognition : Breaking down signs into phonemes. In *Gesture and Sign Language in Human-Computer Interaction. Gesture Workshop'99*, LNAI 1739, pages 211–224. Springer-Verlag.

[Wilson et Martinez, 1997] Wilson, D. R. et Martinez, T. R. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intellignece Research.*

[Young et al., 1989] Young, S. J., Russel, N. H., et Thornton, J. H. S. (1989). Token passing : A simple conceptual model for connected speech recognition systems. CUED/F-INFENG 38, Cambridge University Engineering Departement.

Index

Annexe A

Algorithmes des threads pour la synchronisation de données

```
--------------------------------
Variables globales et sémaphores
--------------------------------

//Groupe 1
S-config-G-lecture : sémaphore    // Sémaphores permettant d'éviter des conflits
S-config-D-lecture : sémaphore    // d'accès entre producteur et consommateurs.
S-config-G-ecriture : sémaphore   // Ils permettent également aux modules de
S-config-D-ecriture : sémaphore   // reconnaissance de décider du moment auquel
S-pos-G-lecture : sémaphore       // ils souhaitent acquérir une nouvelle trame.
S-pos-D-lecture : sémaphore
S-pos-G-ecriture : sémaphore
S-pos-D-ecriture : sémaphore

//Groupe 2
S-2G-lecture : sémaphore  // Sémaphores permettant de synchroniser le module
S-2D-lecture : sémaphore  // de reconnaissance de signes bimanuels avec les
S-2G-écriture : sémaphore // deux autres.
S-2D-écriture : sémaphore

//Groupe 3
S-config-G-changement : sémaphore // Sémaphores permettant d'indiquer l'acquisition
S-config-D-changement : sémaphore // d'une nouvelle donnée au niveau de chacun des
S-pos-changement : sémaphore      // périphériques.

config-G : configuration // variable qui stocke la configuration de la main gauche
config-G : configuration // variable qui stocke la configuration de la main droite
pos-G : position/orientation // position/orientation de la main gauche
pos-D : position/orientation // position/orientation de la main droite
```

```
fin : booléen

--------------------------
Initialisation des données
--------------------------

Pour tous les sémaphores lecture du groupe 1 // On autorise uniquement les
  s = 1                                      //  producteurs au départ.

Pour tous les sémaphores écriture du groupe 1 // Les consommateurs sont bloqués
  s = 0                                        // jusqu'à ce que des données soient
                                               // disponibles.
Pour tous les sémaphores du groupe 2
  s = 0

Pour tous les sémaphores du groupe 3
  s = 0

fin = faux
```

```
------------------------------------------------------------------------
Thread producteur pour la configuration (un pour chaque gant numérique)
------------------------------------------------------------------------

changement : booléen
config-récent-X : configuration

   attendre nouvelles données
   changement = vrai

   tant que non(fin) faire

     si nouvelles données disponibles
       alors
         lire les données depuis le périphérique
         copier les données dans config-récent-X

         si non(changement) // Est-ce que l'on a déjà signalé un changement ?
                           // (ce test permet d'éviter d'incrémenter
           alors           //  S-config-X-changement au dessus de 1)
             changement = vrai
             V(S-config-X-changement)     // Indique qu'une nouvelle donnée est
                                          // disponible pour les consommateurs

     si P'(S-config-X-lecture)            // On regarde si les consommateurs ont fini
       alors
         copier config-récent-X dans config-X // Si oui, on stocke dans la variable
                                              // les dernières données reçues.
         changement = faux

         V(S-config-X-ecriture)         // Puis on indique que l'on a fini l'écriture

   fin tant que
```

```
---------------------------------------------------------------
Thread producteur pour la position/orientation gauche et droite
---------------------------------------------------------------

changement : booléen
pos-récent-G : position/orientation
pos-récent-D : position/orientation

   attendre nouvelles données
   changement = vrai

   tant que non(fin) faire

     si nouvelles données disponibles
       alors
         lire les données depuis le périphérique
         copier les données dans pos-récent-G et pos-récent-D

         si non(changement) // Est-ce que l'on a déjà signalé un changement ?
                           // (ce test permet d'éviter d'incrémenter
           alors           //  S-pos-changement au dessus de 1)
             changement = vrai
             V(S-pos-changement)     // Indique qu'une nouvelle donnée est
                                    // disponible pour les consommateurs


     si P'(S-pos-G-lecture)             // On regarde si les consommateurs ont fini
       alors
         si P'(S-pos-D-lecture)         // Si ils ont tous fini
           alors
            copier les pos-récent-G dans pos-G   // On stocke dans les variables
            copier les pos-récent-D dans pos-D   // les dernières données reçues.
            changement = faux

            V(S-pos-G-ecriture)        // Puis on indique que l'on a fini l'écriture
            V(S-pos-D-ecriture)

         sinon                         // Si tous les consommateurs n'ont pas fini
           V(S-pos-G-lecture)          // On rend le sémaphore afin de continuer
                                       // l'acquisition.
   fin tant que
```

```
-------------------------------------------------------------------------------
Thread consommateur de la reconnaissance des signes monomanuels (gauche ou droit)
-------------------------------------------------------------------------------
   tant que non(fin) faire

     P(S-config-X-ecriture)  // On regarde si le producteur a fini
     P(S-pos-X-ecriture)

       V(S-2X-ecriture)     // On indique au système bimanuel que des données sont
                            // prêtes pour la main X


        traiter les données contenues dans config-X et pos-X

      p(S-2X-lecture)       // Attend que le système bimanuel ait fini

     V(S-config-X-lecture)  // On indique que le module a fini et est prêt
     V(S-pos-X-lecture)     // à traiter de nouvelles données.

   fin tant que

-------------------------------------------------------------
Thread consommateur de la reconnaissance des signes bimanuels
-------------------------------------------------------------
   tant que non(fin) faire

     P(S-2G-ecriture)  // On regarde si les modules main gauche et main droite nous
     P(S-2D-ecriture)  // indiquent des données disponibles

        traiter les données contenues dans config-G, config-D, pos-G et pos-D

      P(S-config-G-changement) // On attend que de nouvelles données soient
      P(S-config-D-changement) // disponibles.
      P(S-pos-changement)

     V(S-2G-lecture) // Indique aux deux autres modules de reconnaissance que
     V(S-2D-lecture) // l'on est prêt pour traiter de nouvelles données

   fin tant que
```

Annexe B

Algorithme pour la synchronisation de signes après reconnaissance

```
tant que (file_gauche et file_droite ne sont pas vides)

 si (file_gauche.tête.début > file_droite.tête.début) // Le signe droit précède
   alors                                               //  temporellement le gauche

     renvoyer (file_droite.tête,vide)

     si (file_gauche.tête.début > file_droite.tête.fin) // Le signe droit finit avant
       alors                                            // le gauche donc aucune
          dépiler la tête de file_droite                // synchronisation

     sinon                                          // Ils sont synchrones plus loin
        file_droite.tête.début = file_gauche.tête.début

 sinon
   si (file_gauche.tête.début < file_droite.tête.début) // Le signe gauche précède
     alors                                              //  temporellement le droit

       renvoyer (vide,file_gauche.tête)

       si (file_gauche.tête.fin < file_droite.tête.début) // Le signe gauche finit
        alors                                               // avant le droit donc
           dépiler la tête de la file_gauche                // aucune synchronisation

      sinon                                     // Ils sont synchrones plus loin
         file_gauche.tête.début = file_droite.tête.début

  sinon // Les deux signes partagent une période temporelle
```

```
    renvoyer (file_droite.tête,file_gauche.tête) // On renvoie la période synchrone

    si (file_gauche.tête.fin = file_droite.tête.fin)
      alors
        dépiler la tête de la file_gauche  // On passe aux signes suivants
        dépiler la tête de la file_droite

      sinon
        si (file_gauche.tête.fin > file_droite.tête.fin)  // Le signe gauche est peut-
          alors                                           // être aussi synchrone avec
            file_gauche.tête.début = file_droite.tête.fin // le signe droit suivant
            dépiler la tête de la file_droite

        sinon // Le signe droit est peut-être synchrone avec le signe gauche suivant
            file_droite.tête.début = file_gauche.tête.fin
            dépiler la tête de la file_gauche

fin tant que
```

Annexe C

Crédits pour les illustrations

Par ordre d'apparition, les illustrations proviennent des sources suivantes :

- Corpus vidéo de l'équipe TCI, IRIT
  (`http://www.irit.fr/ACTIVITES/EQ_TCI/page_princ.html`)
- Dictionnaires de l'International Visual Theatre
  ([Moody, 1983], [Moody, 1986], [Girod et Vourc'h, 1981])
- Corpus vidéo du projet LS-COLIN
  (`http://www.irit.fr/LS-COLIN`)
- Manuel d'Aquitaine ([Guitteny et al., 2004], CISA-2004)
- Corpus vidéo du projet ARC-LSF
  (`http://www-sop.inria.fr/robotvis/projects/ARC-LSF/`)
- Corpus LS-DANCE de l'équipe AT-Geste, LIMSI-CNRS
  (`http://www.limsi.fr/Individu/braffort/ATGeste/`)
- Site de l'équipe Robotvis, INRIA Sophia-Antipolis
  (`http://www.inria.fr/robotvis/personnel/qdelam/`)
- Site du Movement Group, New York/Stanford universities
  (`http://movement.nyu.edu/projects/nyulab.html`)
- Site de la société Ascension technology
  (`http://www.ascension-tech.com/`)
- Equipe AT-Geste, LIMSI-CNRS
  (`http://www.limsi.fr/Individu/braffort/ATGeste/`)
- Site de la société BARCO
  (`http://www.barco.com/`)
- Numéro du 5 novembre 1997 de "The University Record"
  (`http://www.umich.edu/~urecord/9798/Nov05_97/virtual.htm`)
- Action transversale VENISE, LIMSI-CNRS
  (`http://www.limsi.fr/Recherche/ActionVenise/`)
- Site de la société Polhemus
  (`http://www.polhemus.com/`)
- Thèse d'Oliver Bimber ([Bimber, 1999])
- AI Group, université de Bielefeld
  (`http://www.techfak.uni-bielefeld.de/ags/wbski/`)

Index