Université de Nancy I

U.E.R.Sciences mathématiques

Centre de Recherche en Informatique de Nancy

APPLICATION DES TECHNIQUES DE PROGRAMMATION DYNAMIQUE ET DE QUANTIFICATION VECTORIELLE A LA RECONNAISSANCE DES MOTS ISOLES ET DES MOTS ENCHAINES

THESE

PRESENTEE ET SOUTENUE PUBLIQUEMENT LE 1 AVRIL 1987

POUR L'OBTENTION DU TITRE DE

DOCTEUR DE L'UNIVERSITE DE NANCY I EN INFORMATIQUE

PAR

ANNE BOYER

COMPOSITION DU JURY:

PRESIDENT	R.MOHR
RAPPORTEURS	G. CHOLLET J.P. THOMESSE
EXAMINATEURS	M.C. HATON J.P. HATON J.M. PIERREL
INVITE	J. di MARTINO

Université de Nancy I U.E.R.Sciences mathématiques

Centre de Recherche en Informatique de Nancy

APPLICATION DES TECHNIQUES DE PROGRAMMATION DYNAMIQUE ET DE QUANTIFICATION VECTORIELLE A LA RECONNAISSANCE DES MOTS ISOLES ET DES MOTS ENCHAINES

THESE

PRESENTEE ET SOUTENUE PUBLIQUEMENT LE 1 AVRIL 1987

POUR L'OBTENTION DU TITRE DE

DOCTEUR DE L'UNIVERSITE DE NANCY I EN INFORMATIQUE

PAR

ANNE BOYER

COMPOSITION DU JURY:

PRESIDENT	R.MOHR
RAPPORTEURS	G. CHOLLET J.P. THOMESSE
EXAMINATEURS	M.C. HATON J.P. HATON J.M. PIERREL
INVITE	J. di MARTINO

A tous les miens,

avec toute mon affection.

Jean-Paul Haton, professeur à l'Université de Nancy I, a bien voulu m'accueillir dans son laboratoire et assurer la direction de cette thèse, acceptant ainsi de guider mes premiers pas sur les chemins parfois exaltants de la recherche. Qu'il trouve ici l'expression de ma très profonde reconnaissance pour avoir su inspirer ce travail.

Roger Mohr, professeur à l'INPL, me fait l'honneur de présider ce jury, je tiens à lui dire ma vive gratitude.

Je remercie sincèrement Jean-Pierre Thomesse, professeur à l'INPL, qui, rapporteur de ce travail, manifeste une fois encore l'attention bienveillante dont il a toujours fait preuve à mon égard. Je n'oublie pas qu'il a su éveiller mon intérêt pour l'informatique lorsque j'étais son élève à l'ENSEM, je lui en suis infiniment redevable.

Je suis profondément reconnaissante à Gérard Chollet, chargé de recherche à l'ENST, d'avoir bien voulu être rapporteur de cette thèse malgré l'importance des tâches qu'il assume.

Je tiens à exprimer mes vifs remerciements à Jean-Marie Pierrel, professeur à l'Université de Nancy I, pour la sollicitude dont il a toujours fait preuve à mon égard et pour les conseils amicaux qu'il m'a prodigués.

Je remercie Marie-Christine Haton, maître de conférences à l'Université de Nancy I, du témoignage d'amitié qu'elle me donne en participant au jury de ma thèse.

Joseph di Martino, maître-assistant à l'Université de Nancy I, prouve par sa présence dans ce jury l'intérêt qu'il a toujours porté à cette recherche. Qu'il soit vivement remercié pour l'aide qu'il m'a apportée.

Mes remerciements vont à tous ceux qui m'ont aidée, particulièrement à François Charpillet, Dominique Fohr, Yifan Gong, Gisèle Laurette, Odile Mella, Thierry Mondot et Li Wu, qui n'ont épargné ni leur temps ni leur peine.

Merci enfin à tous ceux qui ont la gentillesse de me servir de locuteurs et à tous mes camarades du CRIN pour leur appui amical.

PLAN

INTRODUCTION

Les ordinateurs modernes sont très utiles et très efficaces mais ils seraient incontestablement plus agréables d'emploi s'ils nous comprenaient mieux.

Or l'un des moyens de communication le plus évolué est la parole: rapide, la communication orale n'exige pratiquement aucun effort, ne nécessite aucun contact physique, n'immobilise pas le corps et laisse les mains libres.

Dès lors, l'intérêt de recherches en reconnaissance de la parole s'impose et c'est dans ce cadre que s'insère le travail que nous présentons: reconnaissance de mots isolés et enchainés à l'aide de la technique de programmation dynamique.

Nous rappellerons tout d'abord quelques généralités sur la parole.

La partie A indiquera le principe général d'un système de reconnaissance globale avant de s'intéresser brièvement à quelques méthodes permettant de réaliser l'apprentissage et aux principales techniques d'analyse du signal que l'on utilise en codage de la parole.

La partie B rappelera la technique de programmation dynamique pour la reconnaissance de mots isolés. Des méthodes de réduction de l'espace de comparaison seront présentées afin de diminuer le nombre de calculs nécessaires et donc le temps de reconnaissance. Enfin, pour tenir compte d'éventuelles erreurs de segmentation des mots (détection imparfaite des début et fin d'élocution), nous détaillerons un algorithme de programmation dynamique qui autorise le relâchement aux frontières de mots. Ces diverses méthodes seront comparées à l'aide de tests effectués sur un corpus constitué de répétitions par divers locuteurs des dix chiffres.

La partie C décrira des méthodes de reconnaissance de mots isolés utilisant la technique de quantification vectorielle comme étape préliminaire à la reconnaissance afin de réduire de façon significative la quantité d'informations à garder en mémoire. Diverses techniques de

reconnaissance seront testées sur le même corpus que précédemment afin de comparer leurs performances (temps de calcul, place mémoire occupée, taux de reconnaissance....).

La partie D sera consacrée à la reconnaissance de mots enchainés. Après avoir rappelé l'algorithme de Bridle-Nakagawa, nous verrons comment il est possible d'introduire des contraintes syntaxiques qui permettent de réduire le nombre de comparaisons au prix de contraintes supplémentaires pour le locuteur. Ensuite, nous utiliserons la technique de quantification vectorielle pour réduire la place mémoire nécessaire au stockage des données. Avant de conclure cette étude, nous exposerons un algorithme de reconnaissance de la parole continue qui permet de tenir compte d'éventuelles liaisons à la frontière des mots et présenterons une approche qui utilise à la fois la programmation dynamique et des informations fournies par des analyses plus fines du signal (détection de quelques classes de phonèmes).

GENERALITES

QUELQUES GENERALITES

Réaliser une machine à son image a toujours été un vieux rêve de l'être humain, rêve qui commence à se concrétiser avec la sophistication sans cesse croissante de certains robots actuels.

Si de nombreuses fonctions humaines peuvent être remplies ou simulées au moyen de calculateurs et d'organes mécaniques, il en est une dont la reproductibilité est particulièrement délicate: la parole.

"Au commencement était le verbe", il est vrai que la parole représente certainement un mode de communication privilégié: il n'est rien de plus naturel ni de plus commode pour l'homme que de parler lorsqu'il désire communiquer avec autrui. Instaurer un véritable dialogue homme-machine s'avère donc fondamental, les développements actuels de l'informatique montrent la nécessité d'étudier, tant sur le plan technologique qu'ergonomique et économique, les outils d'une communication véritable entre l'homme et la machine.

La mise en oeuvre de terminaux vocaux (c'est à dire capables de parler intelligiblement et de comprendre la parole de l'utilisateur) est susceptible de modifier considérablement la situation de l'homme face à la machine, puisque la parole libère l'opérateur de son poste de travail. Au lieu de garder le regard rivé sur son écran, celui-ci peut se déplacer, effectuer une autre tâche, en attendant d'être averti par le terminal du déroulement de la première tâche et d'éventuelles actions à effectuer sur celle-ci.

Moyen d'expression le plus spontané de l'homme, la communication vocale permet d'agir quasi instantanément sur la conduite d'un processus alors que la frappe d'instructions au clavier reste une opération souvent longue et fastidieuse. En outre, la parole peut être utilisée de nuit comme de jour et contourner les obstacles, ce que ne fait pas la vision directe.

S'il est vrai que la communication verbale est un moyen d'échange privilégié de l'homme, l'élaboration d'un véritable dialogue homme-machine permettra sans doute une certaine humanisation du travail.

Le traitement de la parole concerne essentiellement les aspects suivants:

- la synthèse: production de la parole par une machine. Elle peut être réalisée de diverses façons, par exemple en restituant des enregistrements élémentaires concaténés entre eux, ou encore en reconstituant la parole grâce à la donnée de ses paramètres caractéristiques.

- la reconnaissance: accès à une machine à l'aide de la voix,

- la compression, le stockage et la transmission de la parole,

- l'identification et la vérification de locuteurs.

Par la suite, nous allons uniquement nous intéresser au problème de la reconnaissance.

On constate que la synthèse aussi bien que la reconnaissance ont de très nombreux débouchés industriels et International Resource Development Inc. (IRD) prévoyait (en octobre 1982) qu'en 1992, les ventes sur le sol américain seulement avoisineraient quatre milliards de dollars dont 60% pour les systèmes de reconnaissance.

Parmi les nombreuses applications de la synthèse et de la reconnaissance de la parole, on peut citer :

- la bureautique (en particulier tout ce qui concerne la dictée automatique [CHARPILLET 85], la saisie de données.....).

- les applications télématiques (l'interrogation de bases de données par exemple, la messagerie vocale,...).

- les automatismes industriels (et notamment la conduite de processus, la détection de pannes...).

- l'ingéniérie médicale (la rééducation d'enfants sourds avec le système SIRENE [HATON 85] , les voitures à commande vocale pour handicapés, les lits pour hémiplégiques,....).

- les transports (la parole est ce que l'on appelle en avionique la troisième main).

- les banques (reconnaissance de codes chiffrés,identification de locuteur pour la signature bancaire [Mollier 84]...).

- le grand public (les voitures "parlent" et une récente étude a montré que les automobilistes sont beaucoup plus attentifs à un indicateur sonore qu'à un témoin visuel, l'EAO ([Benani 84]) et notamment l'enseignement des langues....).

Dans un système de reconnaissance de la parole, deux objectifs sont à envisager:

- ou bien on ne s'intéresse qu'à des mots (commandes vocales simples),

- ou bien on cherche à reconnaitre des suites structurées de mots ou phrases.

Dans le premier cas, il faut encore distinguer deux éventualités :

- compréhension de mots prononcés isolément, c'est à dire séparés par des pauses artificielles.

- recherche de mots clés dans un continuum de parole (parole naturelle sans pauses artificielles).

Dans le deuxième cas (compréhension de la parole continue), une distinction s'impose entre la reconnaissance de phrases extraites d'un langage artificiel ou d'un très petit sous ensemble d'un langage naturel, et la reconnaissance de phrases utilisant la langue naturelle, un vocabulaire important (plusieurs milliers de mots) et une syntaxe très riche.

Pour chacune de ces deux grandes classes d'objectifs (mots isolés et mots enchainés), on peut exiger en outre le caractère multilocuteur du système de reconnaissance, afin d'autoriser l'accès de la machine à toute personne, connue ou inconnue. On doit alors s'affranchir des variantes interlocuteur. Il convient de distinguer les machines typiquement

monolocuteur pour lesquelles l'adaptation à tout nouveau locuteur passe obligatoirement par une phase préalable d'apprentissage individuel (machine que l'on peut rendre artificiellement multilocuteur en multipliant les apprentissages) des machines naturellement multilocuteur. Les machines les plus courantes sont du premier type avec des nuances sur le caractère obligatoire de l'apprentissage individuel.

Le but ultime de la reconnaissance est la possibilité d'utiliser un vocabulaire illimité, sans contraintes particulières ni sur le domaine d'application du discours ni sur l'élocution de l'utilisateur. Le traitement complet de cet objectif serait, pour l'instant, voué à l'échec. C'est pourquoi les systèmes développés jusqu'à présent ne traitent que des sous ensembles abordables du dialogue oral. Les simplifications qui en découlent imposent des contraintes supplémentaires au locuteur qui, par exemple, ne pourra utiliser qu'un nombre restreint de mots, devra prononcer les mots isolément ou devra passer par une phase préalable d'apprentissage avant d'utiliser le système de reconnaissance.

Pour mener à bien une tâche de reconnaissance, il est nécessaire d'intégrer dans un même système un ensemble de sources de connaissances assez diverses:

- connaissances acoustiques: il est nécessaire de connaitre les caractéristiques du signal de la parole afin d'en réduire le plus possible le débit sans perdre l'information sémantique (problème du codage de la parole). Un autre rôle du niveau acoustique est de détecter ce qui est de la parole afin de la séparer du bruit;

- connaissances phonétiques: il faut modéliser les différents phonèmes, leurs invariants, et les décrire en termes de paramètres acoustiques qu'il faut savoir détecter sur le signal lui même;

- connaissances phonologiques: il s'agit de prendre en compte les différentes variantes de prononciation aux frontières de mots ou à l'intérieur d'un même mot dans un contexte bien précis;

- connaissances prosodiques: ce niveau définit les phénomènes de variation du rythme, de l'intensité et de la mélodie de la voix. Il fournit aussi bien des renseignements sémantiques (accent sur les mots importants) que syntaxiques (phrase interrogative par exemple) ou même phonétiques (localisation des phonèmes bien "articulés" et des phonèmes "mal articulés");

- connaissances lexicales: il s'agit de représenter les mots propres à une application donnée en précisant leurs caratéristiques (phonétiques, phonologiques, syntaxiques, sémantiques....);

- connaissances syntaxiques: elles rendent compte de la structure du langage. Il faut à l'aide de règles précises (grammaire) pouvoir détecter les suites de mots qui sont interdites;

- connaissances sémantiques: elles définissent le sens des mots et leurs fonctions d'interprétation afin de déterminer l'action à effectuer pour poursuivre le dialogue;

- connaissances pragmatiques, propres au domaine d'application. Elles sont importantes pour la mise en oeuvre du dialogue.

Ces sources de connaissance ont pour objet de limiter la recherche combinatoire des solutions pouvant interpréter l'énoncé oral analysé. La connaissance partielle que l'on a des différents niveaux de traitement ne permet pas de définir de manière exhaustive les énoncés appartenant à un langage aussi général qu'une langue naturelle. De ce fait, les systèmes de reconnaissance sont limités à des domaines d'application restreints.

La première idée raisonnable est de penser qu'il vaut mieux ne pas utiliser des niveaux de connaissances qui ne sont pas bien maîtrisés.

Une première stratégie consiste donc à ne pas rechercher le contenu phonétique de la parole puisque le niveau de connaissances phonétiques est encore insuffisant, mais seulement son contenu lexical. Ceci conduit à la reconnaissance par mots ou reconnaissance globale. L'unité de décision est l'entité lexicale, qui est le plus souvent un mot, mais qui

peut être un groupe de mots, une phrase complète, un sifflement ou même un bruit. Le processus de reconnaissance globale ne tient aucun compte de la structure et des propriétés phonétiques des mots, seule la forme acoustique est considérée.

Au contraire, si l'on base le processus de reconnaissance sur une connaissance phonétique, on recherche cette fois le contenu fin de la parole (les phonèmes) et l'on emploie une reconnaissance par phonèmes. Une telle reconnaissance est dite analytique exprimant ainsi que l'on recherche la structure fine du langage. Elle consiste à représenter chaque mot par une séquence d'unités phonétiques. Les unités sont dans ce cas décrites une fois pour toutes. Le processus est réalisé en trois étapes principales:

- extraction de traits phonétiques.

- segmentation du signal.

- identification des segments à des phonèmes.

A partir d'un treillis de phonèmes, il est possible de reconnaitre les mots à l'aide des transcriptions phonétiques théoriques contenues dans le lexique. Des règles phonologiques permettent de préciser les variations (par exemple, insertion substitution et élision de phones) que peut subir la transcription phonétique théorique d'un mot. D'après l'intonation, la prosodie différencie les cas litigieux. Cette description présente trois avantages principaux par rapport à la méthode globale:

- elle est moins dépendante du locuteur car les représentations internes des phonèmes se veulent les plus indépendantes possibles des locuteurs. Il est possible de prendre en compte automatiquement les différentes variantes de prononciation grâce à des règles de phonologie.

- la place mémoire nécessaire au stockage du lexique est bien plus faible.

- cette approche permet théoriquement de traiter des vocabulaires quasiment illimités puisque le nombre de phonèmes à reconnaitre est toujours le même (gain en place mémoire et en temps de reconnaissance).

  En pratique, les résultats sont encore médiocres.

  Au chapitre des inconvénients, on peut citer:

- la difficulté d'exporter de tels systèmes puisque la connaissance de la langue est indispensable.

- le caractère heuristique des algorithmes employés rend la mise au point délicate.

- le problème de la segmentation en unités phonétiques. Le décodage acoustico-phonétique est une étape encore mal résolue. Les taux de reconnaissance (environ 65% [Lazrek 83] [Fohr 86]) sont encore insuffisants puisque les expériences ont montré qu'en dessous de 85%, la reconnaissance de phrases était difficile [Liénard 77].

- le câblage des algorithmes mis en oeuvre n'est pas encore envisageable actuellement pour des applications grand public.

En reconnaissance globale, on compare globalement les mots du dictionnaire, considérés comme des entités indivisibles, au signal vocal par l'intermédiaire de paramètres résultant de l'analyse. Les résultats obtenus par cette méthode sont actuellement bien supérieurs à ceux de la reconnaissance analytique. De plus, de tels systèmes présentent les avantages suivants:

- l'exportabilité (ils sont indépendants de la structure phonétique de la langue).

- la simplicité conceptuelle du système de reconnaissance.

  Par contre, on peut citer les inconvénients suivants:

- l'adaptation à un nouveau locuteur se fait plus difficilement.

- une certaine lenteur des algorithmes dès que la taille du vocabulaire devient trop importante.

- la limitation de la taille du vocabulaire.

C'est dans cette dernière approche de la reconnaissance que se situe le cadre de notre travail: application des méthodes de programmation dynamique et de quantification vectorielle à la reconnaissance des mots isolés et des mots enchainés.

De nombreux chercheurs utilisent la programmation dynamique en reconnaissance de mots isolés et leurs travaux montrent que cette technique constitue une solution efficace au recalage temporel des formes comparées.

Ces dernières années, l'algorithme de reconnaissance des mots isolés par programmation dynamique a été généralisé à la reconnaissance de mots enchainés. Nous allons étudier l'algorithme proposé par Bridle et Nakagawa avec et sans contraintes syntaxiques. Nous autorisons également le relâchement des contraintes aux frontières assujettissant les chemins de recalage temporels afin de compenser certaines distorsions dues par exemple à une mauvaise détection parole non parole. Nous proposons un algorithme de reconnaissance de mots enchainés par programmation dynamique qui considère les effets de coarticulation ou de liaison qui apparaissent éventuellement à la frontière des mots prononcés sans pause intermédiaire et qui sont en général négligés dans les algorithmes de reconnaissance globale de la parole continue.

La quantification vectorielle permet de réduire de façon significative la place mémoire occupée par les données nécessaires à la reconnaissance (une ou plusieurs références par mot du vocabulaire dans notre cas). Elle va constituer une étape préliminaire à la reconnaissance et nous proposons quelques algorithmes de reconnaissance des mots isolés utilisant à la fois la quantification vectorielle et la programmation dynamique. Nous montrons que la réduction de la place mémoire occupée ne s'effectue pas au détriment des performances de la reconnaissance. Nous

généralisons enfin l'emploi de la quantification vectorielle à la reconnaissance de mots enchainés.

PARTIE A

PRINCIPE D'UN SYSTEME DE RECONNAISSANCE GLOBALE

INTRODUCTION

([Levinson 84], [Markel 78], [Guibert 79], [Flanagan 72])

Selon la définition du dictionnaire Robert, la parole est "la faculté de communiquer la pensée par un système de sons articulés émis par les organes de phonation".

L'appareil phonatoire peut être assimilé à un instrument à vent dans lequel on peut distinguer:

- une source d'énergie provenant des muscles thoraciques et abdominaux semblable à une soufflerie d'orgue.

- un système d'excitation servant à mettre le flot d'air en vibration (glotte et cordes vocales).

- un système de résonateurs: en série(pavillon laryngo-buccal), en dérivation (conduit nasal).

L'excitation du système phonatoire dépend étroitement des sons émis. Le flux d'air à travers le conduit vocal produit un son de trois manières différentes:

- les cordes vocales vibrent. Quand elles sont pressées l'une contre l'autre, les cordes vocales empêchent le passage de l'air et la pression augmente. Cette pression contraint alors les cordes vocales à s'écarter mais la vitesse de l'écoulement fait que la pression diminue (loi de Bernouilli) et les membranes se rejoignent à nouveau: la pression augmente à nouveau et la vitesse à laquelle se répète ce cycle détermine la fréquence fondamentale du son (pitch [Hess 83], [Gong 86]). Les sons émis alors sont dits voisés.

- un son peut également être émis en réduisant la dimension du conduit vocal en des points de constriction afin de provoquer une turbulence; en forçant l'air à passer dans une petite ouverture entre les dents de la mâchoire supérieure et la lèvre inférieure, on provoque un flux turbulent entendu comme un son "f". Contrairement aux

sons périodiques produits par la vibration des cordes vocales, les sons engendrés par des flux turbulents sont apériodiques: ce sont des bruits, phénomènes aléatoires.

On peut émettre simultanément des sons périodiques et apériodiques: en combinant une vibration des cordes vocales et le bruit d'un "f", on émet un son "v".

- un troisième type d'émission sonore se produit lorsqu'on libère brutalement la pression accumulée derrière un obstacle. Les explosions d'énergie acoustique ont lieu par exemple lorsqu'on prononce "p", "t", "k".

Les sons émis de ces trois façons sont acoustiquement caractérisés par la forme du conduit vocal qui dépend des positions du larynx, de la langue, des lèvres, du palais. Par exemple, si les cordes vocales mettaient directement l'air extérieur en mouvement, sans passer par la gorge, la bouche, le nez, on entendrait un bruit ressemblant plus à une sonnette qu'à une voix; en fait, cette vibration est considérablement altérée par le passage par la gorge, la bouche et les cavités nasales.

Il faut ajouter également que la hauteur tonale est contrôlée grâce à l'audition du son émis. Ainsi, phonation et audition se trouvent liées par une boucle de réaction.

Dans la production de la parole, l'oreille a donc aussi "son mot à dire", puisque la vibration des cordes vocales est commandée par une boucle d'asservissement exploitant le retour auditif de l'oreille.

La parole se caractérise tout d'abord par la grande variabilité des formes vocales: des différences considérables apparaissent entre plusieurs locuteurs du fait du sexe, de l'âge, de l'accent régional..... En effet, le conduit vocal présente des différences physiologiques d'un locuteur à l'autre.

De plus, pour un même locuteur, la prononciation varie suivant l'heure de la journée, l'état émotionnel et l'intensité de la voix: l'élocution d'une phrase ou d'un mot n'est jamais purement répétitive.

Pour un même mot, il suffit de regarder les spectrogrammes représentés figure 1 pour constater ces différences. Les spectres de deux mots distincts bien qu'acoustiquement similaires peuvent présenter plus de ressemblances que les spectrogrammes d'un même mot prononcé par divers locuteurs dans des conditions différentes. Il faut que le système de reconnaissance soit capable de déceler les différences spectrales significatives (lorsqu'elles existent) et d'ignorer les autres.

Pour la reconnaissance automatique de la parole, il semble donc exclu d'établir une correspondance biunivoque entre les formes vocales à étudier et une simple "bibliothèque" de formes enregistrées une fois pour toutes.

Un système de reconnaissance vocale devra comporter une description complète de chaque mot susceptible d'être prononcé. La reconnaissance effectue alors le calcul d'un taux de similitude entre le signal vocal et les éléments de description connus du système.

Au contraire de l'écriture, la parole est un phénomène essentiellement continu, sans frontières apparentes entre les mots. Les silences qui apparaissent correspondent en général à des pauses de respiration dont l'occurence est aléatoire. De ce fait, la localisation des mots dans le signal vocal est une opération difficile et indéterministe, nécessitant le recours à des connaissances linguistiques. Par exemple, oralement, il est presque impossible de distinguer les deux phrases: "l'homme mettait les gants" et "l'homme est élégant".

Une autre caractéristique importante du signal vocal est son extrême redondance. D'après la théorie de l'information définie par Shannon, la transmission de la parole dans de bonnes conditions nécessite un débit d'informations d'au moins 500000 bits/s. Or un locuteur moyen émet environ dix phonèmes par seconde, un phonème pouvant être codé sur cinq bits, il suffit de 50 bits/s environ pour transmettre un message parlé sous la forme de sa transcription phonétique.

Il est intéressant de mettre à profit cette redondance de la parole de façon à diminuer les volumes de données et donc les temps de

traitement informatique: la paramétrisation de la parole, qui fera l'objet du deuxième chapitre de cette partie, consiste à extraire du signal vocal un nombre de paramètres pertinents, tout en éliminant l'information redondante. Il n'existe cependant pas de solution mathématique générale à ce problème et il est impossible de séparer exactement information utile et information redondante de la parole.

Enfin, la syntaxe du langage parlé est généralement moins stricte que celle du langage écrit: les programmes de reconnaissance évolués doivent obligatoirement en tenir compte si l'on veut qu'ils soient utilisables en pratique.

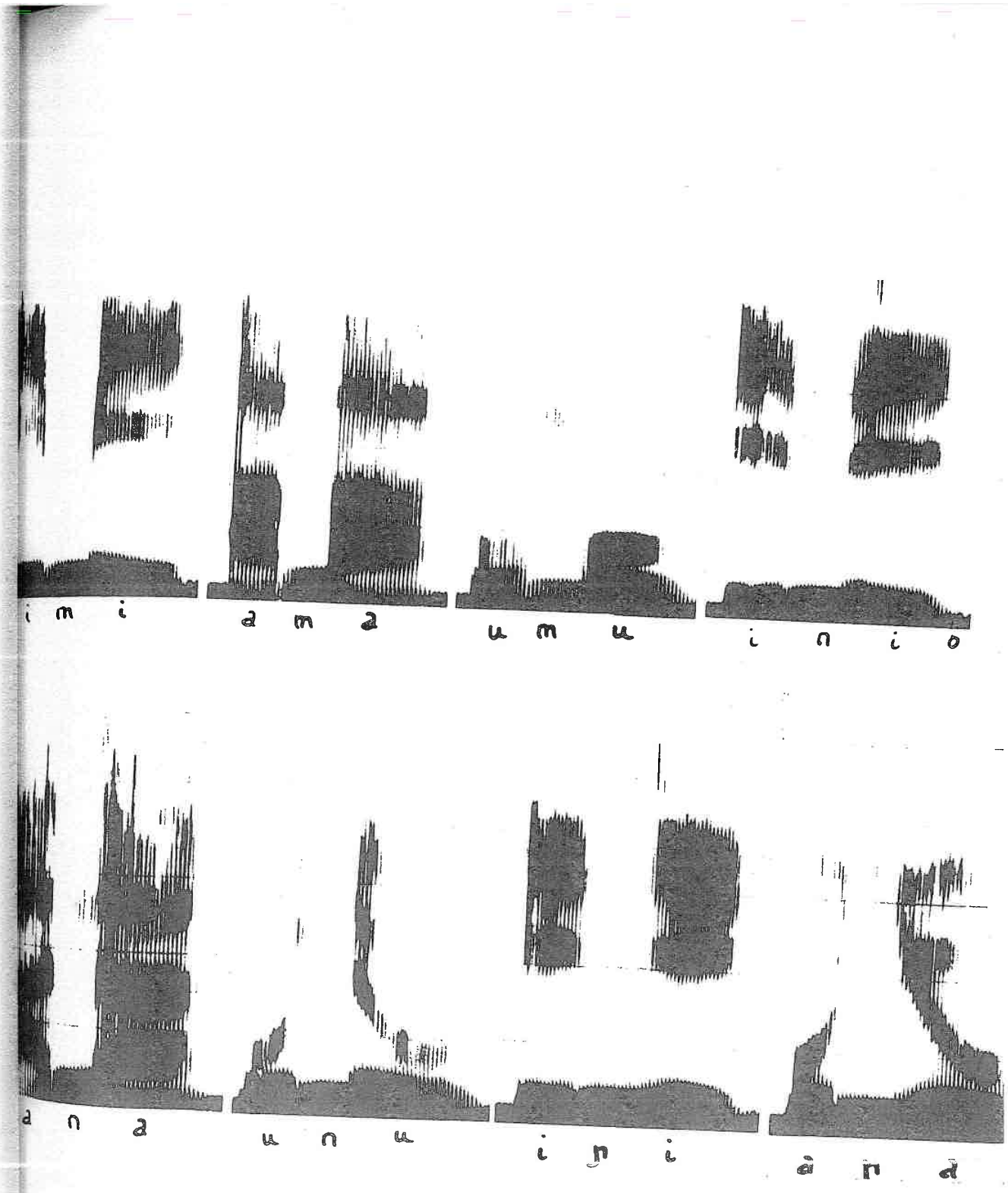


figure 1: exemples de spectrogrammes de parole [Lonchamp].

CHAPITRE 1: DESCRIPTION D'UN SYSTEME DE RECONNAISSANCE GLOBALE

([G.C.P.-SFA 87])

La reconnaissance automatique de la parole rencontre quatre difficultés principales:

- la continuité du signal vocal: il n'y a pas de pauses régulières dans une phrase parlée, pas plus que de marqueurs acoustiques pour indiquer les débuts et les fins des mots. De plus, la prononciation continue entraine des altérations entre mots adjacents,

- la variabilité à la fois inter- et intra-locuteur de la parole: un algorithme de reconnaissance efficace devra donc se baser davantage sur la similitude des formes que sur leur identité,

- l'ambiguité: une même suite de sons peut correspondre à la prononciation de plusieurs mots (par exemple le même son /R/ peut correspondre aux mots air, aire, Eire, ère, erre, haire, hère, R),

- le non-dit: la parole n'est qu'une composante du langage humain, l'intention sous-jacente est parfois plus importante que le message lui même (par exemple, "pouvez-vous me donner l'heure?" attend plutôt "16 heure" comme réponse que "oui, je peux").

Reconnaitre et comprendre la parole constitue donc un problème délicat et complexe.

C'est pourquoi la reconnaissance globale de la parole s'est intéressée à des sous-problèmes de la recherche en parole: elle a d'abord tenté de résoudre le problème de la reconnaissance monolocuteur de mots isolés pour un petit vocabulaire avant d'aborder la reconnaissance multilocuteur de mots enchainés.

A l'origine, l'idée consiste à formuler un certain nombre d'hypothèses simplificatrices afin d'élaborer un système capable de reconnaitre un petit vocabulaire prononcé mot à mot par un seul locuteur. Lors d'une phase d'apprentissage, le locuteur prononce chacun des mots du vocabulaire qu'il souhaite utiliser. L'image acoustique des

différents mots est conservée en mémoire. Lors de la reconnaissance, le système compare l'image acoustique du mot inconnu avec toutes les images acoustiques des mots de référence et choisit la meilleure sur un critère du type "plus proche voisin". C'est sur ce principe que fut réalisé en 1972 par la société Threshold Technology Inc. le premier système commercial de reconnaissance de la parole, le VIP100.

Par la suite, de nombreux progrès ont été accomplis qui ont permis d'améliorer les performances de la reconnaissance.

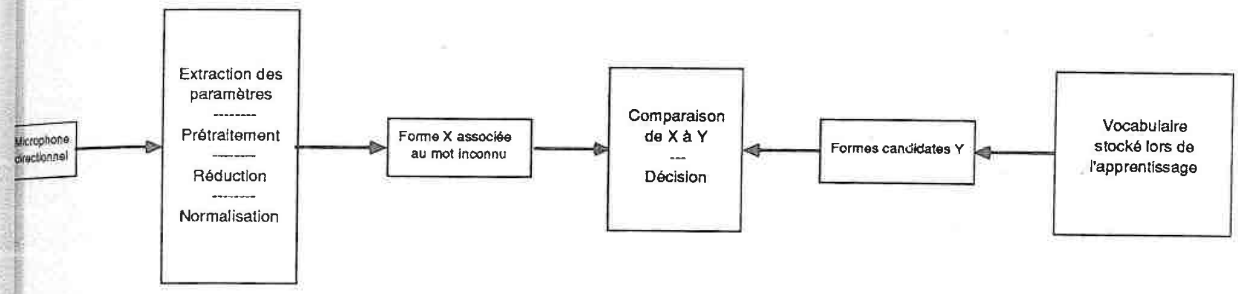
En premier lieu, l'algorithme de comparaison entre le mot inconnu et les mots de référence a été modifié pour mieux prendre en compte les variantes de prononciation d'un même mot. Les principes de la programmation dynamique, d'abord utilisé en recherche opérationnelle, ont été adaptés au problème de la reconnaissance de la parole par les chercheurs soviétiques (Vintsjuk) puis par les Japonais (Sakoe, Chiba).

Il a ensuite été généralisé à la reconnaissance de mots enchainés, c'est à dire à la reconnaissance de mots prononcés sans pauses intermédiaires de silence. Une variante de cet algorithme a également permis la détection de mots-clefs dans un continuum de parole. Sur ces bases fut commercialisé en 1979 par la société japonaise NEC le DP100 qui, le premier, permettait la reconnaissance de mots enchainés.

Une autre amélioration apportée a été d'essayer de rendre multilocuteur la reconnaissance de mots isolés. Une première solution a été d'enregistrer plusieurs locuteurs prononçant le même vocabulaire et d'utiliser un algorithme de classification automatique pour définir une forme de référence par classe de prononciation. En 1979, la société américaine Dialog a commercialisé le M1800 capable de reconnaitre un vocabulaire de dix chiffres par le canal téléphonique pour tout locuteur anglophone.

Un système de reconnaissance globale de la parole admet donc en entrée le signal acoustique correspondant au mot inconnu et fournit en sortie le mot qu'il a reconnu.

Le schéma général d'un système de reconnaissance est le suivant:



Le dictionnaire des formes R1,...,Rn est constitué lors de la phase d'apprentissage par les images acoustiques des différents mots du vocabulaire de l'application.

Un système de reconnaissance comporte donc deux phases principales:

- l'apprentissage qui permet de constituer le dictionnaire des formes de référence,
- la reconnaissance qui détermine quel est le mot qui a été prononcé.

La phase de reconnaissance comprend elle-même deux étapes:

- l'analyse du mot inconnu qui permet d'obtenir son image acoustique,

- La comparaison de cette image à toutes les formes de référence qui aboutit à la prise de décision.

Nous verrons dans le chapitre suivant quelles sont les principales méthodes pour réaliser l'apprentissage.

Le chapitre 3 donnera un aperçu des techniques utilisées en analyse de la parole.

La comparaison entre forme inconnue et formes de référence est l'objet de notre étude et les parties B, C, D lui seront consacrées.

CHAPITRE 2: QUELQUES METHODES D'APPRENTISSAGE

([Gauvain 85])

Le rôle de l'apprentissage est de constituer le dictionnaire des formes de référence de l'application. C'est une étape fondamentale dans tous les systèmes de reconnaissance des formes car le soin apporté à cette opération conditionne les performances du système. En effet, la puissance des techniques utilisées lors de la reconnaissance ne peut compenser un apprentissage incorrect ou insuffisant.

Nous allons distinguer apprentissage monolocuteur et apprentissage multilocuteur car certaines méthodes utilisées dans le cas d'un unique locuteur ne peuvent se généraliser au cas multilocuteur.

De nombreuses techniques d'apprentissage monolocuteur ont été envisagées, parmi celles-ci nous pouvons citer:

- dans le cas d'un locuteur entrainé, qui sait éviter les bruits parasites avant ou après une prononciation (par exemple, un râclement de gorge, une respiration,...), une ou deux élocutions de chaque mot peuvent suffire,

- le locuteur prononce n fois chaque mot du vocabulaire et chacune de ces répétitions constitue une forme de référence. Cette technique est assez efficace mais elle est relativement coûteuse en place mémoire puisqu'à chaque mot du dictionnaire correspondent n formes de référence,

- à partir de ces n répétitions de chaque mot, une unique forme de référence est formée en prenant la moyenne de ces prononciations. Cette méthode est mal adaptée si les locutions du même mot sont assez différentes car elle génère une forme qui ne correspond à aucune occurence du mot. De plus, elle nécessite que les mots soient ajustés sur une même échelle temporelle par un algorithme de programmation dynamique,

- le locuteur répète n fois chaque mot du vocabulaire et un algorithme de classification automatique détermine une forme de référence par classe de prononciation,

- le locuteur prononce la liste des mots du vocabulaire jusqu'à obtenir deux occurences d'un même mot suffisamment proches. Elles sont alors moyennées après alignement temporel pour constituer une forme de référence,

- lorsque les mots sont acoustiquement très voisins, les différences entre les zones discriminantes auront une influence faible sur le taux de dissemblance. Une solution est alors de construire un réseau dans lequel tous les segments discriminants correspondent à des chemins différents alors que les régions non discriminantes donneront lieu à des chemins communs ([Moore 83]),

- le locuteur énonce n fois chaque mot. A partir de ces n répétitions, les densités de probabilité d'une source de Markov modélisant le mot sont évaluées par un algorithme itératif.

Certaines de ces méthodes peuvent s'appliquer au cas d'un apprentissage multilocuteur. Par exemple, les apprentissages par classification automatique ou par modèle de Markov fournissent également une solution au problème de la détermination de références multilocuteur à condition que les répétitions d'un mot soient obtenues pour un nombre suffisant de locuteurs.

Une autre approche consiste à déterminer une transformation qui permet à un nouveau locuteur d'être reconnu à partir du vocabulaire enregistré par un locuteur de référence.

Toutes ces méthodes permettent de constituer le dictionnaire des formes de référence qui seront utilisées lors de la comparaison.

Bien déterminer ces formes est une étape importante qui va influer sur les performances de la reconnaissance.

CHAPITRE 3: ANALYSE DE LA PAROLE

Les données fournies à l'ordinateur par le système d'acquisition de la parole représentent une quantité importante d'informations: par exemple, la représentation d'une seconde de parole, après échantillonnage à 12 Khz effectué sur 10 bits, nécessite 120000 bits.

La quantité de données à traiter dépasse les capacités de traitement et de stockage d'un ordinateur de gamme moyenne. L'une des étapes importantes de la reconnaissance sera donc la réduction du signal vocal en extrayant les paramètres les plus pertinents: on désigne généralement ce traitement sous le nom d'analyse acoustique du signal([Rabiner 78], [Markel 76], [Levinson 84]).

Les méthodes peuvent être classées en deux grandes catégories:

- les méthodes de type temporel: il s'agit d'extraire les informations du signal temporel issu du microphone. Généralement, cette méthode consiste à compter le nombre de passages par zéro du signal ou de ses dérivées successives dans une fenêtre temporelle, ou à mesurer l'emplacement et l'amplitude des extrema;

- les méthodes de type fréquentiel: on considère que le signal se compose d'une somme de sinusoides ou d'exponentielles complexes (théorème de superposition de Fourier). Les représentations de Fourier sont traditionnellement utilisées pour deux raisons principales: d'une part, il est facile de déterminer la réponse d'un système linéaire à une superposition de sinusoides ou d'exponentielles complexes. Le système de production de la parole peut être modélisé par un système linéaire. D'autre part, certaines propriétés qu'il est difficile d'observer dans le domaine temporel apparaissent plus nettement dans le domaine fréquentiel. Par exemple, les formants se déterminent aisément dans le domaine fréquentiel alors qu'il est quasiment impossible de les distinguer dans le domaine temporel.

Parmi ces méthodes, nous pouvons citer les méthodes cepstre ([Bogert 63]), FFT, banc de filtres ([Rabiner 71]), MFCC ([Davis 80])...

Une méthode très efficace est celle de la prédiction linéaire (Linear Prediction Coding) qui considère qu'un échantillon de parole peut être prédit à partir d'une pondération linéaire d'un nombre fini d'échantillons précédents. Cette hypothèse se justifie par le fait que la forme du conduit vocal n'évolue pas rapidement. Il s'agit à l'origine d'une méthode temporelle, mais de nombreux paramètres peuvent également être déduits dans le domaine fréquentiel.

Signalons également l'existence des modèles d'audition qui sont directement issus de recherches menées en physiologie et en psychoacoustique ([Caelen 79]).

D'après Klatt ([Klatt 80]), un modèle de système auditif périphérique doit comporter au minimum les étages suivants:

- un filtre de préemphase avec adaptation pour modéliser les courbes d'isotonie,

- un banc de filtres espacés le long d'une échelle Mel ou Bark,

- des redresseurs simple alternance et des filtres passe-bas de faible constante de temps avec contrôle de gain automatique pour modéliser les phénomènes de distorsion et de saturation dans les cellules ciliées et les fibres nerveuses,

- des circuits d'inhibition latérale et une adaptation temporelle pour rehausser les variations spatio-temporelles des canaux de sortie de la cochlée.

Nous allons passer rapidement en revue les principales méthodes utilisées en reconnaissance de la parole.

1. ANALYSE LPC

([Saito 66])

La prédiction linéaire permet d'éliminer une part importante de la redondance du signal de parole. En effet, cette redondance se traduit par l'arrivée d'éléments qui ne fournissent pas d'information nouvelle; savoir prédire la valeur du signal de parole à un instant t en fonction de ses valeurs passées permet de débarrasser l'ordinateur de ces pseudo-informations. Bien évidemment, cette redondance n'est que partielle et les coefficients de prédiction linéaire devront être réajustés régulièrement.

Cette méthode fait intervenir un modèle de production de la parole (figure 1). Le conduit vocal, qui filtre le signal d'excitation, peut être assimilé à un filtre récursif. Avec une bonne approximation, le signal qu'il émet à un instant t peut être calculé à l'aide des valeurs qu'il a prises aux p instants précédents. Supposons que T soit la période d'échantillonnage. Le signal s(nT) pourra donc s'exprimer comme la combinaison linéaire de son passé aux p instants précédents et d'une erreur e(n), soit:

$$s(nT) = \sum_{i=1}^{p} a_i \cdot s[(n-i)T] + e(n)$$

où

$$\sum_{i=1}^{p} a_i \cdot s[(n-i)T]$$

est la partie prédite.

Le problème va consister à régler les coefficients de combinaison pour que l'énergie de l'erreur e(n), $\sum_{n=n_0}^{n=n_1} e(n)^2$, dans un intervalle de temps $[n_0T, n_1T]$ soit minimale.

Deux méthodes se distinguent suivant la façon de définir l'intervalle de sommation $[n_0T, n_1T]$:

- méthode d'autocorrélation: l'intervalle de temps va de moins l'infini à plus l'infini et on considère le signal nul hors de l'intervalle [0,N] où N est le nombre de points considérés.

- méthode de covariance: cette fois, l'intervalle de temps ne varie plus de moins l'infini à plus l'infini mais est limité à [0,N-1].

2. ANALYSE CEPSTRALE

Soient:

- x(nT) le signal de parole échantillonné à la période T,

- h(nT) la réponse impulsionnelle du conduit vocal,

- e(nT) le signal d'excitation.

Le modèle de production de la parole permet d'établir la relation de convolution: $x(nT)=h(nT)*e(nT)$.

La méthode va consister à effectuer dans la fenêtre temporelle d'analyse une déconvolution afin de séparer les contributions du conduit vocal et de la source d'excitation glottale.

L'analyse cepstrale procède de la façon suivante pour transformer le produit de convolution en addition:

- elle opère une transformée de Fourier, le signal à traiter x(nT) dans le domaine temporel devient $X(\omega)$dans le domaine fréquentiel. Une conversion du produit de convolution en multiplication a été alors effectuée;

- elle réalise ensuite un logarithme qui transforme la multiplication en addition;

- elle met en oeuvre des traitements linéaires: une transformée de Fourier inverse qui repasse le signal dans le domaine temporel. Ce signal est le cepstre (anagramme du mot spectre), un cepstre n'est pas une sorte de spectre puisqu'il se situe dans le domaine temporel. Par filtrage, il est alors possible de séparer les paramètres d'excitation de ceux du conduit vocal.

Un inconvénient majeur de ce traitement est justement cette étape de séparation pour les voix des femmes qui ont une fréquence élevée et un conduit vocal long.

L'analyse cepstrale est relativement lourde: elle nécessite deux transformées de Fourier et un calcul de logarithme.

3. BANC DE FILTRES

Le signal de parole issu d'un microphone est analysé grâce à un banc de filtres passe bande contigus couvrant une étendue spectrale intéressante de la voix (de 200 à 6000 Hz en général). Le signal délivré par chacun de ces filtres est d'abord redressé avant de traverser un filtre passe bas qui ne laisse passer que les variations "lentes" du signal. Le résultat obtenu représente l'évolution au cours de l'élocution de la densité spectrale d'énergie relative à la bande passante du filtre d'analyse considéré.

Un banc de filtres peut être réalisé par une transformée de Fourier ou par des filtres analogiques ou numériques. Il se caractérise par le nombre de filtres, la distribution de leur fréquence centrale (il existe différentes échelles comme l'échelle de Bark, de Mel, ...) et la caractéristique du filtre passe bas à la sortie du redresseur.

Les sorties (Y1,....,Yn) à un instant donné constituent un vecteur que nous appellerons prélèvement.

L'acquisition de notre corpus a été réalisée sur un 68000 Motorola à une fréquence d'échantillonnage de 12 Khz sur 10 bits.

Le banc de filtres numériques que nous avons utilisé ([Dumas 85]) pour traiter le signal ainsi échantillonné possède 16 canaux couvrant la plage de fréquence 300-6000 Hz. Pour déterminer les bandes de fréquence, nous avons imposé une échelle logarithmique, les différentes plages sont indiquées en annexe 1. La fenêtre d'analyse utilisée est de 256 points de signal, c'est l'intervalle sur lequel sera calculé un prélèvement. La plage de recouvrement est de 100 points (la fenêtre n+1 est superposée à la fenêtre n sur cette longueur).

Les valeurs d'énergie sont normalisées afin de s'affranchir des variations de volume de la voix. La normalisation s'effectue bien entendu une fois que tout le signal de parole a été traité.

4. MFCC : MEL-FREQUENCY CEPSTRUM COEFFICIENTS

Ces coefficients cepstraux résultent d'une transformée de Fourier du logarithme du spectre à court terme du signal acoustique.

Cela signifie qu'après acquisition, le signal acoustique passe d'abord dans un banc de filtres dont les fréquences centrales sont réparties suivant une échelle Mel: les filtres sont linéairement espacés aux basses fréquences et suivant une échelle logarithmique aux hautes fréquences. La répartition suivant l'échelle de Mel des filtres présente l'avantage de respecter les propriétés de l'audition.

Les coefficients qui viennent d'être déterminés subissent une transformation logarithmique.

Les coefficients cepstraux sont alors déterminés en calculant la transformée de Fourier des composantes des vecteurs obtenus

précédemment.

Signalons enfin l'existence de LFCC (lineary-frequency cepstrum coefficients) qui suit le même principe que l'analyse MFCC à ceci près que les fréquences centrales des filtres sont réparties linéairement et non plus suivant une échelle Mel.

CONCLUSION

Nous venons de voir quelques méthodes d'analyse du signal de parole.

Pratiquement, les paramètres sont évalués sur des segments de parole et un mot M est représenté par une suite de vecteurs m_i dont les coordonnées $m_{i,p}$ sont les paramètres retenus:

$$M = (m_1, \ldots, m_I)$$
$$m_i = (m_{i,1}, \ldots, m_{i,p})$$

où I est la longueur du mot et p le nombre de paramètres évalués sur chaque segment.

L'analyse étant terminée, la reconnaissance proprement dite peut se dérouler.

Les distances qui seront utilisées en reconnaissance pour comparer les vecteurs de paramètres sont liées au type de paramètres retenus.

En ce qui concerne l'analyse banc de filtres ou cepstre, le choix se porte en général sur des distances associées aux normes L_n, n étant un coefficient entier pouvant varier de 1 à l'infini:

$$d_n(a,b) = \left(\sum_{k=1}^{p} |a_k - b_k|^n \right)^{\frac{1}{n}}$$

Les distances les plus employées sont:

$$d_1(a,b) = \sum_{k=1}^{p} |a_k - b_k|$$

qui est la distance de Hamming.

$$d_2(a,b) = \left(\sum_{k=1}^{p} |a_k - b_k|^2 \right)^{\frac{1}{2}}$$

qui est la distance euclidienne.

Dans le cas de coefficients LPC, la distance la plus souvent retenue est la distance d'Itakura,définie par:

$$d_I(a,b) = \log\left(\frac{aRa^T}{bRb^T}\right)$$

où a^T est le vecteur colonne transposé de a. Si p est l'ordre du modèle, a et b sont deux vecteurs de (p+1) coefficients de prédiction dont la première composante vaut 1, R est la matrice carrée (p+1,p+1) des coefficients d'autocorrélation évalués sur le segment de signal correspondant à b.

Le dénominateur bRb^T représente l'erreur quadratique de prédiction sur le segment du signal correspondant à b, le numérateur peut être évalué à l'aide de l'expression suivante:

$$aRa^T = r(0)r_a(0)+2\sum_{k=1}^{p} r(k)r_a(k)$$

où

- les r(k) sont les coefficients d'autocorrélation sur la portion de signal correspondant à b (valeur de R),
- les $r_a(k)$ sont les coefficients d'autocorrélation sur les coefficients de prédiction de a.

Remarquons que d_I ne respecte pas la condition de symétrie car $d_I(a,b)$ est différent de $d_I(b,a)$ et n'est donc pas une véritable distance.

Par la suite, puisque nous avons effectué une analyse par banc de filtres, nous utiliserons une distance associée à une norme Ln. Nous avons choisi la distance de Hamming (n=1) car elle réalise un bon compromis entre le temps nécessaire à son calcul et les performances obtenues.

PARTIE B

PROGRAMMATION DYNAMIQUE EN RECONNAISSANCE DE MOTS ISOLES

INTRODUCTION

L'idée de base d'un système de reconnaissance globale de la parole est de donner au système au moins une image acoustique de chacun des mots qu'il devra identifier (on peut considérer un mot comme une classe d'équivalence de bruit: c'est l'ensemble de tous les sons qui représentent, dans le contexte où ils sont émis, la même unité lexicale).

Cette opération est effectuée lors de la phase d'apprentissage où chacun des mots est prononcé une ou plusieurs fois.

Au cours de la phase de reconnaissance, la même analyse acoustique fournit l'image du mot à reconnaitre qui est comparée à toutes les images de référence. Le mot dont la référence acoustique est la plus proche de l'image à reconnaitre est déclaré mot reconnu (règle du plus proche voisin).

La comparaison demande donc une estimation du degré de similitude entre le son étudié et celui représenté par l'empreinte en mémoire. Dans tous les systèmes de reconnaissance de mots, le processus se termine par une étape de décision où l'on fait intervenir une mesure du degré de coincidence.

Les deux principaux problèmes inhérents à ce type de reconnaissance sont:

- l'extraction des paramètres pertinents du signal vocal afin de connaitre les images acoustiques des mots.
- la définition de l'indice de dissemblance qui devra prendre en compte les différences entre les échelles temporelles des images acoustiques à comparer.

En effet, lors de prononciations répétées d'un même mot, des différences apparaissent qui font que les images acoustiques de ce même mot ne sont pas identiques: variation dans le débit d'élocution, dans l'accentuation des diverses parties du mot, etc....

Un locuteur même entrainé ne peut prononcer plusieurs fois une même séquence vocale avec exactement le même rythme et la même durée. Les échelles temporelles de deux occurences d'un même mot ne coincident donc pas et les formes acoustiques issues de l'étage d'analyse ne peuvent être simplement comparées point à point.

On peut distinguer a priori deux sources de modifications de l'échelle temporelle:

- le changement de vitesse d'élocution qui est représentable par une transformation linéaire de l'axe des temps.

- les variations dans le rythme de prononciation qui se traduisent par une transformation non linéaire.

En fait, tout changement de vitesse d'élocution s'accompagne d'une transformation non linéaire de l'échelle temporelle car les parties stables du signal sont plus affectées par ce changement que les transitions.

Pour tenir compte des variations de volume de la voix entre deux productions, il est nécessaire d'effectuer une normalisation en énergie. Les différences énergétiques peuvent être traitées directement par le système d'acquisition ou par une normalisation dans le calcul des distances inter-formes.

Si on compare directement les formes acoustiques, on ne tient pas compte des variations du rythme d'élocution et cela conduit à des erreurs de reconnaissance. La figure 1 montre une telle comparaison entre deux mots anglais "masses" et "mashes". La comparaison des empreintes du mot inconnu (représenté sur l'axe supérieur) et du mot étudié (représenté sur l'axe inférieur) aboutit à une erreur d'identification.

Il convient donc d'utiliser, pour les comparer, un indice de ressemblance ou de dissemblance qui tienne compte de ces légères variations. La programmation dynamique, introduite par Joseph Bellman [Bellman 57] pour résoudre certains problèmes de conception de

servomécanismes, se propose de pallier les distorsions essentiellement non linéaires dues aux variations du rythme d'élocution. La figure 2 montre que cette technique permet de conclure à une décision juste dans la comparaison des deux mots "masses" et "mashes".

A chaque intervalle du mot prononcé, la programmation dynamique tente d'associer un intervalle de l'image acoustique de référence, afin de minimiser une mesure globale de l'écart entre le son prononcé et l'empreinte.

C'est cette méthode que nous allons étudier par la suite.

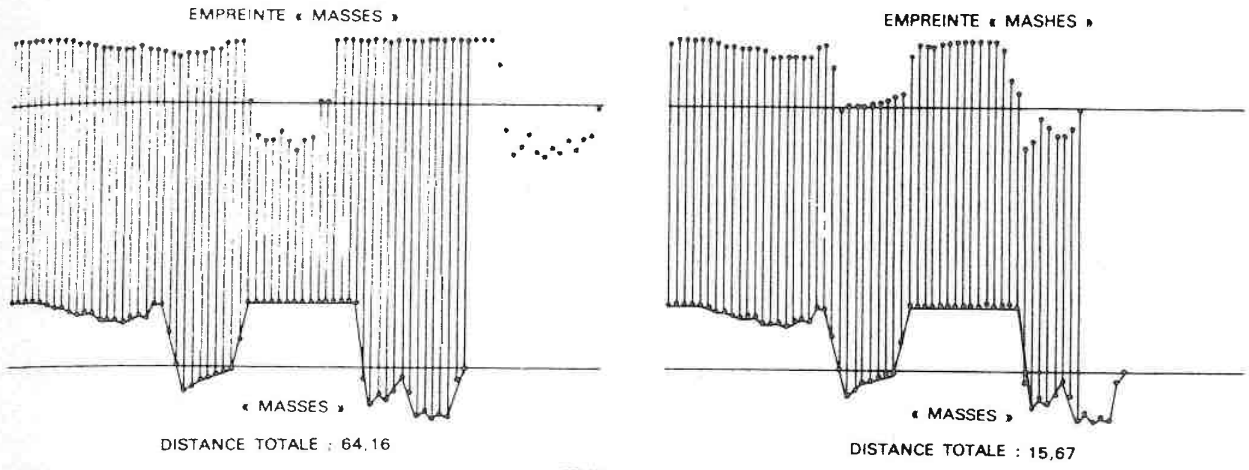


figure 1: comparaison directe [Levinson 83].

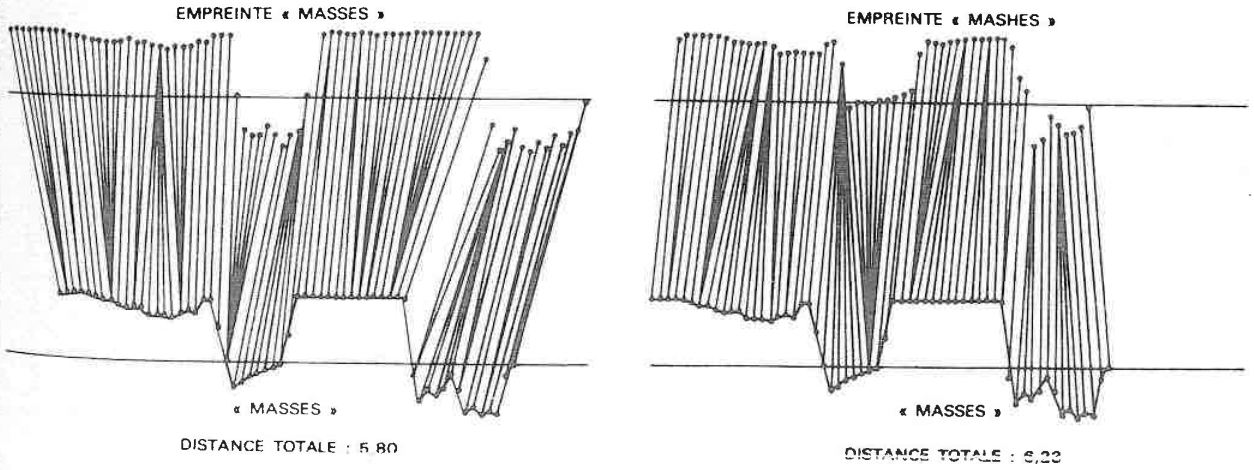


figure 2: comparaison dynamique [Levinson 83].

CHAPITRE 1: TECHNIQUE DE PROGRAMMATION DYNAMIQUE

En reconnaissance globale, il s'agit de comparer la forme inconnue à un ensemble de formes de référence constituant le vocabulaire de l'application.

Les formes vocales qui sont considérées lors de la comparaison ont été au préalable codées par un analyseur acoustique sous la forme d'une suite de vecteurs dont les composantes sont dans notre cas les sorties d'un vocoder à canaux. L'algorithme de comparaison va donc comparer des suites de vecteurs.

La variation de la vitesse d'élocution conduit à l'obtention de formes vocales qui ont des longueurs et des rythmes différents pour un même mot, ces distorsions n'étant pas linéaires. L'algorithme de comparaison devra tenir compte de ces variations et effectuer un "recalage temporel".

1. LE RECALAGE TEMPOREL

1.1. Principe

Le rôle du recalage temporel est de pallier les distorsions dues aux variations du rythme d'élocution.

Il s'agit donc de synchroniser les échelles de temps des deux formes à comparer (figure 1).

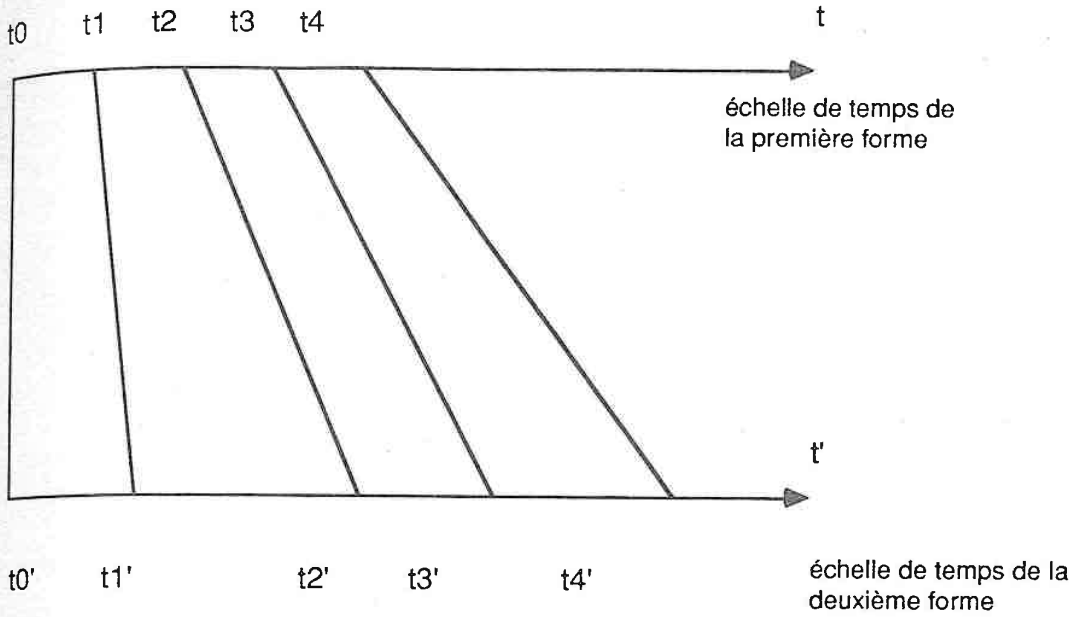


figure 1: principe du recalage temporel.

1.2. Chemin et fonction de recalage

Soient:

- t(1)....t(I) la suite de vecteurs caractérisant la forme T (I représente la longueur en prélèvements de la forme T).
- r(1)....r(J) la suite de vecteurs caractérisant la forme R (J représente la longueur en prélèvements de la forme R).

Le recalage temporel doit donc établir une correspondance qui réalise les meilleures mises en correspondance entre ces suites de vecteurs.

Le problème consiste donc à trouver parmi toutes les fonctions:

$$\{1,\ldots,I+J\} \longrightarrow \{1,\ldots,I\} \times \{1,\ldots,J\}$$
$$k \longrightarrow w(k) = (i(k),j(k))$$

$(w(k) = (i(k),j(k))$ indique que la fonction de recalage met en coincidence le i(k)ème vecteur de la forme T et le j(k)ème vecteur de la forme R), la fonction optimale au sens d'une certaine métrique, $\hat{w}$, qui réalise la correspondance optimale entre les deux formes.

Si l'on porte sur un axe horizontal les différents vecteurs de la forme T (un point représentant un vecteur) et sur un axe vertical les différents vecteurs de la forme R, une représentation de la fonction w est un chemin dans le plan ainsi défini, appelé chemin de recalage. (figure 2)

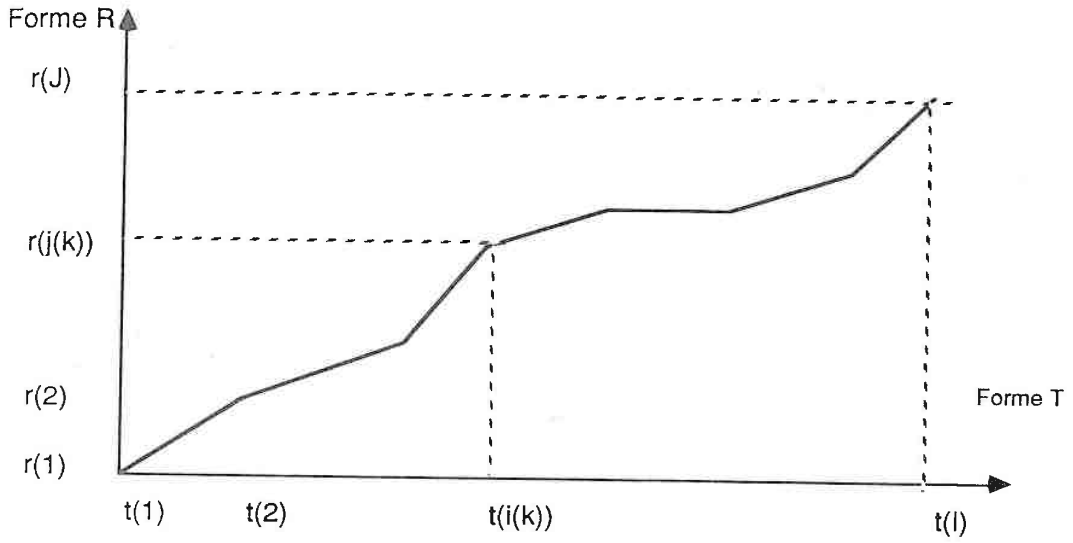


figure 2: exemple de chemin de recalage.

1.3. Contraintes

Il faut que la fonction de recalage respecte l'évolution dans le temps du signal de parole. Elle est donc soumise à une condition de

monotone croissance qui s'exprime par les relations:

$$i(k) \leq i(k+1)$$
$$j(k) \leq j(k+1)$$

De plus, elle ne doit pas donner lieu à des compressions ou des dilatations irréalistes (figure 3).

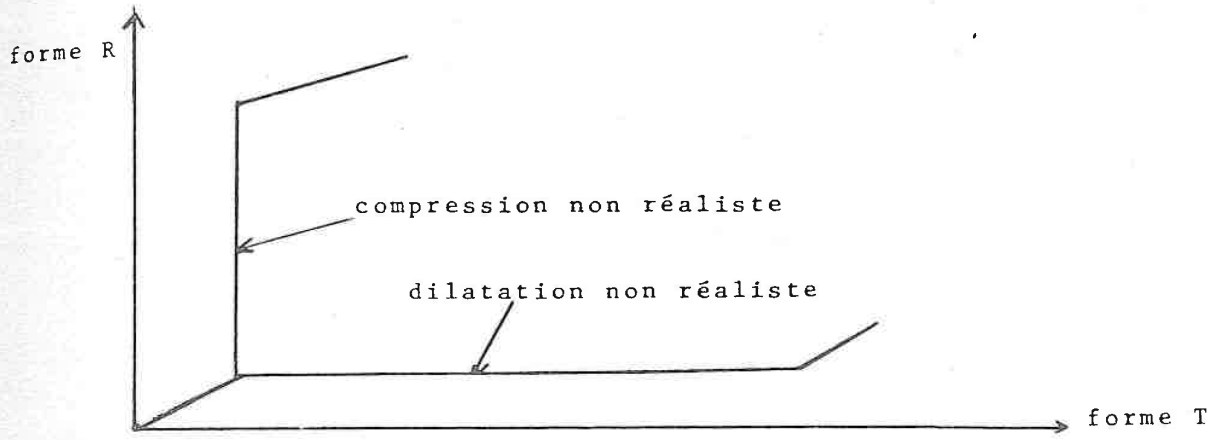


figure 3: exemple de dilatation et de compression irréalistes.

On assujettit donc la fonction de recalage à des contraintes locales qui lui interdisent un certain nombre de déplacements locaux. La figure 4 indique les trois principales contraintes utilisées.

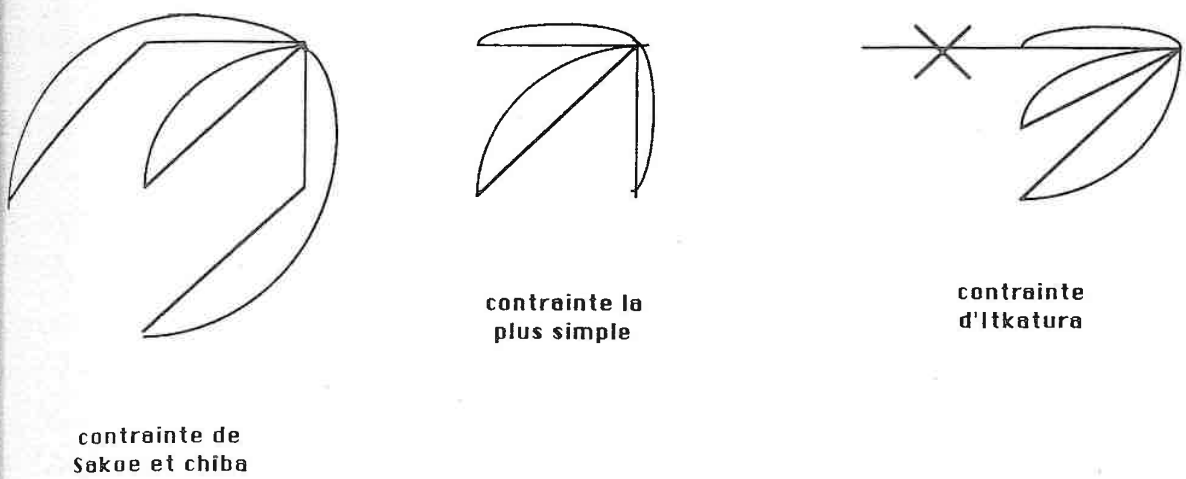


figure 4: exemples de contraintes.

La contrainte sans condition de pente définit simplement les conditions de monotonie: tous les chemins possibles sont pris en considération.

La contrainte de Sakoe et Chiba [Sakoe 78] interdit deux déplacements consécutifs dans la même direction lorsque celle-ci est horizontale ou verticale.

La pente globale des chemins de recalage est comprise entre 0,5 et 2.

La contrainte d'Itakura [Itakura 75] interdit deux déplacements horizontaux consécutifs ainsi que tout déplacement vertical.

Le taux de compression ou de dilatation est, comme pour la contrainte précédente, compris entre 0,5 et 2.

Ainsi, les relations suivantes doivent être vérifiées:

- Si la contrainte retenue est la contrainte de Sakoe et Chiba:

$$i(k) - i(k-1) \leqq 1$$

et

$$j(k) - j(k-1) \leqq 1$$

- Si la contrainte retenue est la contrainte la plus simple sans condition de pente:

$$i(k) - i(k-1) = 1$$
$$j(k) - j(k-1) = 0 \text{ ou } 1$$

ou

$$i(k) - i(k-1) = 0$$
$$j(k) - j(k-1) = 1$$

- Si la contrainte retenue est la contrainte d'Itakura:

$$j(k) - j(k-1) = 0,1,2 \qquad \text{si } j(k-1) \neq j(k-2)$$
$$j(k) - j(k-1) = 1,2 \qquad \text{si } j(k-1) = j(k-2)$$
$$i(k) - i(k-1) = 1$$

De plus, pour que la fonction de recalage prenne en compte l'ensemble des prélèvements des deux formes T et R, on oblige la fonction de recalage à respecter les contraintes aux frontières suivantes:

$$i(1) = 1 \qquad \text{et} \qquad i(K) = I$$
$$j(1) = 1 \qquad\qquad j(K) = J$$

où K est le nombre de coincidences effectuées par la fonction de recalage. Cette dernière contrainte impose à la fonction de recalage de prendre en compte tous les vecteurs des deux formes mises en comparaison.

2. LA PROGRAMMATION DYNAMIQUE

([Vintsyuk 68], [Vintsyuk 71], [Ney 82], [Gauvain 85])

La fonction de recalage optimale est la fonction définie dans le paragraphe précédent qui minimise une certaine métrique. Le problème se ramène à la minimisation d'une fonctionnelle, la fonctionnelle que l'on associe habituellement à une fonction w de recalage est définie par:

$$D_N(w) = \frac{\sum_{k=1}^{K} d(i(k),j(k)) \times P(k)}{N(P)}$$

où:

- K représente le nombre de points de la fonction de recalage.
- d(i(k),j(k)) est la distance locale entre le i(k)ème vecteur de la forme T et le j(k)ème vecteur de la forme R. Par exemple, ce peut être la distance euclidienne, ou la distance de Hamming.
- P(k) est une pondération qui diffère suivant la transition locale (i(k),j(k)) -----> (i(k+1),j(k+1)).
- N(P) est un facteur de normalisation dont le rôle est de rendre D(w) indépendante de la longueur du chemin de recalage.

$\hat{w}$ est donc la fonction de recalage qui minimise la fonctionnelle D, soit:

$$\hat{w} = \text{Argmin } D(w).$$

(la fonction Argmin f(x) donne la valeur de l'argument x qui rend la fonction f minimale).

En réalité, lorsque l'on effectue une reconnaissance ce n'est pas $\hat{w}$ qui nous intéresse directement.

L'information importante est D(T,R), taux de dissemblance entre les deux formes comparées T et R, D(T,R) est défini par la relation:

$$D(T,R) = D(\hat{w})$$

soit

$$D(T,R) = \min_{K,i(k),j(k)} \frac{\sum_{k=1}^{K} d(i(k),j(k)) \times P(k)}{N(P)}$$

Explicitons plus avant ce que sont les pondérations P(k).

Habituellement, on considère deux types de pondération:

- La pondération asymétrique notée Pa:

$$Pa(k) = i(k) - i(k-1)$$
$$\text{ou} \quad = j(k) - j(k-1)$$

suivant qu'elle est asymétrique par rapport à T ou par rapport à R.

- La pondération symétrique que nous noterons Ps:

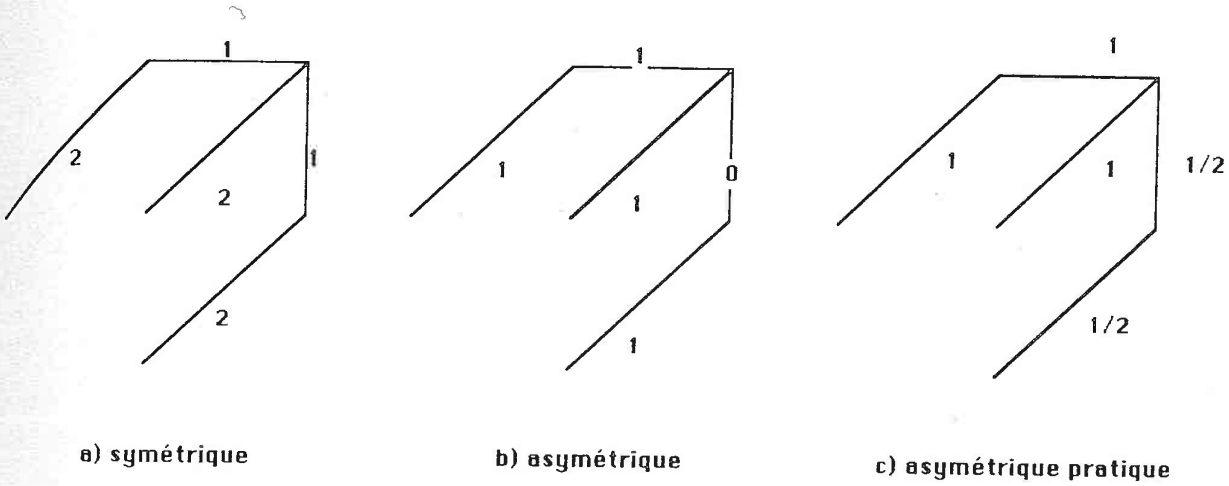
$$Ps(k) = i(k) - i(k-1) + j(k) - j(k-1)$$

Ces deux types de pondérations vont nous amener à distinguer deux types de contraintes locales:

- Si la pondération utilisée est la pondération asymétrique Pa, la contrainte locale sera dite asymétrique.

- Les contraintes locales dites symétriques sont pondérées par Ps.

La figure 5 donne la contrainte symétrique (a) et la contrainte asymétrique (b) de Sakoe et Chiba.



a) symétrique b) asymétrique c) asymétrique pratique

figure 5: Contrainte de Sakoe et Chiba.

La pondération asymétrique de Sakoe et Chiba (figure 5b) est essentiellement théorique. En pratique, on utilise plutôt celle représentée figure 5c afin de ne pas avoir un arc de poids nul.

Si à tout arc élémentaire $\{(i(k-1),j(k-1)),(i(k),j(k))\}$ d'un chemin de recalage w, on associe les pondérations P(k), la longueur du chemin de recalage w s'exprime par:

$$L(w) = \sum_{k=1}^{K} P(k)$$

Puisque N(P) est un facteur de normalisation chargé de rendre la métrique D (w) associée à w indépendante de la longueur du chemin de recalage, il apparait judicieux de choisir:

$$N(P) = L(w) = \sum_{k=1}^{K} P(k)$$

Ainsi, par exemple, dans le cas où la pondération utilisée est la pondération symétrique Ps, on obtient:

$$N(Ps) = I+J$$

Dans le cas où la pondération P est asymétrique en T, c'est à dire $P(k) = i(k) - i(k-1)$, N(Pa) vaut alors:

$$N(Pai) = I$$

Si $Pa(k) = j(k)-j(k-1)$ alors $N(Paj) = J$

On constate donc que si les points terminaux sont fixes, si la pondération choisie est Pa ou Ps, la longueur L(w) est une constante indépendante de la fonction de recalage w que l'on considère.

Le problème de minimisation précédent revient dans ce cas à minimiser:

$$D_p = \min_{K,i(k),j(k)} \left[\sum_{k=1}^{K} d(i(k),j(k)) \times P(k) \right]$$

et le taux de dissemblance D(T,R) entre les deux formes comparées T et R s'écrit alors:

$$D(T,R) = \frac{D_p}{N(P)}$$

La détermination du taux de dissemblance peut alors se résoudre par programmation dynamique, grâce au principe d'optimalité locale de Bellman [Bellman 57]:

Soit $C_{(1,1)}^{(i,j)}$ le chemin optimal joignant les points (1,1) et (i,j), alors pour tout point (i',j') appartenant à $C_{(1,1)}^{(i,j)}$, le chemin partiel $C_{(1,1)}^{(i',j')}$ est optimal.

Soit Dpp(i,j) la distance cumulée optimale au point (i,j) associée au chemin partiel optimal $C_{(1,1)}^{(i,j)}$ on a:

$$Dpp(i,j) = \min_{K',i(k),j(k)k=1} \sum_{K'} d(i(k),j(k))\times P(k)$$

avec $i(1) = 1$ et $i(K') = i$

$j(1) = 1$ $\quad$ $j(K') = j$

D'après le principe d'optimalité locale de Bellman,il vient :

$$Dpp(i,j) = \min_{(i',j')\in V(i,j)} Dpp(i',j')+dp((i',j'),(i,j))$$

où:

- (i',j') est un point appartenant au voisinage V(i,j) du point (i,j) défini par la contrainte locale utilisée.

- dp ((i',j'),(i,j)) est la distance locale entre les deux points (i',j') et (i,j).

Explicitons sur quelques exemples cette relation.

- Contrainte symétrique de Sakoe et Chiba (figure 5a)

Nous avons alors:

$$V(i,j) = \{(i-2,j-1)(i-1,j-1),(i-1,j-2)\}$$

Calculons les distances locales correspondantes:

$$dp((i-2,j-1),(i,j)) = 2.d(i-1,j) + d(i,j)$$
$$dp((i-1,j-1),(i,j)) = 2.d(i,j)$$
$$dp((i-1,j-2),(i,j)) = 2.d(i-1,j-2) + d(i,j)$$

Le problème s'exprime alors par les relations suivantes:

$$Dpp(i,j) = \min \begin{cases} Dpp(i-2,j-1)+2.d(i-1,j)+d(i,j) \\ Dpp(i-1,j-1)+2.d(i,j) \\ Dpp(i-1,j-2)+2.d(i-1,j-2)+d(i,j) \end{cases}$$

qui constituent les relations traditionnelles de la programmation dynamique.

Nous pouvons alors évaluer Dpp(i,j) en tout point (i,j) du plan de comparaison $1 \leq i \leq I$ et $1 \leq j \leq J$.

Le taux de dissemblance D(T,R) entre les deux formes T et R mises en comparaison est:

$$D(T,R) = \frac{Dpp(I,J)}{I+J}$$

- Contrainte asymétrique pratique de Sakoe et Chiba (figure 5c)

Nous avons toujours $V(i,j) = \{(i-2),j-1),(i-1,j-1),(i-1,j-2)\}$

Les distances locales correspondantes sont alors:

$$dp((i-2,j-1),(i,j)) = d(i-1,j) + d(i,j)$$
$$dp((i-1,j-1),(i,j)) = d(i,j)$$
$$dp((i-1,j-2),(i,j)) = 0,5 \cdot d(i,j-1) + 0,5 \cdot d(i,j)$$

Le problème s'exprime alors par les relations suivantes:

$$Dpp(i,j) = \text{Min} \begin{cases} Dpp(i-2,j-1) + d(i-1,j) + d(i,j) \\ Dpp(i-1,j-1) + d(i,j) \\ Dpp(i-1,j-2) + 0,5 \cdot d(i,j-1) + 0,5 \cdot d(i,j) \end{cases}$$

Le taux de dissemblance entre les formes T et R s'écrit alors:

$$D(T,R) = \frac{Dpp(I,J)}{I}$$

- Contrainte symétrique sans condition de pente (figure 6).

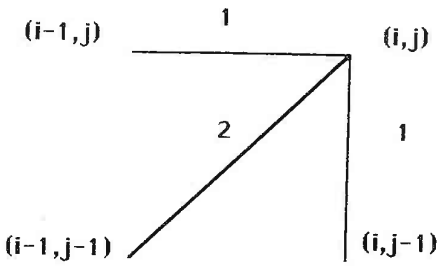


figure 6: contrainte symétrique sans condition de pente.

Le voisinage V(i,j) est l'ensemble {(i-1,j),(i-1,j-1),(i,j-1)}

Les distances locales correspondantes s'écrivent:

$$dpp((i-1,j),(i,j)) = d(i,j)$$
$$dpp((i-1,j-1),(i,j)) = 2 \cdot d(i,j)$$
$$dpp((i,j-1),(i,j)) = d(i,j)$$

Le problème s'écrit donc:

$$Dpp(i,j) = \text{Min} \begin{cases} Dpp(i-1,j) + d(i,j) \\ Dpp(i-1,j-1) + 2 \cdot d(i,j) \\ Dpp(i,j-1) + d(i,j) \end{cases}$$

Nous pouvons maintenant calculer le taux de dissemblance:

$$D(T,R) = \frac{Dpp(I,J)}{I+J}$$

Nous allons maintenant donner un algorithme simple de détermination du taux de dissemblance.

3. UN ALGORITHME DE PROGRAMMATION DYNAMIQUE

Un algorithme général de calcul du taux de dissemblance entre les deux formes T et R s'écrit simplement de la façon suivante:

- Initialisation:

$$Dpp(1,1) = P(1) \cdot d(1,1)$$

- Programmation dynamique:

```
pour 1≤i≤I faire
   pour 1≤j≤J faire
       évaluer Dpp(i,j)
   fin
fin
```

- Détermination du taux de dissemblance:

$$D(T,R) = \frac{Dpp(I,J)}{N(P)}$$

Si l'on désire connaître le chemin optimal et non seulement la distance cumulée le long de ce chemin, il est nécessaire de mémoriser le choix effectué pour la fonction de minimisation pour chaque point (i,j).

Nous pouvons toute de suite faire une première remarque.

Tel qu'il est décrit, l'algorithme de programmation nécessite l'utilisation d'un tableau Dpp à deux dimensions i,j avec $1 \leq i \leq I$ et $1 \leq j \leq J$.

Or nous constatons, que suivant la contrainte utilisée seules deux colonnes ou trois colonnes sont nécessaires. Prenons deux exemples:

- Contrainte sans condition de pente (figure 6)

Pour évaluer la valeur de Dpp en (i,j), nous n'avons besoin que des valeurs en (i-1,j),(i-1,j-1),(i,j-1).

Il suffit donc de définir Dpp comme un tableau à deux dimensions (i,j) mais avec cette fois $1 \leq i \leq 2$ et $1 \leq j \leq J$. La figure 7 montre comment évolue ces colonnes numérotées C1 et C2 par permutation circulaire.

Un tel algorithme (programmation dynamique avec la contrainte sans condition de pente) est quelque fois appelé à un niveau de profondeur.

- Contrainte de Sakoe et Chiba (figure 5).

Les points (i-2,j-1) et (i-1,j-1) et (i-1,j-2) sont nécessaires à l'évaluation de Dpp(i,j).

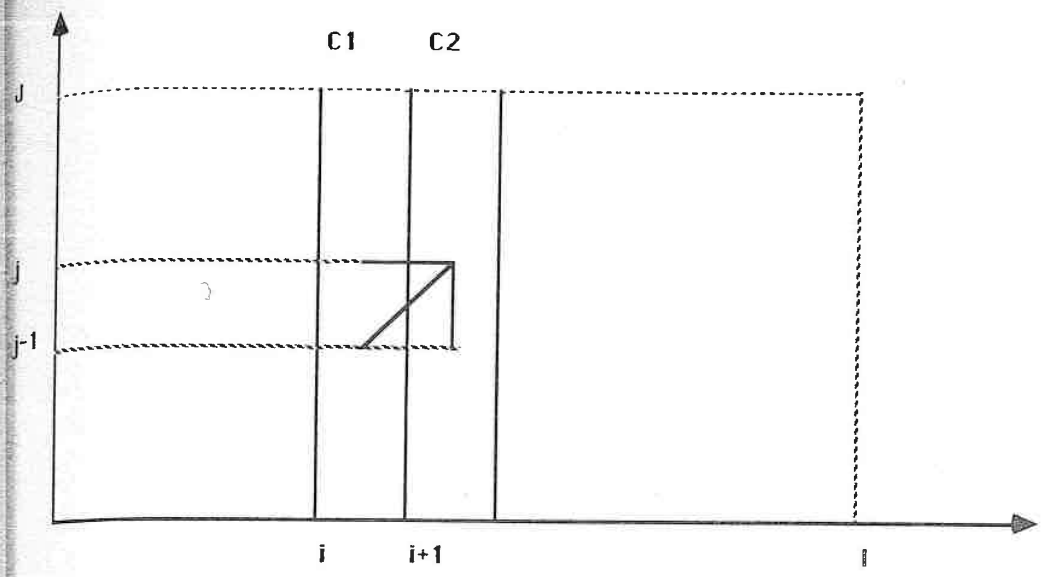
Il suffit donc de considérer Dpp comme un tableau à deux dimensions (i,j) avec $1 \leq i \leq 3$ et $1 \leq j \leq J$.

La figure 8 montre comment les trois colonnes C1,C2,C3 sont évaluées.

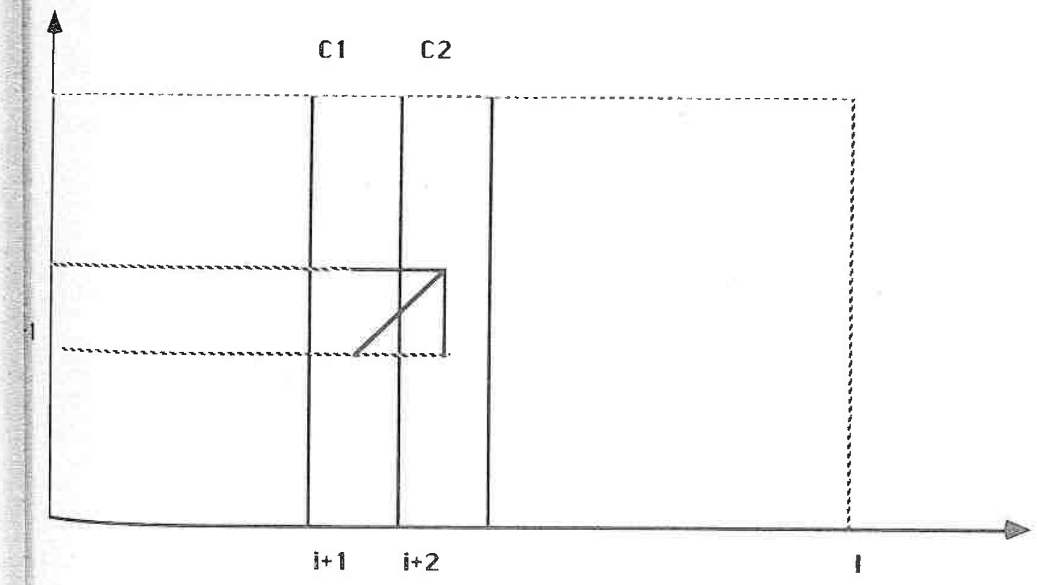
Un tel algorithme est parfois dit à deux niveaux de profondeur.

4. CONCLUSION

L'algorithme de programmation dynamique tel que nous venons de le voir nécessite le calcul des distances locales d(i,j) pour tous les i et pour tous les j. Or en fonction de la contrainte locale, il est possible de définir une zone du plan à l'extérieur de laquelle il est inutile de rechercher le chemin de recalage. Cette réduction de l'espace de comparaison sera l'objet du chapitre suivant.



Détermination de C2 à l'étape i+1.



Détermination de C2 à l'étape i+2.

figure 7: évolution des colonnes C1 et C2.

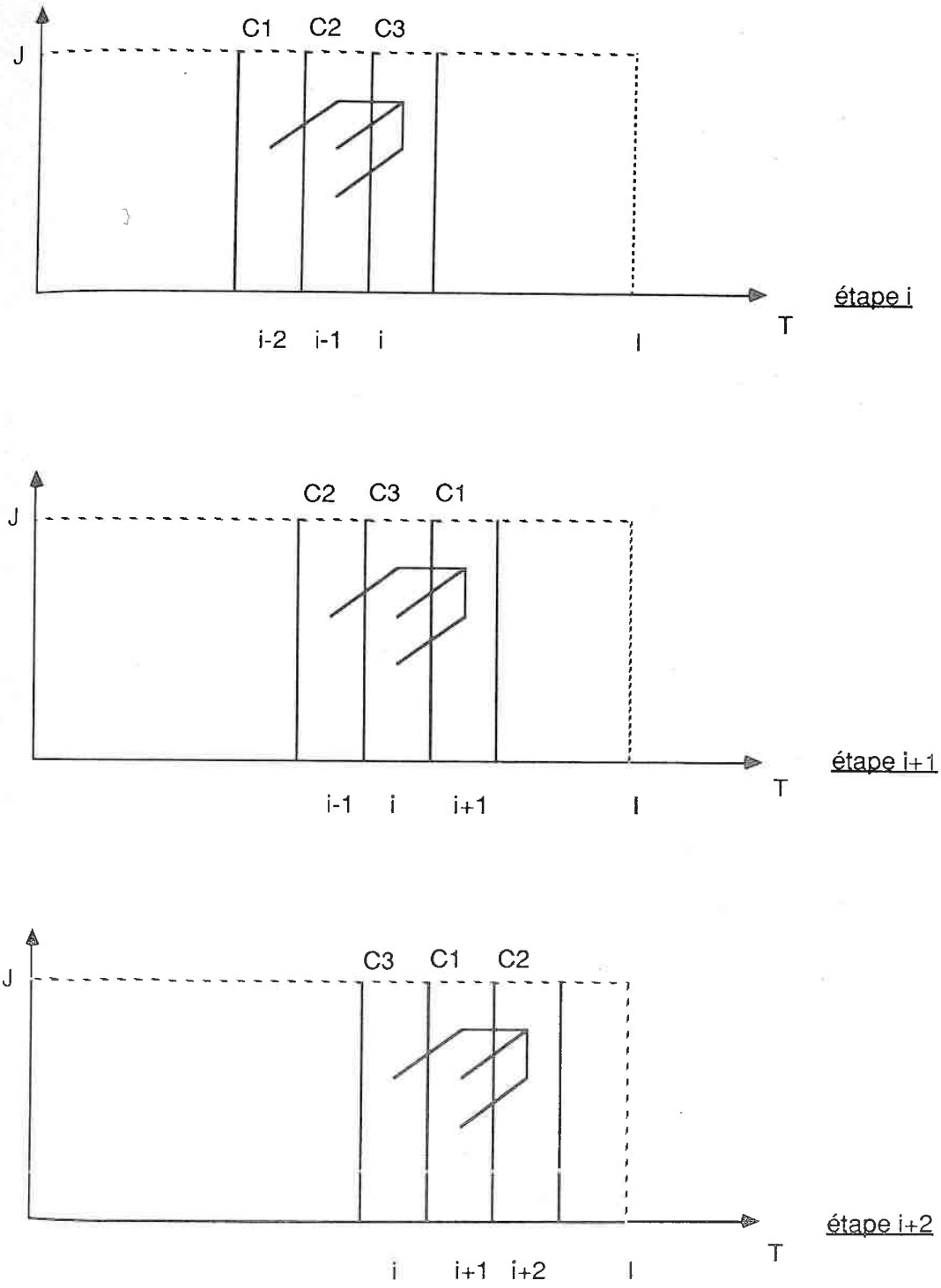


figure 8: évolution des colonnes C1, C2 et C3.

CHAPITRE 2: REDUCTION DE L'ESPACE DE COMPARAISON, ALGORITHME A OPTIMUMS LOCAUX

Dans ce chapitre, nous allons décrire quelques méthodes qui permettent de diminuer le nombre de calculs à effectuer. Dans un premier paragraphe, nous nous intéressons à la réduction de l'espace de comparaison afin de minimiser le nombre de distances locales à déterminer. Dans un deuxième paragraphe, nous verrons quelques algorithmes à optimums locaux.

1. LIMITATION DU DOMAINE DE RECHERCHE

La figure 1 montre le domaine où le chemin de recalage est forcé de se trouver si l'on utilise la contrainte de Sakoe et Chiba.

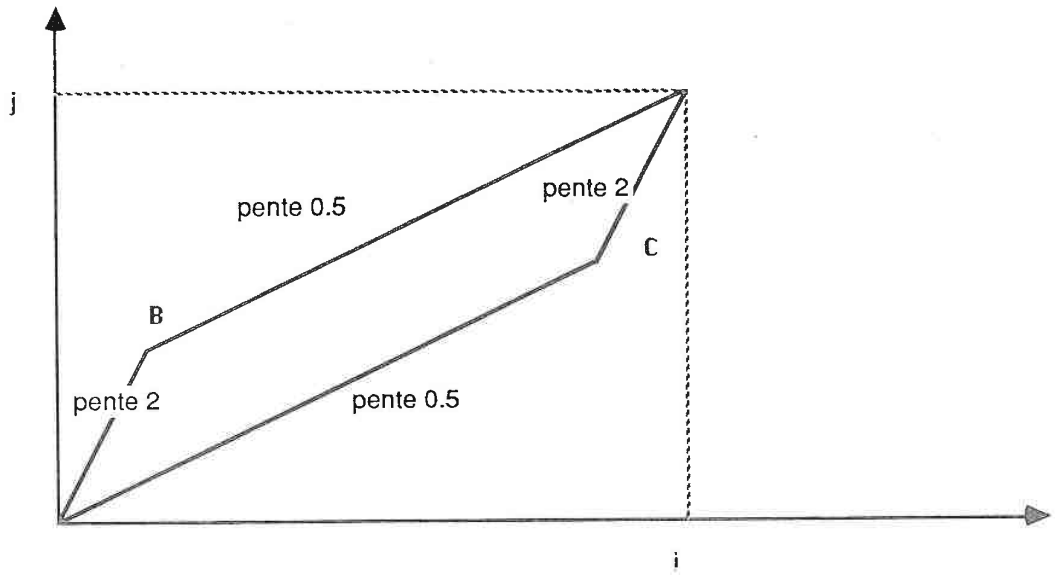


figure 1: Domaine contenant le chemin de recalage.

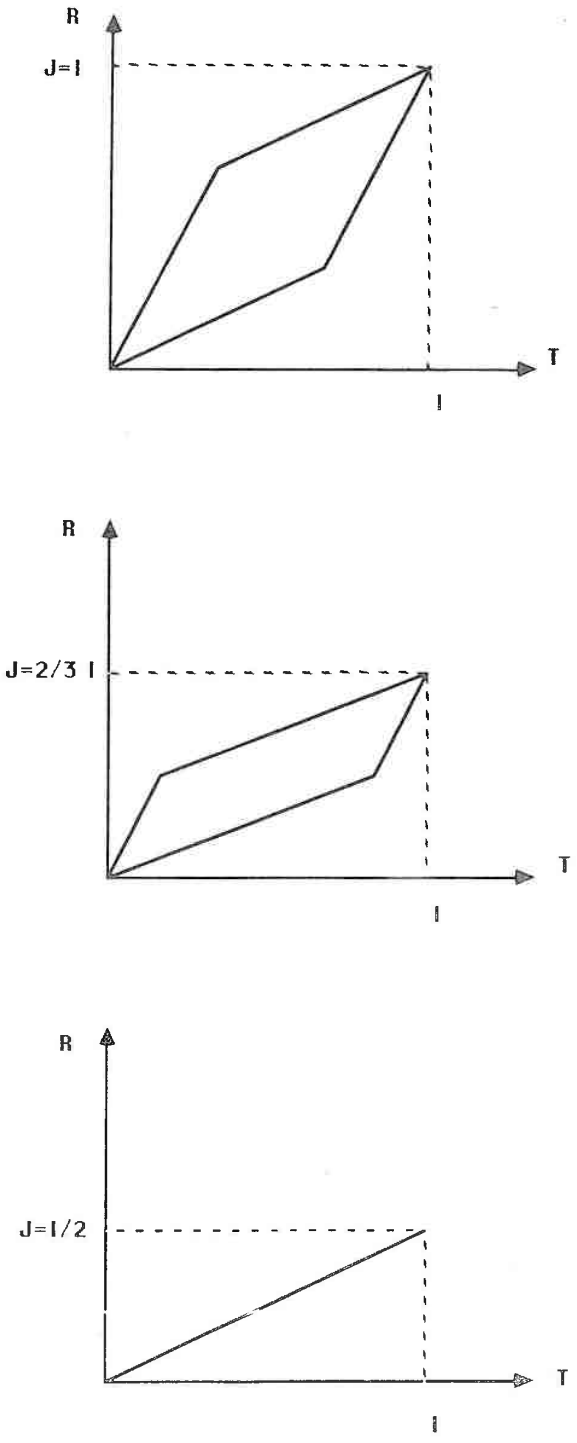


figure 2: taille du parallélogramme.

Il est bien sûr très intéressant de réduire l'espace de comparaison du point de vue du temps de calcul, mais encore faut-il que la limitation de l'espace de recherche ne soit pas trop sévère afin de ne pas dégrader les performances de l'algorithme.

Quelques résultats établis par Myers [Myers 80] sont illustrés figure 2.ℓ Cette figure montre la taille du parallèlogramme dans lequel se trouve le chemin de recalage optimal (contrainte de Sakoe et Chiba ou Itakura) en fonction du rapport I/J.

Nous pouvons remarquer que lorsque J = 2/3 I, la taille du parallélogramme (c'est à dire la région dans laquelle le chemin de recalage optimal est contraint de se trouver) diminue fortement, et dans le cas extrême J=0,5*I, un seul chemin est possible, c'est la diagonale qui détermine une transformation linéaire. Dans le cas limite J < I/2, ce domaine n'existera pas et la recherche va avorter.

Myers a alors montré qu'il est possible de définir en fonction de la contrainte locale utilisée une zone en dehors de laquelle il est inutile de chercher le chemin de recalage optimal.

Notons:

- Emin la pente minimale du chemin local,

- Emax la pente maximale.

Le domaine de recherche du chemin de recalage est défini par les deux relations suivantes:

$$\begin{cases} 1 + EMIN\ [i(k)-1] \leq j(k) \leq 1 + EMAX\ [i(k)-1] \\ J + EMAX\ [i(k)-I] \leq j(k) \leq J + EMIN\ [i(k)-I] \end{cases}$$

Reprenons l'exemple de la contrainte de Sakoe et Chiba.

Nous avons alors:

EMIN = 0,5

EMAX = 2

La fenêtre d'exploration est visualisée sur la figure 3. Le nombre de distances à calculer dépend de la longueur des deux formes.

Sakoe [Sakoe 78] a traduit le fait que le i(k)ème prélevement de la forme $\vec{T}$ ne peut pas être en retard ou en avance de plus d'un certain nombre Rt de prélevements sur le j(k)ème prélèvement de la forme R par la relation:

$$|i(k)-j(k)| \leq Rt$$

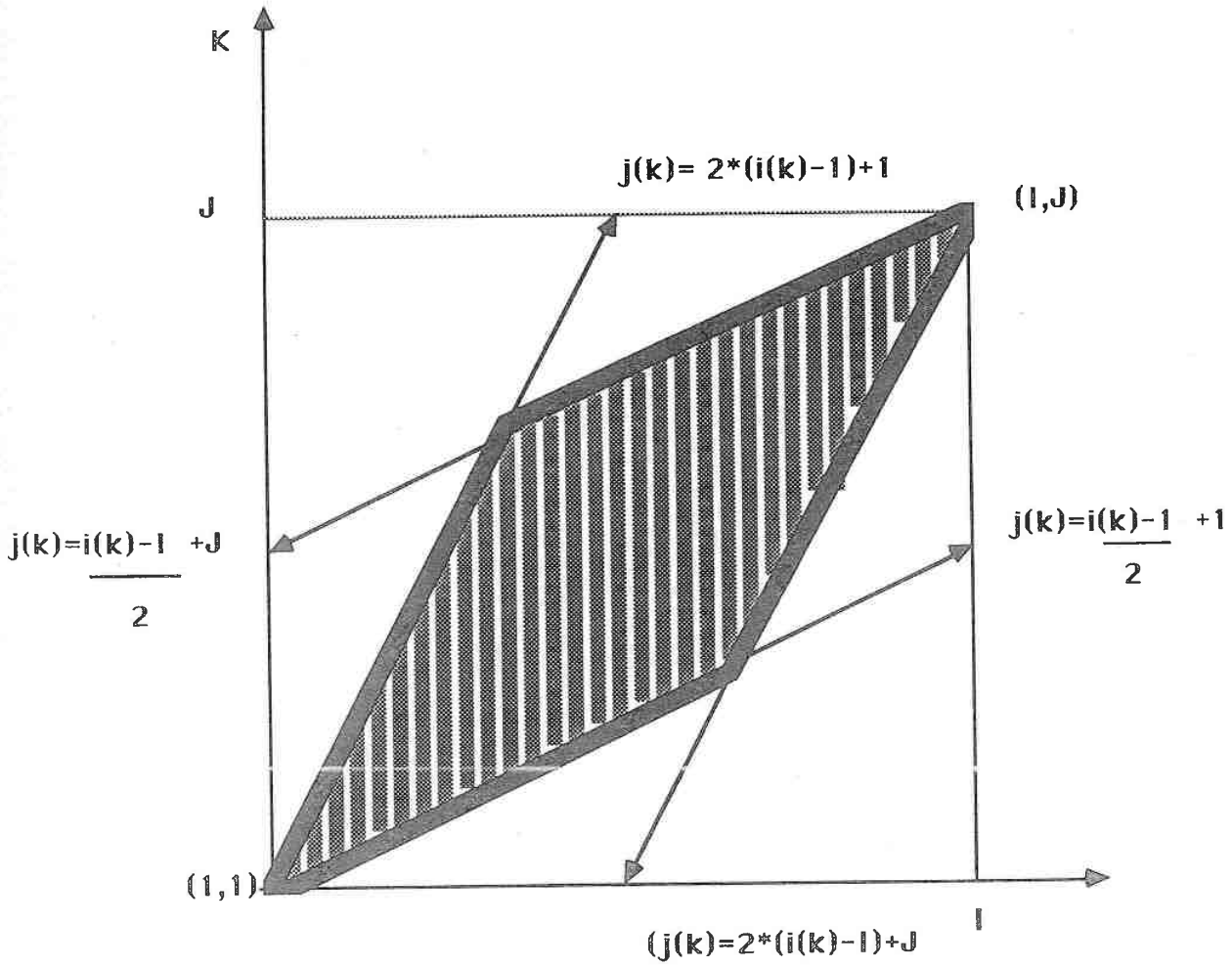


figure 3: fenêtre d'exploration d'après Myers.

Sur la figure 4 est visualisée la fenêtre d'exploration définie par Sakoe et Chiba.

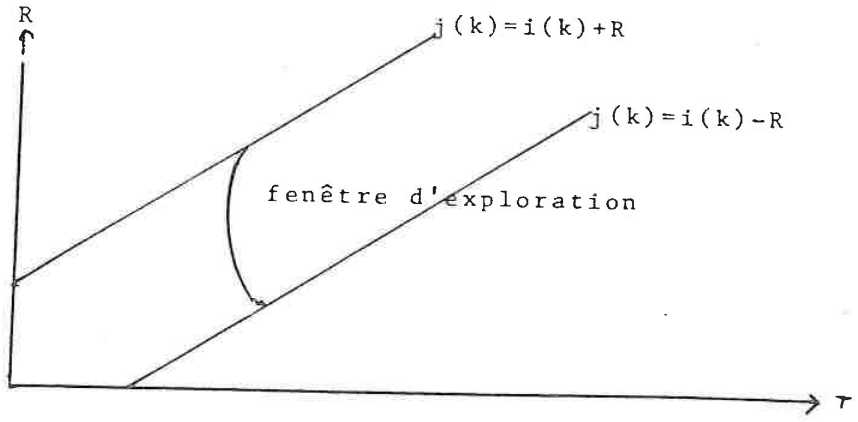
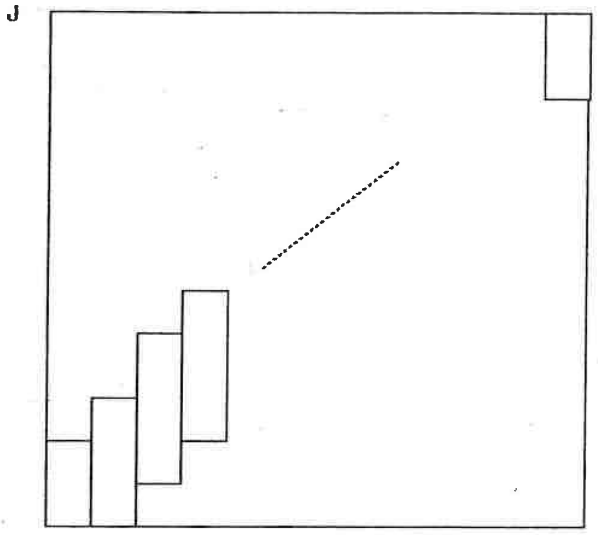
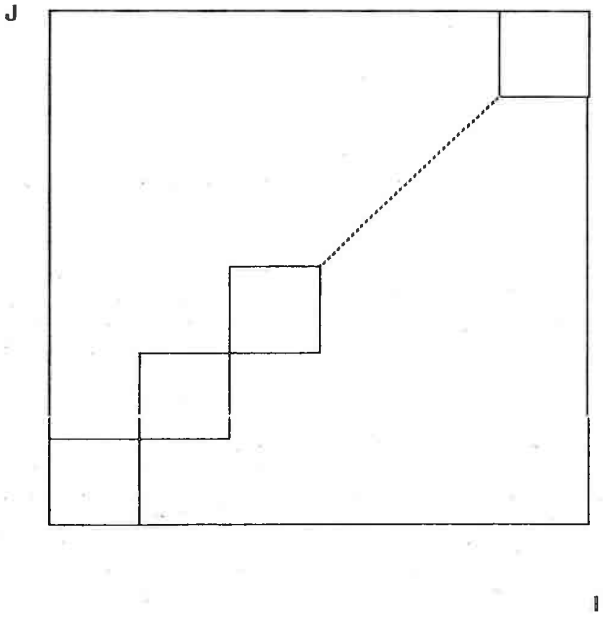


figure 4: domaine de recherche d'après Sakoe et Chiba.

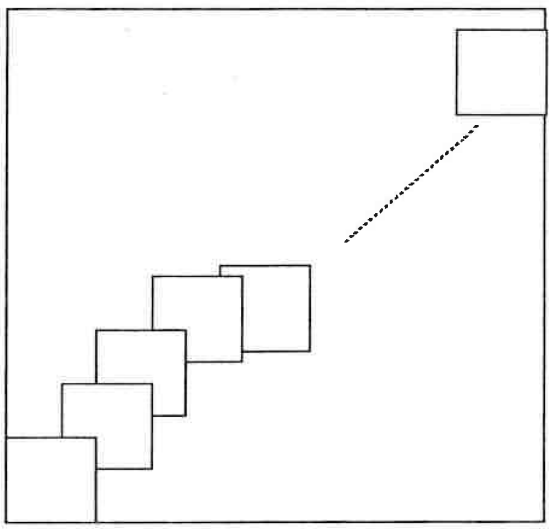


Pavés unicolonnes centrés
sur la diagonale



Pavé matriciels centrés
sur la diagonale

FIGURE 5: Pavés de position prédéfinie



**Pavés matriciels évolutifs:
Le sommet inférieur gauche
du pavé est positionné sur le
minimum trouvé au pavé (1-i)
d'aprés [Haton 74]**

FIGURE 6 Positions évolutives

2. ALGORITHMES A OPTIMUMS LOCAUX

([Haton 74])

Les algorithmes à optimums locaux abandonnent le principe d'optimalité globale afin de diminuer le nombre de distances à calculer au cours de la comparaison et par suite le temps de réponse.

La fonctionnelle associée à ces algorithmes est la somme des minima locaux dans des pavés du plan de comparaison:

$$D(T,R) = \frac{1}{NP} \sum_{k=1}^{NP} \underset{n}{\text{Min}}\ d(ik_n, jk_n)$$

où:

-NP est le nombre de pavés

-(ikn,jkn) est un point quelconque du kème pavé.

Les pavés peuvent être prédéfinis (figure 5) ou ils peuvent avoir une position évolutive (figure 6).

Nous pouvons constater que ces algorithmes seront mis en défaut si le chemin de recalage "réel" entre deux formes s'écarte de la diagonale puisqu'ils se limitent à évaluer le score d'un chemin sous optimal appartenant à un voisinage de la diagonale.

Le nombre de distances à déterminer, et donc le temps de calcul dépendent du nombre de pavés et de leur taille.

CHAPITRE 3: RELACHEMENT DES CONTRAINTES

Un des inconvénients majeurs de l'algorithme de programmation dynamique que nous avons décrit dans le chapitre 2 réside dans les conditions aux frontières: il suppose que les deux formes T et R correspondent exactement à leur début et à leur fin.

Si la détermination du point initial et du point final a été effectuée correctement sur les deux formes, cette contrainte est acceptable et ne détériore pas les performances de l'algorithme. Mais dans tous les cas où la segmentation est imparfaite, les conditions aux frontières sont trop sévères et nuisent à une bonne reconnaissance. Plusieurs variations de l'algorithme de programmation dynamique ont été proposées.

1. METHODE UE2-1

Cette méthode a été proposée par Rabiner [Rabiner 78] et s'appelle "unconstrained endpoints, 2 to 1 slope range".

Elle conserve toutes les contraintes locales mais relâche les conditions aux frontières sur l'axe vertical de la façon suivante:

$$1 \leq j(1) \leq 1+\& \quad \text{et} \quad J-\& \leq j(K) \leq J$$

où K est le nombre de points du chemin de recalage et & un paramètre de l'algorithme de programmation dynamique.

Si & = 0, l'algorithme UE2-1 décrit ci-dessus devient l'algorithme simple de programmation dynamique avec conditions aux frontières. Pour des valeurs de & différentes de 0, la zone du plan où le chemin de recalage peut se trouver est augmentée de façon significative, comme le montre la figure 1.

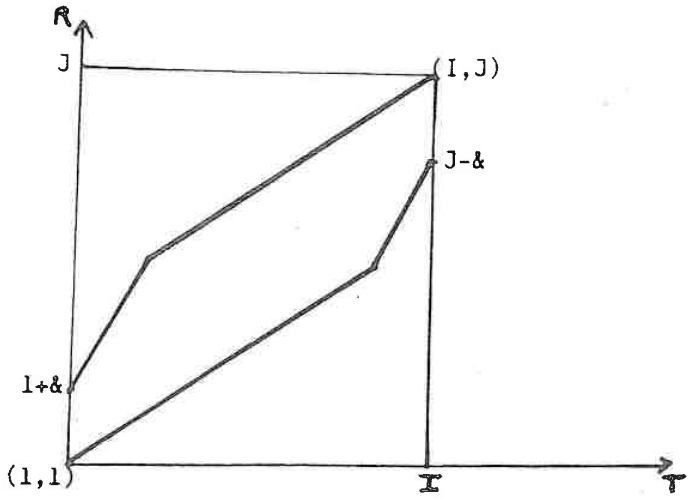


figure 1: domaine du chemin de recalage avec l'algorithme UE2-1.

Si la contrainte utilisée est asymétrique, il est inutile de redéfinir les relations de programmation dynamique puisque tous les chemins ont la même longueur.

2. UNCONSTRAINED ENDPOINTS LOCAL MINIMUM: UELM

Rabiner [Rabiner 78] [Rabiner 81] a proposé une autre variante de l'algorithme de programmation dynamique autorisant le relâchement aux contraintes, qu'il a baptisé UELM.

Il utilise toujours la condition précédente:

$$1 \leq j(1) \leq 1+\&$$
$$J-\& \leq j(k) \leq J$$

mais il ajoute une contrainte supplémentaire de la forme:

$$Max(1,P(i)-E) \leqq j \leqq Min(P(i)+E,j)$$

où P(i) désigne la position du minimum des distances cumulées Dpp(i-1,j) évaluées à l'étape i-1 dans la fenêtre d'exploration correspondante.

Cette nouvelle condition détermine à chaque étape i de la comparaison la fenêtre de largeur 2E susceptible de contenir le chemin de recalage en fonction de la position de la fenêtre à l'étape i-1.

Comme le montre la figure 2, l'algorithme UELM trace le chemin localement optimal afin d'estimer le chemin globalement optimal avec un minimum de calculs. De façon claire la contrainte en fin de mots i(K)=I et j(K)=J est éliminée puisque le chemin détermine lui-même le point final de la comparaison.

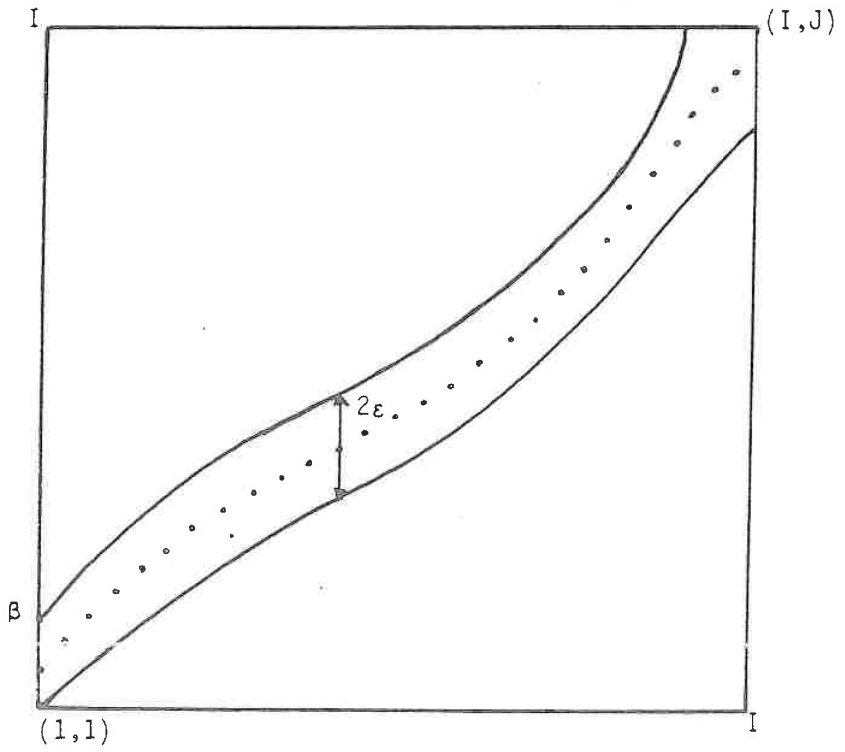


figure 2: domaine de recherche du chemin de recalage d'après UELM.

Avec ce type de contrainte globale, une difficulté peut survenir, toujours dans le cas d'une pondération asymétrique, si le chemin de recalage aboutit à l'ordonnée J avant d'arriver à l'abscisse I (figure 3).

En effet, dans ce cas le facteur de normalisation dépend de l'abscisse I du point terminal du chemin de recalage. La distance cumulée est donc pondérée par le rapport I/IF, afin de la rendre indépendante de la longueur du chemin de recalage.

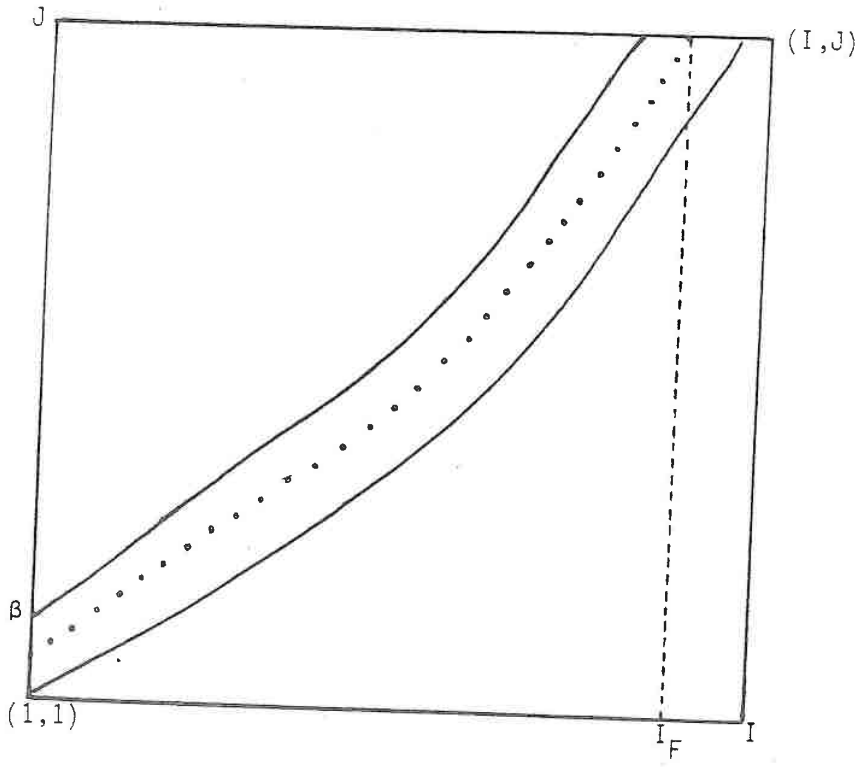


figure 3: exemple de chemin de recalage n'aboutissant pas en I.

3. RELACHEMENT SUR LES DEUX AXES

Un certain nombre de points peuvent être omis aux extrémités des deux formes T et R (figure 4) [di Martino 83] [di Martino 84] [Boyer 85].

Cela aide à la mise en comparaison des deux formes: de cette manière des erreurs sont corrigées dans le cas où les deux formes ou

l'une d'entre elles sont trop longues. Mais il existe un certain nombre de cas où cette méthode engendre de nouvelles erreurs puisque l'unique possibilité est de raccourcir les formes. Si le vocabulaire contient par exemple des mots similaires dont les différences se situent à leur début ou à leur fin, ces parties peuvent être omises lors du calcul du score et de nouvelles erreurs vont apparaitre.

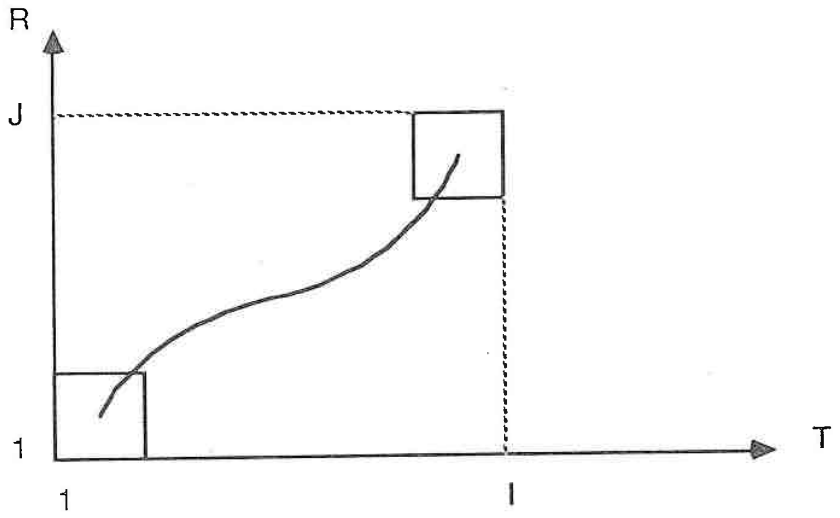


figure 4: exemple de chemin de recalage avec relâchement sur les 2 axes.

Les relations récursives de programmation dynamique doivent être redéfinies globalement car elles sont fondées sur deux hypothèses (contraintes aux frontières, longueur constante de tous les chemins de recalage grâce à un choix judicieux des fonctions de pondération) qui ne sont plus vérifiées. Le principe d'optimalité locale doit être modifié afin que de nouvelles relations de programmation dynamique tenant compte de la longueur variable des chemins de recalage puissent être établies.

3.1. Nouveau principe d'optimalité locale de Bellman

Soit D un sous ensemble du plan de comparaison contenant l'ensemble des points pouvant être origine d'un chemin optimal de recalage.

Soit P: P------>D la fonction qui à tout point (i,j) du plan P associe l'origine du chemin partiel optimal aboutissant en ce point.

Soit

$$c^{(i,j)}_{p[(i,j)]}$$

le chemin optimal joignant les points P[(i,j)] et (i,j). Pour tout point (i',j') appartenant à

$$c^{(i,j)}_{p[(i,j)]}$$

on a:

$$c^{(i',j')}_{p[(i',j')]}$$

est optimal et P[(i',j')] = P[(i,j)].

Ce nouveau principe d'optimalité est en quelque sorte une généralisation de celui que nous avons énonçé au chapitre 1 et il va nous permettre de déduire une nouvelle formulation plus générale des relations récursives de programmation dynamique.

3.2. Généralisation des relations récursives de programmation dynamique.

Pour calculer la distance cumulée Dpp(i,j) en tout point (i,j) du plan de comparaison en fonction des distances Dpp(i',j') évaluées aux points (i',j') appartenant au voisinage de (i,j) défini par la contrainte locale utilisée, étant donné la nouvelle formulation du principe d'optimalité de Bellman, il faut déterminer le point $(\hat{i},\hat{j})$ pour lequel passe le chemin optimal d'extrêmité (i,j).

Le "meilleur" point précédent (i,j) est calculé par la relation:

$$(\hat{i},\hat{j}) = \underset{(i',j')\in V(i,j)}{\text{Argmin}} \frac{Dpp(i',j')+dp((i',j')+(i,j))}{L(C^{(i,j)}_{p[(i',j')]})}$$

où:

$$L(C^{(i,j)}_{p[(i',j')]})$$

est la longueur du chemin de recalage aboutissant en (i',j') et débutant en P[(i',j')].

De là, nous pouvons évaluer:

$$Dpp(i,j) = Dpp(\hat{i},\hat{j}) + dp((\hat{i},\hat{j}),(i,j))$$

Ces deux dernières relations peuvent être considérées comme une généralisation des relations classiques et nous pouvons constater qu'aucune hypothèse n'a été formulée sur le type de contrainte locale utilisée (symétrique, asymétrique...).

3.3. Spécification de l'algorithme

3.3.1. notations

- ideb: abcisse du dernier point pouvant être l'origine d'un chemin de recalage.
- ifin: abcisse du premier point pouvant être l'extrémité d'un chemin de recalage.

- jdeb: ordonnée du dernier point pouvant être l'origine d'un chemin de recalage.

- jfin: ordonnée du premier point pouvant être l'extrémité d'un chemin de recalage.

- Dpp(i,j): distance cumulée au point (i,j).

- Lpp(i,j): longueur du chemin aboutissant en (i,j).

- V(i,j): voisinage du point (i,j) défini par la contrainte locale utilisée.

- P: pondération définie par la contrainte locale utilisée.

3.3.2. algorithme

1)initialisation
pour: $0 \leq j \leq$ jdeb faire:
Dpp(0,j) = 0
Lpp(0.j) = P(0)
fin pour

pour: jdeb+1 $\leq j \leq J$ faire:
Dpp[0,j] = infini
Lpp[0,j] = 1
fin pour

2)programmation dynamique
pour $1 \leq i \leq I$ faire
pour $1 \leq j \leq J$ faire

$$d = \min_{(i',j') \in V(i,j)} \frac{Dpp(i',j')+dp((i',j'),(i,j))}{Lpp(i',j')+lp((i',j'),(i,j))}$$

$$(\hat{i},\hat{j}) = \operatorname{Argmin}_{(i',j') \in V(i,j)} \frac{Dpp(i',j')+dp((i',j'),(i,j))}{Lpp(i',j')+lp((i',j'),(i,j))}$$

si $i \leqq ideb$ et $j \leqq jdeb$ et $d(i,j) < \hat{d}$

alors $Dpp(i,j) = P(0)d(i,j)$

$Lpp(i,j) = P(0)$

sinon $Dpp(i,j) = Dpp(\hat{i},\hat{j})+dp((\hat{i},\hat{j})(i,j))$

$Lpp(i,j) = Lpp(\hat{i},\hat{j})+lp((\hat{i},\hat{j})(i,j))$

finsi

finpour

finpour

3)évaluation du taux de dissemblance

$$D(T,R) = \min_{\substack{i \geqq I-ifin \\ j \geqq J-jfin}} \frac{Dpp(i,j)}{Lpp(i,j)}$$

4. ALGORITHME A POINT AJUSTE

Dans cette première méthode [Das 80], décrite figure 5, la forme de référence R n'est pas modifiée, mais de nouveaux points extrêmes sont calculés pour la forme inconnue T. Une fenêtre d'ajustement est spécifiée autour des points finaux de la forme test T. Le point à l'intérieur de la fenêtre qui a la distance cumulée la plus petite est choisi comme point ajusté. La faiblesse d'une telle méthode est à la

fois que la détermination des nouveaux points extrêmes est effectuée avant le recalage temporel, mais aussi que les points extrêmes de la forme de référence ne peuvent pas être modifiés. Ainsi, il n'y a aucune garantie d'amélioration de la comparaison et cette méthode corrige et génère des erreurs.

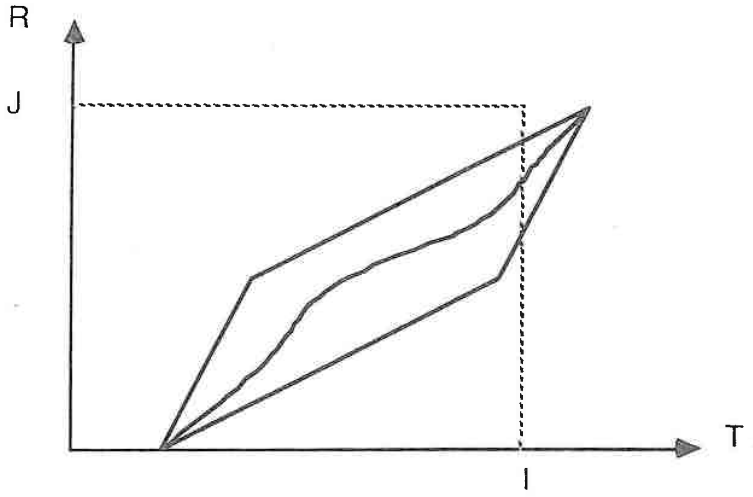


figure 5: principe de l'algorithme à point ajusté.

Un but de cette technique pour compenser les erreurs de détection des début et fin d'élocution est d'inclure les points déterminés par les points extrêmes exacts des deux formes dans le domaine de recherche. Un moyen simple d'y arriver est d'allonger les formes en ajoutant un nombre suffisant de points aux deux extrêmités de telle sorte que des points correspondant au silence figurent.

L'augmentation de la place mémoire nécessaire n'est pas la seule faiblesse de cette méthode. Si le chemin de recalage juste est trouvé, il compare du silence avec du silence et de la parole avec de la parole. La distance entre T et R ne dépend pas seulement de la parole mais également du silence qui l'entoure.

Comme la distance due aux parties de silence varie avec la forme de référence, ces différences inutiles génèrent de nouvelles erreurs.

Haltsonen [Haltsonen 84] a généralisé cette méthode en étendant les deux formes et non pas uniquement la forme de référence (figure 6).

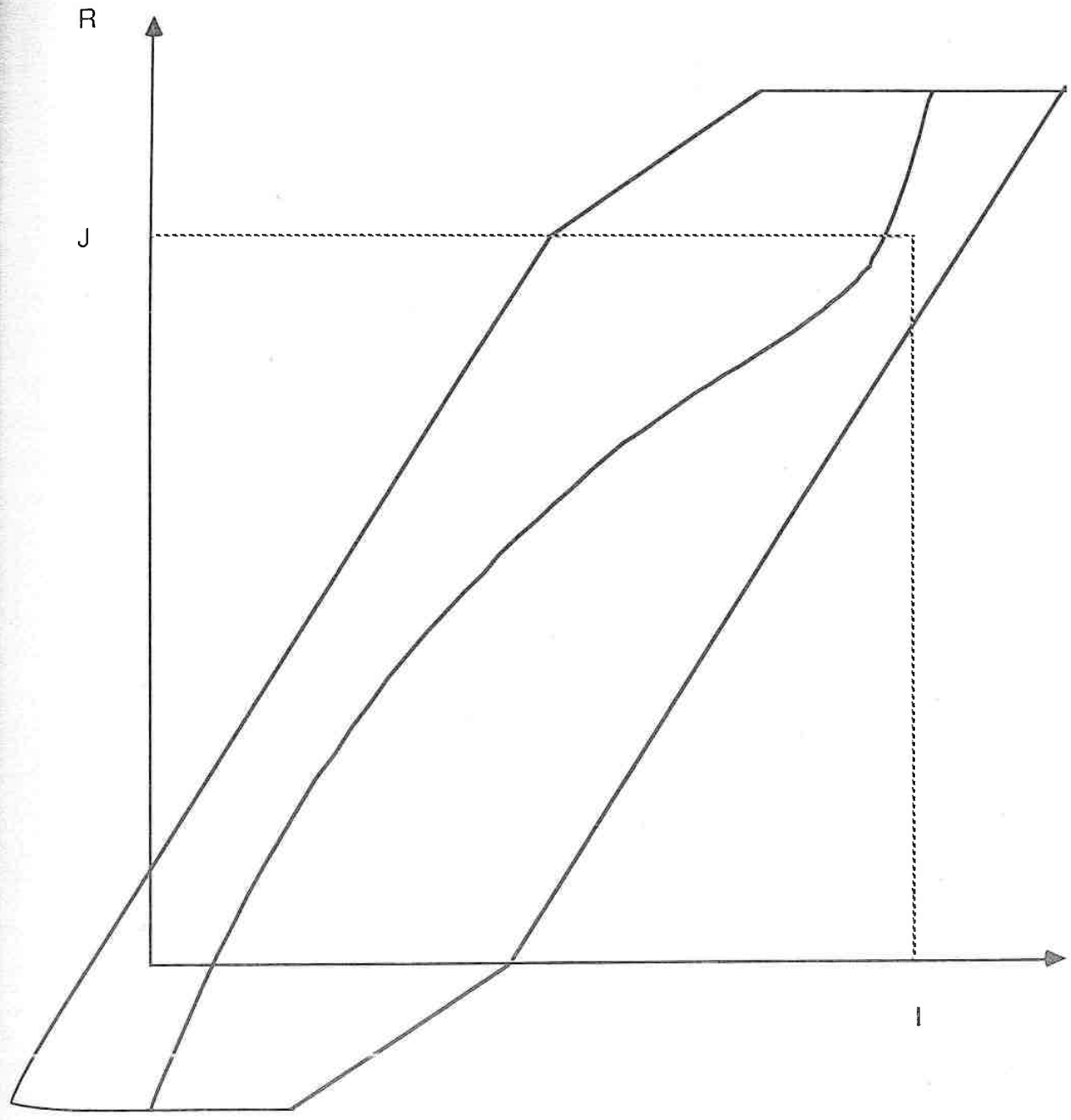


figure 6: généralisation de l'algorithme à point ajusté.

CHAPITRE 4: CONSTITUTION DU CORPUS ET RESULTATS

Quelques unes des méthodes précédentes ont été testées sur un corpus réel.

Nous avons implanté sur un MASSCOMP en langage C les algorithmes suivants:

- algorithme de programmation dynamique simple.

- algorithme de programmation dynamique avec la limitation du domaine de recherche d'aprés Sakoe et Chiba. Plusieurs valeurs du paramètre Rt qui définit la taille de la fenêtre d'exploration ont été essayées.

- algorithme à optimums locaux avec pavés "unicolonne" évolutifs. La taille de la colonne a été fixée arbitrairement à 7.

- algorithme avec relâchement aux contraintes suivant les deux axes horizontal et vertical. Deux types de zone de flou ont été essayés:

  1) zone de flou de taille fixe égale à la longueur de la forme divisée par un coefficient arbitraire. Plusieurs valeurs de ce paramètre ont été testées.

  2) la dimension de la zone de flou est déterminée pour chaque forme par une procédure qui calcule le nombre de prélèvements de type "silence". Dans ce but, deux stratégies ont été implantées: un vecteur est considéré comme du silence si au moins une de ses composantes est supérieure à un seuil donné, ou alors si la moyenne de ses composantes est supérieure à un certain seuil.

Dans tous ces algorithmes, la contrainte utilisée est la contrainte la plus simple sans condition sur la pente avec une pondération symétrique (figure 1).

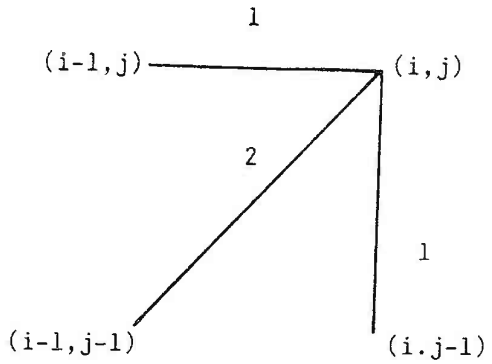


figure 1: contrainte sans condition de pente.

Avant de donner les résultats obtenus, voyons comment le corpus de tests a été obtenu.

1. CORPUS DE TESTS

L'apprentissage est une opération d'importance capitale pour un système de reconnaissance. En effet, le soin apporté à cette opération conditionne les performances du système, la puissance des méthodes utilisées ne peut pas compenser un apprentissage incorrect.

Notre corpus est constitué de répétitions des dix chiffres (0,1,...,9) du français prononcés par deux locuteurs, (un masculin et un féminin). Chaque locuteur a répété cinquante fois chaque chiffre.

En monolocuteur, une référence par mot a été sélectionnée et toutes les autres occurences du corpus pour ce locuteur ont été comparées aux 10 références. Il s'agit d'un locuteur masculin.

Un deuxième type d'essais a été effectué en prenant deux références par locuteur (une pour le locuteur masculin, une pour le locuteur féminin) et toutes les autres répétitions des mots dans le corpus ont servi de formes tests.

2. RESULTATS

2.1. Algorithme de programmation dynamique simple

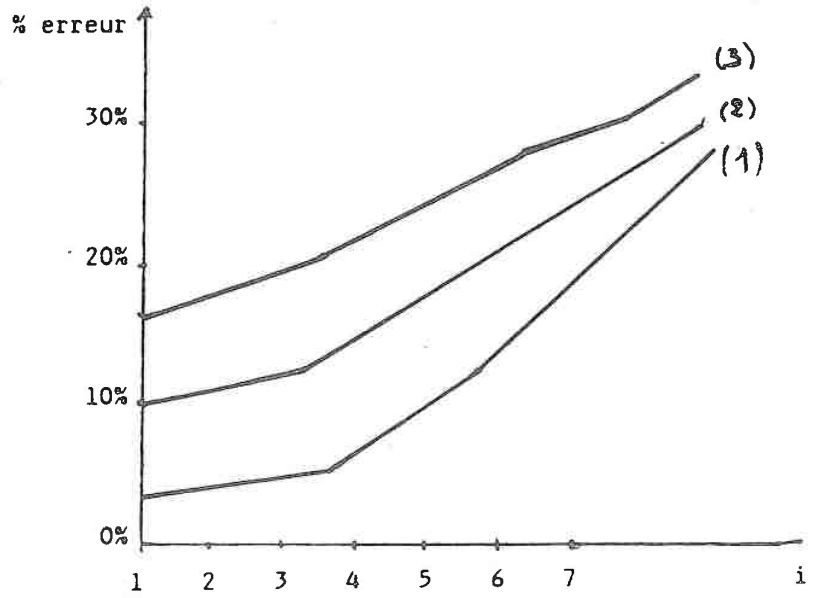
- en monolocuteur, pour 500 formes testées, le taux de bonne reconnaissance obtenu est de 97,1%.
- pour deux locuteurs, pour 1000 formes testées, le score de reconnaissance est de 89,3%.

Nous pouvons constater que le score de bonne reconnaissance chute brutalement lorsque l'on passe de un à deux locuteurs (8,2% d'erreur supplémentaires). Le résultat confirme que la programmation dynamique voit ses performances diminuer lorsque le nombre de locuteurs augmente. La chaine d'acquisition utilisée explique en partie les 2,9% d'erreur obtenus en monolocuteur.

2.2. Limitation de l'espace de recherche

La figure 2 donne le taux d'erreurs en fonction de la dimension de l'espace de recherche.

Le paramètre i indique que la limitation de l'espace de recherche est: Min(I,J)/i où I et J sont les longueurs en prélèvements des deux formes.



-monolocuteur masculin (1)
-monolocuteur féminin (2)
-mixte (3)

figure 2: taux d'erreur en fonction de la taille de l'espace de recherche.

Nous pouvons tout de suite remarquer que le locuteur masculin obtient un taux d'erreur bien inférieur au locuteur féminin. Le taux d'erreur augmente lorsque i augmente : pour i compris entre 1 et 4,

l'accroissement est faible (et ceci pour les deux locuteurs), mais dès que i devient supérieur à 4 la pente est beaucoup plus élevée, ce qui s'explique par le fait que l'espace de recherche est trop petit et ne contient pas toujours le chemin de recalage optimal.

2.3. Algorithme à optimums locaux (pavés unicolonnes évolutifs)

En monolocuteur (locuteur masculin), le taux d'erreur obtenu est de 10,8% au lieu des 2,9% obtenus avec l'algorithme de programmation dynamique simple. Cette augmentation de 7,9% est en partie due au fait que l'algorithme à optima locaux privilégie les chemins de recalage voisins de la diagonale.

2.4. Algorithme avec relâchement des contraintes

La taille de la zone de flou est égale à la longueur de la forme divisée par un coefficient arbitraire C.

La figure 3 donne le taux d'erreur en fonction de C en monolocuteur.

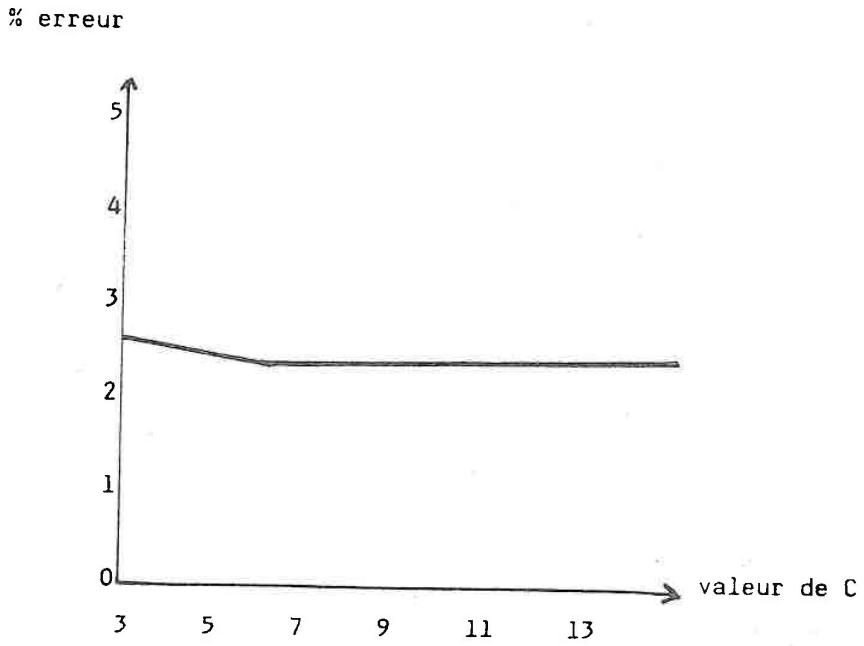


figure 3: taux d'erreur en fonction de C.

Le taux de reconnaissance est relativement constant pour $3 \leq C \leq 20$, le taux de reconnaissance est de 97,4%, ensuite le score est de 97,1% pour des valeurs de C supérieures à 20.

Pour deux locuteurs, le taux de reconnaissance est de 90,9%.

- la zone de flou est cette fois déterminée par la zone de vecteurs dont toutes les composantes sont inférieures à un seuil donné. Différents seuils ont été testés (allant du silence complet à un faible bruit).

Le taux de reconnaissance est également constant et égal à 97,3% tant que le seuil et supérieur à 5 et chute ensuite à 97,1%.

La figure 4 donne le taux d'erreur en fonction du seuil S exprimé en % de la valeur maximale d'une composante.

- la zone de flou est la zone de vecteurs dont la moyenne des composantes est inférieure à un certain seuil. Les mêmes valeurs de seuil que précédemment ont été testées. Le taux de reconnaissance est également constant égal à 97,3 % tant que le seuil est supérieur à 3 et tombe ensuite à 97,1%.

La figure 5 donne le taux d'erreur en fonction du seuil S exprimé en % de la valeur maximale d'une composante.

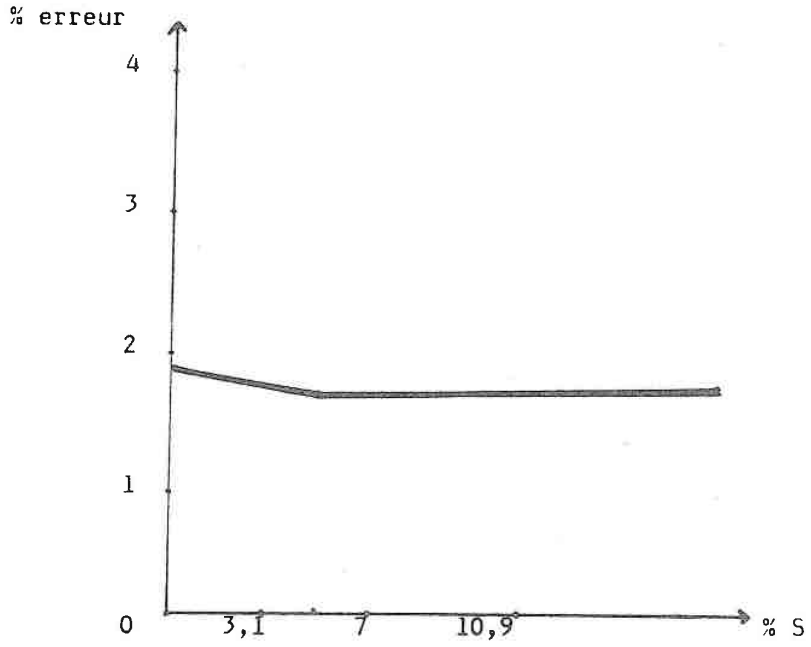


figure 4: taux d'erreur en fonction de S.

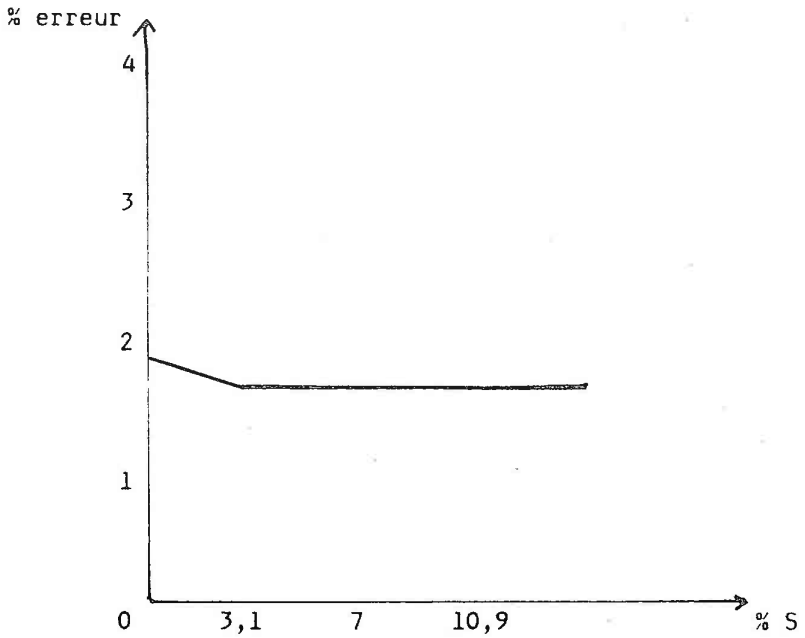


figure 5: taux d'erreur en fonction de S.

L'annexe 2 indique les principales erreurs rencontrées.

Tous ces résultats montrent que sur le vocabulaire des chiffres français prononcés isolément, le relâchement des contraintes sur les deux axes ne conduit qu'à une faible amélioration des performances de l'algorithme de programmation dynamique, ceci s'explique en partie par le fait que le corpus a été correctement segmenté et ne nécessite pas l'utilisation des zones de flou (leur rôle est essentiellement de pallier les erreurs de détection parole non parole).

CONCLUSION

Le tableau ci-dessous résume les résultats obtenus par les différents algorithmes testés (chapitre 4, partie B) pour le locuteur masculin.

méthode de programmation dynamique	simple	réduction de l'espace C=2	optima locaux	zones de flou
% de bonne reconnaissance	97,1	96,1	89,8	97,4

Nous constatons que le relâchement des contraintes sur les deux axes améliore légèrement le score de reconnaissance alors que l'algorithme à optima locaux dégrade nettement le taux de reconnaissance. Par contre, la réduction de l'espace de comparaison (c=2) ne diminue que très légèrement le taux de bonne reconnaissance.

Maintenant, comparons ces algorithmes du point de vue de leur temps d'exécution. Pour cela, nous allons déterminer le nombre de fois où les relations récursives de programmation dynamique sont calculées:

$$(\hat{i},\hat{j}) = \underset{(i',j')\in V(i,j)}{\operatorname{argmin}} \frac{D(i',j') + dp((i',j'),(i,j))}{L(i',j') + lp((i',j'),(i,j))}$$

$$D(i,j) = D(\hat{i},\hat{j}) + dp((\hat{i},\hat{j}),(i,j))$$
$$L(i,j) = L(\hat{i},\hat{j}) + lp((\hat{i},\hat{j}),(i,j))$$

Le tableau suivant indique le nombre de fois où les relations de programmation dynamique sont déterminées lors de la comparaison d'un mot de longueur I et d'un autre de longueur J. Il précise également sa valeur estimée dans le cas moyen I=J=60.

méthode utilisée	simple	C = 2	optima locaux	zones de flou
nombre d'appels	I.J	$\frac{\min(I,J).I}{2}$	7.I	I.J
estimation I=J=60	3600	1800	420	3600

Nous remarquons tout de suite que l'algorithme à optima locaux appelle beaucoup moins ces équations, mais ce gain de temps (8 fois plus rapide que la programmation dynamique simple) se réalise au détriment des performances. L'algorithme de réduction de l'espace de comparaison (C=2) est environ 2 fois plus rapide que la programmation dynamique simple et ne perd que 1% environ de performance: il réalise un bon compromis temps de calcul performance. L'algorithme avec relâchement des contraintes met environ le même temps d'exécution que l'algorithme de programmation dynamique simple et en améliore légèrement les performances: il semble donc plus efficace et mieux adapté à la reconnaissance de mots isolés.

PARTIE C

QUANTIFICATION VECTORIELLE EN RECONNAISSANCE DE MOTS ISOLES

INTRODUCTION

La plupart des systèmes de reconnaissance de parole qui existent à l'heure actuelle fonctionnent en mémorisant des formes de référence. L'occurence inconnue fournit une nouvelle information qui doit être comparée aux données préalablement stockées.

Imaginons que l'on veuille étendre le vocabulaire de l'application: le dictionnaire devra contenir l'image acoustique de tous les mots utilisés. Pour peu que le lexique soit important, on se trouve vite confronté à un problème de place mémoire que la quantification vectorielle peut aider à résoudre.

Un problème similaire se rencontre également lorsque l'on veut rendre un système indépendant du locuteur. Il faut alors mémoriser un ensemble de données représentatives des différentes prononciations possibles d'une population. Or si celle-ci est trop importante, il est impossible de stocker les caractéristiques de la voix de tous les locuteurs. On est donc conduit à sélectionner un certain nombre de locuteurs représentatifs des différentes façon de parler de la population. Après leur avoir fait prononcer une (ou plusieurs fois) l'ensemble des locutions du vocabulaire, on doit extraire et stocker suffisamment d'informations pertinentes pour discriminer les différents mots, quelque soit la personne qui les dit.

Là encore se pose le problème de la place mémoire occupée: la quantification vectorielle est à même de réduire l'espace nécessaire au stockage des données. C'est pourquoi cette technique de codage est utilisée de plus en plus souvent comme première étape dans un système de reconnaissance.

Comme nous l'avons déjà vu dans un chapitre précédent, tous les systèmes de reconnaissance de la parole utilisent déjà des techniques de codage qui permettent de diminuer la quantité d'informations sans perdre les caractéristiques importantes de la parole. C'est ce que nous avons appelé paramétrisation. Ainsi, on se contente souvent de mémoriser des coefficients (FFT, LPC, Cepstre.....) qui caractérisent le spectre à

court terme du signal. Un mot du dictionnaire est alors représenté par une succession de vecteurs de coefficients. Cependant la compression ainsi obtenue n'est pas suffisante pour faire par exemple de la reconnaissance de la parole indépendante du locuteur, car on ne peut pas garder tous les mots du vocabulaire prononcés par tous les locuteurs sélectionnés. On va alors tenir compte du fait qu'il existe une certaine ressemblance entre les références d'un même mot prononcé par plusieurs personnes, sans oublier que l'on n'est pas obligé de garder toute l'information du signal de parole mais que seuls les éléments caractérisant les différences entre les mots nous intéressent.

En regroupant les informations de parole en classes, et en ne gardant qu'un représentant par classe, on peut encore réduire la quantité de données à mémoriser tout en s'affranchissant dans une certaine mesure des variations inter-locuteur.

Après avoir étudié quelques techniques permettant de réaliser cette classification, nous allons l'appliquer à la reconnaissance des chiffres isolés.

CHAPITRE 1: METHODES DE QUANTIFICATION VECTORIELLE

([Adoul 85])

La quantification vectorielle de la parole a donc pour but de coder le spectre du signal sur des fenêtres temporelles. Elle nécessite une base de données de parole suffisamment importante pour contenir toutes les informations de parole que l'on souhaite mémoriser. Sur chaque fenêtre temporelle, on calcule un vecteur de coefficients qui constituent une représentation du spectre du signal. Les vecteurs sont ensuite regroupés en classes, et de chaque classe, on extrait un ou plusieurs représentants.

Deux stratégies peuvent être retenues lors de la création de classes:

- on se fixe a priori le nombre de classes (ex: Algorithme de Linde-Buzo-Gray)
- on impose une distorsion maximale sur l'ensemble d'apprentissage (ex: Algorithme à seuil)

1. ALGORITHME DE LINDE-BUZO-GRAY [1980]

Il s'agit donc de trouver un vecteur de paramètres p(1)...p(I) à l'intérieur d'un ensemble qui minimise une mesure de distorsion.

Linde, Buzo et Gray ont utilisé une analyse de type LPC et la distorsion est alors la distance d'Itakura qui a été explicitée dans la partie A.

Elle présente deux avantages principaux:

- elle est facilement calculable,

- elle a un sens en perception.

De plus, elle est toujours positive et si elle n'est pas symétrique, elle vérifie néanmoins l'inégalité triangulaire.

Le quantifieur que ces auteurs ont proposé ([Llyod 82]) résoud de manière élégante le problème non trivial suivant: comment déterminer un ensemble de références ou codebook. Un codebook est un ensemble fini de vecteurs prototypes référence.

La méthode proposée pour créer un ensemble de vecteurs référence qui minimise une distorsion sur un grand nombre de données est itérative. Comme beaucoup de problèmes de minimisation, elle converge vers un minimum local et non pas obligatoirement vers un minimum global.

Le but est donc, à partir d'un ensemble de vecteurs issus de la paramétrisation, de trouver un ensemble de B vecteurs prototypes qui les représentent le mieux.

A l'étape d'initialisation, B vecteurs sont choisis arbitrairement. Comme première étape, chaque vecteur de données est affecté par la méthode du plus proche voisin à la classe dont le représentant est l'un des B vecteurs choisis précédemment.

La distorsion totale est déterminée en sommant la distorsion de chaque vecteur de la base de données lors de la répartition en classes.

La deuxième étape se propose d'améliorer le codebook en choisissant un meilleur représentant pour chaque classe qui sera par exemple le centroide de la classe évaluée à l'étape 2. Par définition du plus proche voisin, la distorsion de chaque vecteur de données, et par conséquent la distorsion globale, diminue ou, au pire, reste constante.

Cet algorithme conduit donc à une distorsion globale monotone décroissante qui converge donc vers un optimum local.

La vitesse de convergence dépend du choix des B vecteurs initiaux.

Algorithme 1:

1)Initialisation

On se donne un dictionnaire Yk de taille B.
On pose k = 0.

2)Construction de la partition

On possède un dictionnaire Yk = {Yik}i = 1,K aprés k étapes.
On cherche la partition composée des classes Sik qui minimise l'erreur de quantification associée à Yk,

$$x \in Sik \Leftrightarrow d(X,Y) \geq d(X,Yjk)$$

pour j=1,..,K

l'erreur de quantification vaut:

$$D_k = E(\min_i d(X,Yik))$$

3) Test d'arrêt

Si (par exemple): $(D_{k-1}-D_k)/D_k<e$ (e fixé), on s'arrête
Sinon aller en 4.

4) Recalcul des centroides

A chaque classe Sik de la partition, on associe le centroide Yik+1 qui est, par exemple en distance euclidienne, le centre de gravité de cette classe. Le dictionnaire est désormais composé des nouveaux Yik. On fait k = k+1
Aller en 2.

Une variante de cet algorithme 1 a également été proposée par Linde, Buzo et Gray.

A la première étape, le codebook ne contient que deux vecteurs qui servent à partager l'ensemble des données de la même manière que précédemment, en deux classes. Chacune de ces classes est ensuite séparée en deux par la même méthode, et ainsi de suite jusqu'à obtenir les $B = 2^b$ prototypes désirés, un prototype étant le centroide d'une classe.

Une manière de choisir les deux premiers prototypes peut être par exemple de calculer le centroide de l'ensemble des données et de le perturber légèrement de manière à obtenir deux vecteurs.

si C=(c1,...,cp) est le centroide global
les deux prototypes initiaux peuvent être:

$$C1 = \begin{bmatrix} 1{,}01\ c1 \\ \\ 1{,}01\ cp \end{bmatrix} \qquad C2 = \begin{bmatrix} 0{,}99\ c1 \\ \\ 0{,}99\ cp \end{bmatrix}$$

Supposons que l'on désire construire un dictionnaire de 2^{k0} vecteurs et que la distance utilisée est la distance euclidienne [Miclet 85]:

Algorithme 2:

1)Initialisation

Choix du centre de gravité de l'ensemble d'apprentissage. Soit Yo ce vecteur.
Le dictionnaire est constitué de Yo. Faire k = 0.

2)Eclatement:

Tous les éléments y (en nombre 2^k) du dictionnaire sont "éclatés" en deux vecteurs. Ceci se fait par exemple en transformant chaque y en $y+\varepsilon$ et $y-\varepsilon$, où y est un vecteur donné de norme faible.

3)Convergence:

On applique l'algorithme 1 de Linde-Buzo-Gray présenté précédemment avec les 2^{k+1} éléments déterminés en 2, comme dictionnaire de départ. On récupère aprés convergence un codebook de 2^{k+1} vecteurs.

4) Arrêt:

On fait k=k+1. Si k>kO fixé à l'avance,arrêt
Sinon aller en 2.

2. L'ALGORITHME A SEUIL

Nous venons de voir des solutions "classiques" au problème de la construction d'un codebook.

La solution que nous avons retenue est différente: elle utilise un algorithme simple et rapide, l'algorithme à seuil [Tou 74], qui ne donne pas un nombre fixé de classes mais garantit une distorsion maximale.

Soit d la distance utilisée.

Soit {x1,..,xN} l'ensemble des vecteurs de la base de données. Ils sont introduits dans cet ordre.

Le premier vecteur x1 forme à lui seul une classe dont il est le représentant. Ensuite, au cours du déroulement de la classification, l'introduction du vecteur xi aboutira à la situation suivante:

- si toutes les distances d(xi,Cj) entre xi et les classes Cj déjà créées (c'est à dire entre xi et le représentant de Cj) sont supérieures à un seuil donné S, on crée une nouvelle classe dont xi est le seul élément et le représentant.

- sinon on affecte xi à la classe la plus proche et on réactualise éventuellement son représentant.

Algorithme 3

- C1 = {x1} P1 = x1 k = 1
- pour i de 2 à N faire

  j = Argmin d(xi,Ct)
  $1 \leq t \leq k$

  si d(xi,Cj) = d(xi,Pj) > S alors k = k+1
  Ck={xi}
  Pk=xi

  sinon ajouter xi à Cj
  recalculer Pj

fin

où

- Pi est le représentant de la classe Ci,

- N est le nombre de points de l'espace d'apprentissage et S la valeur du seuil.

Le nombre de classes est la valeur de k rendue à la fin de l'exécution.

Cet algorithme est extrêmement simple mais il présente quelques inconvénients:

- il est sensible à l'ordre d'arrivée des données.

- la distance et le seuil jouent un rôle primordial.

- le nombre de classes n'est pas contrôlé avec précision puisqu'il dépend du seuil.

Par contre, il a les avantages suivants:

- il est simple et permet de traiter des masses de données importantes avec des temps de calcul raisonnables.

- le fait que les données sont ordonnées présentent même des avantages puisqu'il est alors possible d'interrompre l'apprentissage, d'obtenir un dictionnaire, de l'utiliser et de l'agrandir si de nouvelles données se présentent, au contraire de l'algorithme de Linde-Buzo-Gray qui nécessite la reprise complète de l'apprentissage.

- expérimentalement, il est facile d'évaluer grossièrement le nombre de classes en fonction du seuil.

- il est rapide puisque chaque échantillon n'est considéré qu'une seule fois (au contraire de la méthode de Linde-Buzo-Gray).

Il nous a paru intéressant à utiliser puisque nous envisageons de réaliser par la suite une application multilocuteur, nous pouvons

augmenter le dictionnaire de vecteurs de référence chaque fois qu'un locuteur est mal reconnu sans avoir pour autant à recommencer tout l'apprentissage.

Nous avons indiqué que le représentant d'une classe peut être réactualisé ou non; en effet, deux stratégies se présentent:

- le représentant d'une classe est l'élément qui a servi à la créer. Cette technique garantit une distorsion maximale mais peut aboutir à des classes dispersées.

- le représentant est réactualisé à chaque nouveau vecteur ajouté à la classe. Ce représentant peut être par exemple le centre de gravité de la classe. On risque alors de voir se former des classes allongées, mais cet effet de chaine peut être limité par l'utilisation de seuil relativement petit. De plus, le prototype du codebook sera un vecteur "fictif" puisque créé en calculant une moyenne.

3. EXPERIMENTATION

Nous avons choisi l'algorithme à seuil pour sa rapidité, sa simplicité et surtout parce que, désirant réaliser une application multilocuteur, il nous a paru indispensable de pourvoir augmenter le dictionnaire chaque fois qu'un locuteur mal reconnu se présente.

Nous avons alors, à l'aide de notre algorithme de classification, déterminé un codebook par mot du vocabulaire, donc un dictionnaire par chiffre. Pour chaque codebook, la séquence d'apprentissage était constituée de vingt cinq répétitions du mot considéré par deux locuteurs (1 féminin, 1 masculin) et le seuil est identique pour tous les codebooks.

Le nombre de classes par locution est donnée figure 1.

mot	0	1	2	3	4	5	6	7	8	9
nbre	31	16	17	28	25	19	11	19	12	15

figure 1: Nombre de prototypes par codebook.

Les codebooks seront utilisés dans les chapitres suivants.

Nous constatons que le nombre de classes varie de 11 à 31, "11" correspond au mot six, "31" à zéro.

CHAPITRE 2: METHODE DE BURTON

([Burton 83], [Tassy 85])

Il s'agit maintenant de reconnaître des mots prononcés isolément après qu'ils aient été codés à l'aide des classes obtenues par quantification vectorielle.

Supposons que le vocabulaire de l'application considérée soit constitué de N mots distincts que nous noterons dorénavant $M_1 \ldots M_N$.

A chaque mot Mi du dictionnaire ($1 \leq i \leq N$) correspond un codebook CBi déterminé à l'aide d'une quelconque méthode de quantification vectorielle.

Chaque codebook CBi est constitué de ni classes, chaque classe étant représentée par un vecteur prototype v_j^i (où j varie de 1 à n_i).

1. PRESENTATION DE LA METHODE

Soit T = t(1).....t(I) un mot inconnu. Pour l'identifier, Burton a proposé la méthode suivante:

- tout d'abord,la forme T est codée à l'aide de chaque codebook CBi, c'est à dire qu'à l'issue de cette première étape,on dispose de N codages du mot inconnu.

Le codage de l'occurence T à l'aide du codebook CBi est réalisé de la façon suivante: pour chaque vecteur t(k) de T($1 \leq k \leq I$), on recherche le prototype t(i,k) de CBi qui lui est le plus proche au sens de la distance utilisée (distance de Hamming dans notre cas):

$$t(i,k) = \text{ARGMIN}_{1 \leq j \leq ni} d(t(k), v_j^i)$$

Le prélèvement t(k) est alors remplacé par l'indice j du prototype t(i,k).

T est alors codée comme une suite d'entiers, un entier correspondant à l'indice du prototype qui représente le mieux le vecteur considéré.

- en même temps que l'on code la forme T avec le codebook CBi, on va calculer une mesure de distorsion DT(i) entre la forme test et CBi. Cette distorsion rend compte en quelque sorte de la distance entre la forme réelle T et la forme codée.

  DT(i) est calculé à l'aide de la formule:

$$DT(i) = \sum_{j=1}^{I} d(t(j), t(i,j)).$$

- les distorsions DT(i) ayant été calculées pour chaque valeur de i ($1 \leq i \leq N$), il s'agit maintenant de déterminer quel mot Mc du vocabulaire coincide le mieux avec la forme inconnue T. Nous dirons que le mot identifié est Mc si le codebook CBc qui a été obtenu à partir de Mc réalise la plus petite valeur de distorsion, c'est à dire si c = Argmin DT(i).

Cette méthode a l'avantage d'être simple et rapide. Nous l'avons testée sur le corpus décrit au chapitre 4 de la partie B.

2. RESULTATS EXPERIMENTAUX

Pour les deux locuteurs, les tests ont été effectués sur les 500 mots (250 par locuteur) qui n'ont pas servi à l'apprentissage, le score de reconnaissance est de 83,1%.

Ce score de reconnaissance est relativement médiocre, ceci peut en partie s'expliquer par le fait que nous n'avons pas tenu compte d'informations temporelles.

Toutefois, nous avons pu constater les faits suivants:

- il est possible de supprimer toutes les classes à un élément sans altérer le taux de reconnaissance.En effet,ces classes correspondent en quelque sorte à des "accidents " de prononciation, par exemple à un râclement de gorge,une toux,etc...

- la bonne réponse figure toujours dans les trois mots qui réalisent les trois distorsions minimales.

D'où l'idée d'utiliser la méthode suivante:

- le mot inconnu est codé par rapport à chaque codebook et sa distorsion par rapport à chaque CBi est déterminée par la méthode de Burton.

- la décision de reconnaissance est prise si l'écart entre la valeur de distorsion minimale et les autres distorsions est supérieure à un certain seuil fixé arbitrairement. C'est en quelque sorte un critère de confiance. Avec la valeur du seuil que nous avons utilisé, la décision de reconnaissance est prise dans environ 20% des cas.

- si le critère de confiance n'est pas satisfait, afin de tenir compte d'informations temporelles, nous comparons par programmation dynamique le mot inconnu et les trois références correspondant aux trois codebooks donnant les distorsions les plus faibles.

([Bonneau 85])

Nous disposons donc d'un dictionnaire de vecteurs CBi par mot du vocabulaire considéré.

Par la méthode de Burton, nous avons sélectionné les trois meilleurs candidats à la reconnaissance (par minimisation de la mesure de distorsion).

L'occurrence inconnue T va être comparée aux trois références correspondant à ces trois mots candidats par programmation dynamique.

Pour cela, il va falloir constituer un dictionnaire des mots de référence. Chaque référence Ri est codée comme une suite de numéros faisant référence aux vecteurs du codebook CBi correspondant.

1. CHOIX DES FORMES DE REFERENCE

Nous avons décidé de garder deux formes de référence par mot du vocabulaire. Pour sélectionner les "meilleures références", nous avons utilisé deux techniques de choix différentes ([Divoux 85]):

1.1. Procédure Minisomme

Il s'agit de déterminer les deux formes les plus représentatives d'un mot Mi du dictionnaire. Nous disposons des nb=50 répétitions de Mi qui ont servi à élaborer le codebook CBi. Chacune de ces élocutions est

codée comme une suite de vecteurs de CBi et est ensuite comparée à toutes les autres par programmation dynamique (algorithme le plus simple, avec la contrainte locale sans condition de pente et avec une pondération symétrique). Dans l'algorithme de programmation dynamique, il est inutile de recalculer toutes les distances de vecteurs: puisque les vecteurs sont connus à l'avance, une matrice des distances (distance de Hamming dans notre cas) est précalculée et l'algorithme y lira les distances dont il a besoin.

Pour chaque forme j, nous calculons la valeur:

$$D(j) = \sum_{k=1}^{nb} D(j,k)$$

où D(j,k) est le taux de dissemblance entre les formes j et k

Les deux références R1 et R2 seront les deux formes qui minimisent D(j).

$$R1 = \underset{1 \leq j \leq nb}{\text{Argmin}}\ D(j)$$

$$R2 = \underset{1 \leq j \leq nb,\, j \neq R1}{\text{Argmin}}\ D(j)$$

1.2. Procédure Minimax

Tous les scores D(j,k) entre toutes les répétitions j et k d'un même mot Mi ayant été calculés par programmation dynamique de la même manière que précédemment, nous choisissons les deux références R1 et R2 par:

$$R1 = \underset{1 \leq j \leq nb}{\text{Argmin}}\ \max_{1 \leq k \leq nb} D(j,k)$$

$$R2 = \underset{1 \leq j \leq nb,\, j \neq nb}{\text{Argmin}} \; \max_{1 \leq k \leq nb} D(j,k)$$

Les références sont maintenant choisies pour chaque mot par l'une ou l'autre méthode, la reconnaissance peut commencer.

2. RECONNAISSANCE PAR PROGRAMMATION DYNAMIQUE

Le mot inconnu est comparé par programmation dynamique à six références (deux références par mot, trois mots choisis d'aprés la méthode de Burton). Le mot de référence le plus proche du mot inconnu au sens de la distance de programmation dynamique sera désigné comme le mot reconnu.

L'algorithme de programmation dynamique que nous avons utilisé est celui qui est décrit au chapitre 1 de la partie B. Aucun relâchement des contraintes sur les deux axes n'est effectué.

Le corpus de tests est constitué des cinq cents mots n'ayant pas participé à l'apprentissage.

3. RESULTATS EXPERIMENTAUX

Nous avons obtenu les scores de reconnaissance suivant:

- lorsque les références ont été déterminées par la procédure "min-isomme", le taux de bonne reconnaissance est de 84,3 %

- lorsque les références ont été déterminées par la procédure "minimax", le taux de bonne reconnaissance est de 85,4 %.

Nous constatons donc que les résultats sont meilleurs lorsque les formes de référence sont sélectionnées par la procédure "Minimax".

Désormais, tous les résultats seront donnés dans le cas où le dictionnaire des références a été établi en utilisant "Minimax".

Le taux de reconnaissance (85,4%) constitue une amélioration par rapport à la méthode de Burton (83,1%), ceci s'explique aisément par le fait que nous avons tenu compte de l'alignement temporel. Malgré tout, le résultat est encore insuffisant, et ceci en partie parce que nous n'avons pas tenu compte de la distorsion introduite au moment du codage. En effet, il se peut très bien que la classe "la plus proche" d'un vecteur soit malgré tout très éloignée de ce vecteur. Le chapitre suivant exposera une façon de tenir compte à la fois de la distorsion de codage et de l'alignement temporel.

4. CONCLUSION

A titre de remarque, calculons la place mémoire nécéssaire au stockage des données:

Soient: -N :le nombre de mots du vocabulaire.
-T :le nombre moyen de vecteurs par mot.
-QV:le nombre total de vecteurs de tous les codebooks.
-D :la dimension des vecteurs.

La place mémoire occupée est:

-sans codage vectoriel:

$MO1 = N*T*D$

-avec codage vectoriel:

$MO2 = N*T + QV*D$

Par exemple dans notre cas:

$N = 20$, $T = 50$, $QV = 193$, $D = 16$

$MO1 = 16000$, $MO2 = 4088$

soit un gain $\frac{MO1}{MO2} = 4$

Par quantification vectorielle, nous avons diminué par quatre la place mémoire nécessaire au stockage des données.

CHAPITRE 4: DISTORSION ET PROGRAMMATION DYNAMIQUE

La méthode est la même que précédemment (détermination des trois meilleurs candidats par minimisation de la distorsion de codage et programmation dynamique si le critère de confiance n'est pas satisfait), mais la décision est prise non pas en minimisant le taux $D_{DTW}(j)$ de dissemblance de la forme inconnue avec la référence Rj mais en minimisant:

$$D_T(j) = (1-al) \cdot D_{TW}(j) + al \cdot D(j)$$

où:

- al est un coefficient réel compris entre 0 et 1.
- D(j) la valeur de distorsion du codage de T avec le codebook correspondant à la référence j.

Le mot reconnu est celui qui réalise:

$$\min_j \; D_T(j)$$

Les tests ont été réalisés sur le même corpus que précédemment pour les valeurs de al 0,1 0,2.....0,9 1. Les résultats sont indiqués sur la courbe de la figure 1 qui donne le taux d'erreurs en fonction de la valeur du paramètre al.

Nous constatons que al = 0,5 donne le taux d'erreurs le plus faible (4,8%). C'est donc en tenant compte de la même façon de l'alignement temporel et de la distorsion de codage que nous obtenons la meilleure reconnaissance (95,2% de bons résultats).

Une autre amélioration a également été constatée lorsque pour calculer la distorsion de codage, nous avons procédé de la façon suivante:

- Soit S le seuil qui a été utilisé pour déterminer les classes par l'algorithme de quantification vectorielle à seuil.

- Soit $t^i(j)$ le prototype qui est le plus proche du vecteur $t(j)$ de la forme inconnue.

- Si $d(t^i(j),t(j)) \leq S$, alors la distorsion n'est pas modifiée, sinon lui ajouter $A \times d(t^i(j),t(j))$ (A est un paramètre qui doit être supérieur ou égal à deux d'après nos tests).

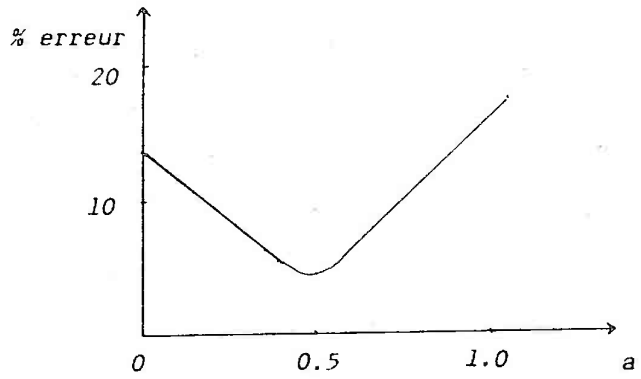


figure 1: taux d'erreur en fonction de a.

En bref, cela signifie que si le vecteur est suffisamment proche du prototype qui le code (de telle sorte que lors de la classification, il aurait été affecté à cette classe), nous considérons que le vecteur est "bien codé" et donc qu'il n'y a pas d'erreur de codage.

Au contraire, s'il est éloigné du prototype le plus proche, nous estimons que le codage est mauvais et nous pénalisons la distorsion.

En prenant la décision sur le score:

$$D_T(j) = 0,5 \cdot D_{DTW}(j) + 0,5 \cdot D(j)$$

où Dj est calculée en prenant A=4, nous obtenons un taux de reconnaissance de 96,5%.

En regardant les codages des formes, nous avons remarqué que nous avions fréquemment des séquences du type PjPj...Pj (c'est à dire que le prototype Pj apparait n fois consécutives). Une idée qui nous a paru a priori intéressante a été de réduire le codage des formes en appliquant la règle:

Pj...Pj Pj---> Pj

qui signifie que la succession de n Pj est remplacée par une seule occurrence de Pj. Par exemple, PjPjPjPjPj donne Pj.

Avec al = 0,5, nous avons obtenu un score de bonne reconnaissance de 80,5 %. Ainsi, ce mauvais score ne nous a pas autorisé à conserver pour la suite cette "réduction de codage" qui conduisait à un gain de temps non négligeable.

Nous avons alors essayé une compression plus "douce" en remplaçant une suite de deux Pj par un seul Pj. Par exemple, la succession PjPjPjPjPj done PjPjPj. Avec une telle méthode, le taux de bonne reconnaissance est moins altéré que précédemment et nous avons obtenu 88,25% de bonne reconnaissance. Ce pourcentage est moins mauvais que le précédent, seulement il est encore trop faible pour que nous puissions retenir cette simplification.

Une dernière série de tests a été effectuée à l'aide de l'algorithme de relâchement aux contraintes suivant les deux axes horizontal et vertical. La zone de flou a été déterminée par la méthode qui donne la taille de la zone de flou en divisant la longueur de la forme par un coefficient arbitraire r. Pour al=0.5 et r=3 ou 5, nous avons obtenu un taux de bonne reconnaissance de 96,3%. L'amélioration n'est pas significative, c'est ce que nous avons déjà constaté en reconnaissance de mots isolés monolocuteur sans quantification vectorielle.

CHAPITRE 5: PONDERATIONS VARIABLES

([Mari 85], [Moore 85], [Gauvain 85] [Tassy 86] [Boyer 86])

Une autre solution au problème de l'alignement temporel des formes acoustiques est obtenue en représentant chaque référence sous la forme d'un graphe où les noeuds représentent les états d'un processus de Markov (figure 1).

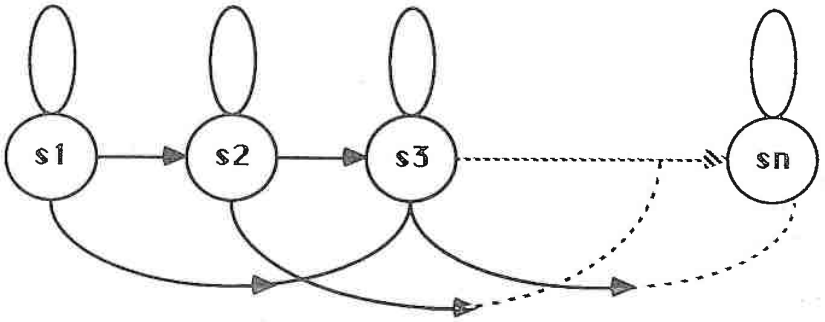
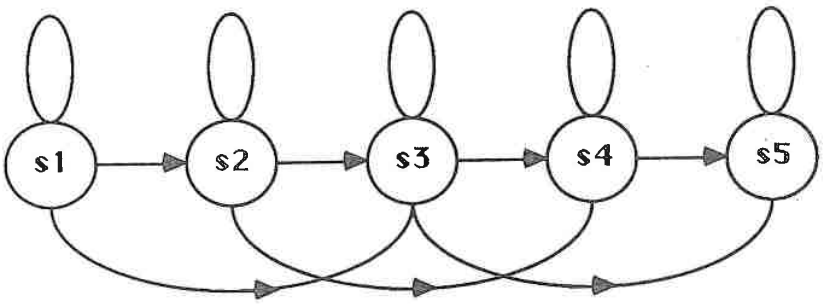


figure 1 : exemple de modèle de Markov.

Un tel graphe se compose d'un nombre fini d'états, en général 5 [Levinson 83] (figure 2). Les états représentent les segments stables du signal vocal alors que les variations spectrales sont modèlisées par des arcs de transition. Sur ce modèle, chaque état peut donner lieu à 3 transitions:

- retour sur lui-même,
- transition vers l'état suivant,
- saut de l'état suivant.

On peut considérer en quelque sorte ce graphe comme un modèle de production d'un mot où chaque transition s'accompagne de l'émission d'un vecteur de paramètres spectraux. Pour un même mot, plusieurs images acoustiques différentes peuvent être générées.



A chaque état est associée une liste commune de vecteurs spectraux pouvant apparaitre, ainsi qu'une distribution de probabilités associée.

figure 2: un modèle de Markov.

Notons $S_1, S_2, \ldots, S_J$ les J états d'un modèle de Markov (par exemple, J=5, figure 2). Une distribution de probabilités P(ai/sj), probabilité que l'évènement ai se produise sur une transition d'origine sj, est associée à chaque état. Une probabilité p(sj/si), probabilité que le modèle passe de l'état si à l'état sj en une seule transition, est associée à chaque arc.

Les paramètres du modèle, J, p(sj/si), sont obtenus au cours d'une phase d'apprentissage à partir d'un grand nombre d'énoncés du même mot.

Supposons que nous disposions de n occurrences d'un même mot, soit $O = O_1, \ldots, O_n$. Les paramètres du modèle S sont évalués en maximisant la probabilité de l'ensemble d'apprentissage $P(O/S) = \prod_{k=1}^{n} P(Ok/S)$, où P(Ok/S) est la probabilité que le modèle génère la forme Ok. Baum [Baum 72] a montré que l'on pouvait résoudre ce problème à l'aide d'un algorithme itératif: il suffit d'initialiser les paramètres par une estimation quelconque. A chaque itération, les P(ai/aj) et p(si/sj) sont estimées à nouveau et fournissent un modèle S' qui vérifie $P(O/S') \geq P(O/S)$. L'algorithme s'arrête lorsqu'il n'y a plus de changement

significatif.

P(ai/sj) peut se représenter de deux manières différentes:

- par une distribution gaussienne, mais elle présente deux inconvénients principaux: son évaluation peut nécessiter un grand nombre de calculs, une distribution gaussienne peut ne pas modéliser correctement la distribution des spectres (par exemple, dans le cas multilocuteur).

- par une liste de valeurs, l'ensemble des ai étant supposé fini. Le principal avantage de cette distribution discrète est qu'elle n'est pas limitée à une gaussienne. Lors de la reconnaissance, l'évaluation de p(ai/sj) consiste à lire une valeur dans une table, la taille de la table est réduite par l'utilisation de la quantification vectorielle.

Le problème de l'alignement temporel entre un modèle de Markov S et une forme inconnue T consiste à estimer la suite d'états du modèle $S_{n(1)}.....S_{n(k)} = S_T$ (n(1) = 1, n(k) = J) qui a engendré la séquence (t(1)....t(I)). Il s'agit donc de trouver la suite S_T qui maximise:

$$P(S/T) = \frac{P(T,S)}{P(T).}$$

P(T) ne dépend pas de S, il suffit donc de maximiser P(T,S).

Or P(T,S) est le produit des probabilités de transition P(Sn(i)/Sn(i-1)) et des probabilités d'émission P(aj/Sn(i)) le long du chemin décrit par S dans le graphe du modèle.

L'algorithme de Viterbi ([Forney 73] [Jelinek 76]), qui est un algorithme de programmation dynamique similaire à celui que nous avons étudié précédemment où les distances sont remplacées par des probabilités, permet de résoudre ce problème. L'équation récursive locale s'écrit :

$$p(i,j) = P(ai/sj) \cdot \text{Max} \begin{cases} p(i-1,j) \cdot P(sj/sj) \\ p(i-1,j) \cdot P(sj/sj-1) \\ p(i-1,j-1) \cdot P(sj/sj-2) \end{cases}$$

et alors P(T,S) = P(I,J).

Remarquons tout de suite que si nous appliquons la transformation f(x)=-logx aux deux membres de cette équation, les produits vont être transformés en somme et la maximisation en minimisation. Cette équation a alors une forme semblable à l'équation classique de la programmation dynamique et l'indice de dissemblance entre la forme de référence et la forme inconnue T est D(T,S)=-logP(I,J).

Poussons le parallèle un peu plus loin. Reprenons la contrainte locale d'Itakura (figure 3).

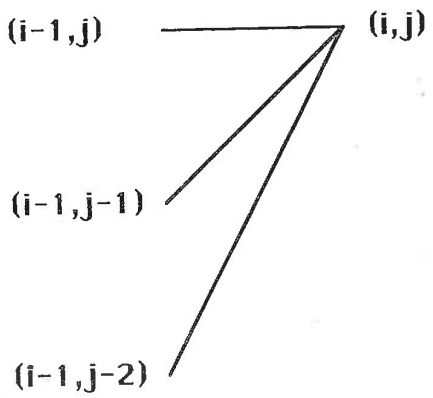
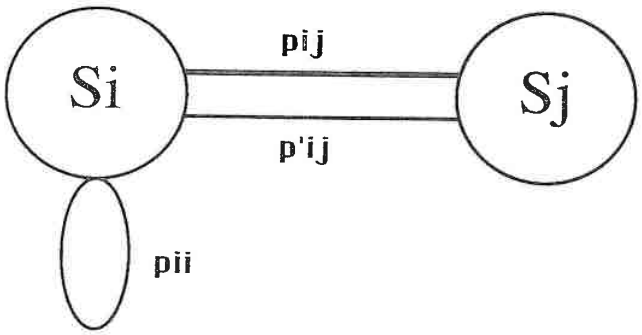


figure 3: contrainte d'Itakura.

Nous pouvons remarquer que les trois transitions (retour sur lui-même, transition vers l'état suivant, saut de l'état suivant) correspondent aux contraintes locales imposées par la contrainte d'Itakura à l'algorithme de programmation dynamique.

Considérons maintenant le cas général d'un modèle de Markov (figure 4).



pii:probabilité de rester dans l'état i.

pij:probabilité de passer de l'état i à l'état j avec émission d'un prélèvement.

p'ij:probabilité de passer de l'état i à l'état j sans émission d'un prélèvement.

figure 4: cas général de transition entre 2 états i et j.

Nous pouvons constater que pii correspond à la pondération a de la contrainte sans condition de pente (figure 5), pij à b, p'ij à c.

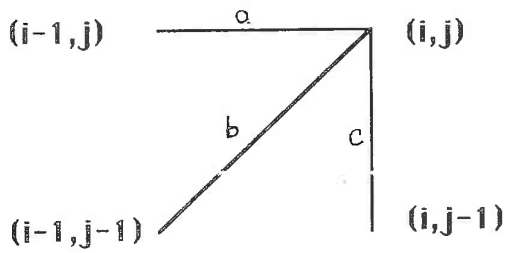


figure 5: contrainte la plus simple sans condition de pente.

Il est intéressant de constater que la transition de l'état i à l'état j de probabilité p'ij sans émission de prélèvement n'a guère été utilisée jusqu'à présent dans les modèles de Markov cachés car l'algorithme de Viterbi ne le permet pas.

Ainsi, les coefficients (a,b,c) peuvent être associés à un état d'un modèle de Markov - il y a dans ce cas autant d'états que de triplets- qui lui-même peut être associé à une liste de vecteurs prototypes. Cette dernière solution modèlise plus convenablement la structure temporelle des formes vocales, structure temporelle qui est perdue lorsqu'une liste de prototypes est associée à un état d'un modèle de Markov.

1. Triplets (a,b,c) par couple de prototypes d'un codebook

Nous disposons d'un ensemble de vecteurs prototypes par mot du vocabulaire.

La stratégie de reconnaissance est celle qui a été utilisée au chapitre 3 de la partie C:

- le mot inconnu est codé à l'aide de chaque codebook j séparément, la distorsion de codage Dc(j) est calculée.

- les trois mots candidats qui obtiennent les valeurs de Dc(j) les plus petites sont retenus. Si le critère de confiance est satisfait (la différence entre les deux scores les plus petits inférieure à un seuil donné), le mot de distorsion minimale est déclaré reconnu.

- sinon le mot inconnu est comparé par programmation dynamique aux trois références correspondant aux trois candidats retenus. La décision finale est prise par minimisation de la quantité Dc(j) + Ddtw(j) où Ddtw est le taux de dissemblance entre la forme inconnue

et la référence j déterminée par programmation dynamique.

Cette fois, les coefficients (a,b,c), qui sont en quelque sorte des paramètres de l'algorithme de programmation dynamique, dépendent du couple de prototypes (i,j). Nous les noterons dorénavant a(i,j), b(i,j), c(i,j).

Les relations récursives de programmation dynamique s'écrivent maintenant de la manière suivante (dans le cas de la contrainte sans condition de pente):

$$d1 = D(i-1,j) + a(i,j).d(i,j)$$
$$l1 = L(i-1,j) + a(i,j)$$
$$d2 = D(i-1,j-1) + b(i,j).d(i,j)$$
$$l2 = L(i-1,j-1) + b(i,j)$$
$$d3 = D(i,j-1) + c(i,j).d(i,j)$$
$$l3 = L(i,j-1) + c(i,j)$$
$$d = \text{Arqmin}\left(\frac{d1}{l3}, \frac{d2}{l2}, \frac{d3}{l3}\right)$$

si $d = \frac{d1}{l1}$ alors D(i,j) = d1 et L(i,j) = l1

si $d = \frac{d2}{l2}$ alors D(i,j) = d2 et L(i,j) = l2

si $d = \frac{d3}{l3}$ alors D(i,j) = d3 et L(i,j) = l3

où L(i,j) est la longueur du chemin de recalage partiel optimal aboutissant au point (i,j).

Les références sont de nouveau déterminées par la procédure Minimax (chapitre 1, partie C), mais l'algorithme de programmation dynamique utilise des pondérations (a(i,j),b(i,j),c(i,j)) par couple (i,j) de prototypes.

Il s'agit maintenant de déterminer les valeurs a(i,j), b(i,j) et c(i,j) pour tout couple (i,j).

Les différentes répétitions qui ont servi à l'apprentissage pour constituer le codebook correspondant à un mot sont comparées deux à deux par programmation dynamique avec les coefficients a = 1, b = 1, c = 1.

Pour chaque couple (i,j) de prototypes (concrétisé par un point du chemin de recalage), nous évaluons le nombre de déplacements horizontaux a(i,j), obliques b(i,j), verticaux c(i,j) qui ont provoqué le passage du chemin optimal par le couple en question. C'est-à-dire, pour toutes les prononciations d'un mot utilisées, nous comptons le nombre de fois ca(i,j) où le couple de prototypes (i,j) a été atteint à l'aide de la contrainte horizontale, de même pour cb(i,j) et cc(i,j).

Ensuite nous déterminons les quantités:

$$cf(i,j) = ca(i,j) + cb(i,j) + cc(i,j)$$

$$la(i,j) = \frac{ca(i,j)}{cf(i,j)}$$

$$lb(i,j) = \frac{cb(i,j)}{cf(i,j)}$$

$$lc(i,j) = \frac{cc(i,j)}{cf(i,j)}$$

Les pondérations a(i,j), b(i,j), c(i,j) sont obtenues en appliquant une transformation à la(i,j), lb(i,j) et lc(i,j).

Nous avons testé les transformations suivantes:

$$a(i,j) = 1 + \frac{1}{la(i,j)}$$

$$b(i,j) = 1 + \frac{1}{lb(i,j)}$$

$$c(i,j) = 1 + \frac{1}{lc(i,j)}$$

$$a(i,j) = 1 - \frac{1}{Ln(la(i,j))}$$

$$b(i,j) = 1 - \frac{1}{Ln(lb(i,j))}$$

$$c(i,j) = 1 - \frac{1}{Ln(lc(i,j))}$$

$$a(i,j) = \frac{1}{la(i,j)}$$

$$b(i,j) = \frac{1}{lb(i,j)}$$

$$c(i,j) = \frac{1}{lc(i,j)}$$

$$a(i,j) = 1 - Ln(la(i,j))$$
$$b(i,j) = 1 - Ln(lb(i,j))$$
$$c(i,j) = 1 - Ln(lc(i,j))$$

Toutes ces transformations ont été testées sur le même corpus que précédemment et les résultats sont résumés dans le tableau ci-dessous.

transformation	1+1/X	1-(1/LnX)	1/X	1-LnX
% erreur	2,5	>20	5,3	>20

Le meilleur taux de bonne reconnaissance est obtenu pour la transformation X-->1+1/x et vaut 97,5%

Nous pouvons constater que par cette méthode nous avons obtenu 2,5% d'erreur, soit une amélioration de 2% sur le score réalisé au chapitre 3 de la partie C. Un triplet de pondérations (a,b,c) par couple de prototypes permet de tenir compte de la structure temporelle des mots et améliore la reconnaissance.

2. PONDERATIONS (a,b,c) PAR MOT

Nous avons essayé cette fois d'introduire un triplet de pondérations (a,b,c) par mot et non plus par couple de prototypes. La méthode de reconnaissance est la même que précédemment.

Pour calculer chaque triplet (a,b,c), nous avons procédé de la façon suivante:

- nous avons comparé deux à deux toutes les répétitions d'un même mot du corpus d'apprentissage avec la pondération (1,1,1).

- nous avons compté le nombre total ca(m) où un point quelconque du chemin de recalage optimal est atteint par la transition horizontale de la contrainte. De même, nous avons déterminé cb(m) qui évalue le nombre de transitions obliques et cc(m) qui est le

compteur des transitions verticales.

- posons:

$$cf(m) = ca(m) + cb(m) + cc(m) \quad | \quad la(m) = \frac{ca(m)}{cf(m)}$$

$$lb(m) = \frac{cb(m)}{cf(m)}$$

$$lc(m) = \frac{cc(m)}{cf(m)}$$

- nous avons fait subir les transformations suivantes à la(m), lb(m), lc(m) pour calculer a(m), b(m), c(m):

$$a(m) = 1 + \frac{1}{la(m)}$$

$$b(m) = 1 + \frac{1}{lb(m)}$$

$$c(m) = 1 + \frac{1}{lc(m)}$$

$$a(m) = -Ln(la(m)) \qquad b(m) = -Ln(lb(m)) \qquad c(m) = -Ln(lc(m))$$

$$a(m) = 1 - \frac{1}{Ln(la(m))}$$

$$b(m) = 1 - \frac{1}{Ln(lb(m))}$$

$$c(m) = 1 - \frac{1}{Ln(lc(m))}$$

$$a(m) = \frac{1}{la(m)}$$

$$b(m) = \frac{1}{lb(m)}$$

$$c(m) = \frac{1}{lc(m)}$$

$$a(m) = 1 - Ln(la(m)) \quad b(m) = 1 - Ln(lb(m)) \quad c(m) = 1 - Ln(lc(m))$$

Les résultats obtenus sur le même corpus que précédemment sont indiqués dans le tableau suivant:

transformation	1+1/X	-LnX	1-1/LnX	1/X	1-lnX
% erreur	3,7	>20	17,5	3,3	12

Les deux meilleurs scores de reconnaissance sont obtenus pour les mêmes transformations qu'auparavant mais dans un ordre inverse. Cette fois, c'est la transformation X-->1/X qui obtient le meilleur score de reconnaissance (96,7%) mais il est inférieur à celui que nous avons eu en utilisant un triplet (a,b,c) par couple de prototypes, ce qui est compréhensible puique nous n'avons plus tenu compte de la structure temporelle.

CHAPITRE 6: INFORMATIONS TEMPORELLES.

Si l'on observe, une fois codés, les mots du corpus d'apprentissage qui ont servi à déterminer un codebook, on peut constater que les vecteurs prototypes apparaissent dans des portions précises des diffèrentes répétitions. Par exemple, si l'on considère le mot "un", le prototype "silence" n'a d'occurrence qu'au début et à la fin de la locution. Cette constation a conduit Burton [Burton 84] et Rabiner [Rabiner 85] à modifier légèrement leur stratégie de reconnaissance.

Burton a modifié la méthode qu'il avait développée au préalable en constituant un codebook non plus pour un mot tout entier mais pour chaque portion du mot. En effet, il considère les différentes prononciations comme "alignées à gauche" et opère une normalisation linéaire de façon que toutes les répétitions d'un même mot aient la même longueur n. C'est ce qu'il a dénommé "normalisation par la gauche". Les mots sont alors partagés en Ns parties de longueur égale, et pour chacune de ces parties, il calcule un codebook dit de "section". Il y a donc autant de zones dans le mot que de sections du codebook. Un codebook est alors un ensemble ordonné de sections du codebook, c'est un codebook multisection (la relation d'ordre est la relation "précède dans le temps").

Il détermine ensuite comme précédemment un codebook multisection par mot du vocabulaire. Lors de la reconnaissance, l'occurence inconnue est codée suivant chaque codebook multisection de la façon suivante:

- elle est tout d'abord normalisée par la gauche.
- elle est ensuite partagée en zones de la même manière qu'à l'apprentissage et chaque portion est codée à l'aide de la section correspondante du codebook.

La distorsion de codage est déterminée de la même façon que lorsque Burton utilisait un codebook unisection. Le mot reconnu est celui qui a servi à constituer le codebook multisection qui minimise la distorsion de codage totale.

Rabiner a procédé différemment. Il constitue également un codebook pour chaque mot du vocabulaire, mais il utilise la méthode de Burton des codebooks unisection comme un préprocesseur qui élimine les mots candidats qui obtiennent un score de distorsion de codage trop important. Le choix final est réalisé par un algorithme de programmation dynamique. Pour incorporer des informations temporelles, Rabiner propose de tenir compte de la probabilité d'apparition d'un prototype à un rang donné dans le mot pour prendre la décision de reconnaissance. Pour cela, il a utilisé la stratégie suivante:

- il dispose d'un codebook unisection par locution du vocabulaire considéré. Lors de l'apprentissage, chaque prononciation a été normalisée linéairement à une longueur n donnée et codée à l'aide du codebook correspondant au mot prononcé. Pour chaque valeur i ($1 \leq i \leq n$) et pour chaque prototype v_j d'un codebook, il calcule la probabilité d'apparition du prototype v_j à la place i. Nous la noterons $p(v_j,i)$. Elle est déterminée de la façon suivante:

- pour chaque valeur i, tous les vecteurs du codebook dont le score de distorsion au codage est inférieur à une limite fixée sont considérés comme ayant apparu.

- $p(v_j,i)$ est le rapport entre le nombre de fois où le vecteur v_j est apparu au temps i (avec la définition ci-dessus) et le nombre total de fois où un quelconque prototype du codebook figure dans le corpus d'apprentissage au temps i.

De cette façon, $\sum_{j \in CBk} p(v_j) = 1$ et ceci pour chaque valeur de k comprise entre 1 et n.

Rabiner a utilisé une analyse LPC et il a stocké ces probabilités sous la forme $-A.Ln(p(v_j,i))$ où A est un coefficient choisi de telle sorte que la moyenne des probabilités soit du même ordre de grandeur que la moyenne des distances LPC.

Finalement, à la reconnaissance, la distorsion de codage est calculée à chaque instant i par:

$$D(i) = (1-al)Dlpc(T(i),R(i)) + Dprob(T(i),R(i))$$

où Dlpc est la distance LPC (distorsion de codage) et Dprob la distance des probabilités. al est un coefficient choisi pour optimiser le score de reconnaissance.

Dprob est déterminé par $\hat{p}(v_j,i)$ si v_j est le prototype qui code le prélèvement T(i) au temps i.

La reconnaissance fonctionne donc en trois étapes:

- 1)tous les mots candidats qui obtiennent une distorsion moyenne inférieure à un seuil fixé Δ sont retenus, avec

$$\frac{1}{I}\sum_{i=1}^{I} D(i) = \frac{1}{I}\sum_{i=1}^{I} (1-al)\ .\ Dlpc(T(i),R(i)) + al\ .\ Dprob(T(i),R(i)) \leq \Delta$$

- 2)s'il n'y a qu'un candidat, il est déclaré mot reconnu.

- 3)sinon,la décision finale de reconnaissance est prise par un algorithme de programmation dynamique.

Des expériences réalisées par Rabiner ont montré que al=0,5 donne le meilleur taux de reconnaissance.

Nous avons adopté une approche quelque peu différente:

- les mots ne sont pas normalisés en longueur, le codebook est unisection, chaque locution est partagée en n tranches égales.

- au codage, lors du calcul de la distorsion D entre le prélévement t(i) de la forme test et le prototype v_j du codebook k, nous procédons en retenant comme vecteur code t(i,k) non plus le prototype qui minimise la distance de Hamming $d(t(i),v_j)$, mais celui qui rend minimale la valeur $d(t(i),v_j)+\frac{1}{p(l,v_j)}$ où $p(l,v_j)$ est le nombre d'occurences de v_j dans la zone l de toutes les répétitions d'un mot du corpus d'apprentissage.

- nous retenons les trois candidats qui minimisent la distorsion totale calculée en sommant sur tout le mot la distorsion D(i)

calculée précédemment.

- la décision de reconnaissance est prise par minimisation du score Ddtw(k) + D(k) où Ddtw est le score de programmation dynamique entre la forme inconnue et la référence correspondant au codebook k.

Nous avons testé différentes valeurs de n (n=5,10,15,20,25) et nous avons obtenu, sur le même corpus que précèdemment (500 formes, 25 répétitions de chacun des 10 chiffres par locuteur, 2 locuteurs) les scores de bonne reconnaissance qui sont rassemblés dans le tableau ci-dessous:

n	5	10	15	20	25
% erreur	2,8	2,3	2	2,4	2,8

La figure 1 illustre les résultats ainsi obtenus.

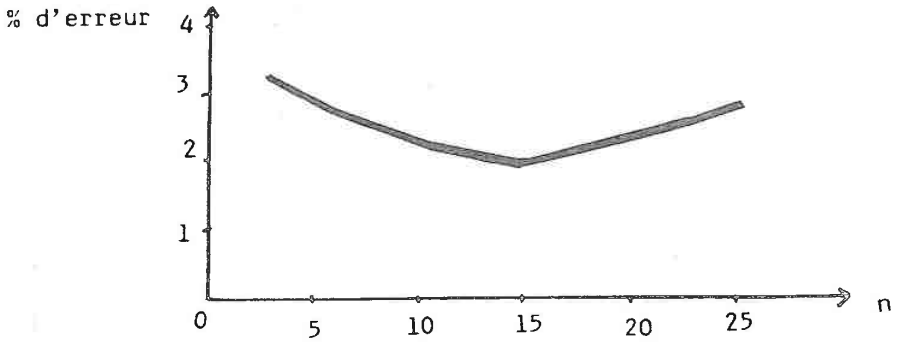


figure 1 : taux d'erreur en fonction de n.

Nous pouvons constater que le minimum a été atteint pour une valeur de n égale à 15 et que le taux de reconnaissance vaut alors 98%. Ainsi, prendre en compte la fréquence d'apparition d'un prototype à une place donnée dans le mot améliore sensiblement le taux de bonne reconnaissance.

CONCLUSION

Nous venons d'étudier quelques algorithmes de reconnaissance de mots isolés utilisant à la fois les techniques de programmation dynamique et de quantification vectorielle.

Nous avons constaté que le gain de place mémoire réalisé ne s'accompagne pas d'une détérioration des performances de la reconnaissance: le taux de 98% obtenu à l'aide de la méthode que nous venons de décrire au chapitre 6 (reconnaissance effectuée par minimisation de la somme du score de programmation dynamique, de la distorsion de codage et des probabilités d'apparition des prototypes) est même légérement supérieur au score de 97,1% réalisé sur le même corpus en ne considérant que la programmation dynamique simple (chapitre 4 de la partie B).

Burton [Burton 85] a proposé de remplacer la quantification vectorielle par une quantification matricielle. Un prototype n'est plus un vecteur de coefficients mais une matrice (en quelque sorte, une suite ordonnée dans le temps de quelques vecteurs de coefficients). Les matrices sont obtenues à l'aide d'une généralisation de l'algorithme de Linde, Buzo et Gray (chapitre 1 de la partie C).

Il a utilisé la même méthode de reconnaissance que pour les codebooks multisection (chapitre de la partie C) et il a testé sur le même corpus de mots isolés la reconnaissance avec quantification vectorielle multisection, puis la reconnaissance avec quantification matricielle. Il a constaté une amélioration du score de bonne reconnaissance en utilisant la quantification matricielle par rapport à celui qu'il avait obtenu avec la quantification vectorielle et les codebooks multisection.

le problème des références se pose lorsque l'on désire réaliser une application multilocuteur: une forme par mot du vocabulaire ne suffit pas à représenter toutes les variantes de prononciation et il faut alors garder en mémoire un certain nombre de formes par mot.

Une manière intéressante de stocker les références est de les mettre sous la forme d'un réseau ([Lee 85] [Kopek 85]). A chaque mot du vocabulaire correspond un réseau, un chemin dans le réseau est une forme

de référence. Ainsi, les parties communes ne sont représentées qu'une seule fois, ce qui économise de la place mémoire.

PARTIE D

RECONNAISSANCE DE MOTS ENCHAINES

INTRODUCTION

CHAPITRE 1 : L'ALGORITHME DE BRIDLE ET NAKAGAWA

CHAPITRE 2 : RELACHEMENT DES CONTRAINTES

CHAPITRE 3 : QUANTIFICATION VECTORIELLE ET RECONNAISSANCE DE MOTS ENCHAINES

CHAPITRE 4 : CONTRAINTES SYNTAXIQUES

CHAPITRE 5 : LIAISONS ET COARTICULATION

CHAPITRE 6 : APPROCHE SEMI GLOBALE

CONCLUSION

INTRODUCTION

Les techniques que nous avons décrites jusqu'à maintenant en reconnaissance de mots isolés peuvent être étendues à la reconnaissance de séquences de mots prononcés sans pause (mots enchainés). Cependant, signalons qu'aux problèmes soulevés par la reconnaissance de mots isolés viennent s'ajouter, entre autres, la difficulté de la segmentation de la suite de mots, le problème de la coarticulation entre mots adjacents, l'accroissement de la quantité de calcul requise par les algorithmes ([Vintsyuk 71], [Gauvain 82]).

Les mots qui forment la phrase à reconnaitre sont prononcés de façon continue sans silences intermédiaires, leurs frontières ne peuvent donc pas être déterminées par une simple analyse de l'enveloppe du signal.

Il est alors possible d'envisager deux méthodes différentes pour réaliser la segmentation en mots:

- une première méthode est de définir des règles de segmentation, ceci impose un vocabulaire précis et bien défini puisqu'elles sont fondées sur des critères acoustiques. Sambu et Rabiner ([Sambu 76]) ont établi de telles règles pour un vocabulaire composé des chiffres anglais. Cependant, cette méthode ne peut s'appliquer qu'à des vocabulaires simples et peu importants, ne comprenant que des mots d'une ou au plus deux syllabes. Remarquons que les inévitables erreurs commises à ce niveau limitent et pénalisent les performances du système avant même de considérer l'étape de reconnaissance;

- une autre solution consiste à ne pas segmenter la parole, les frontières des mots étant déterminées au cours de la reconnaissance. Cette méthode repose sur l'idée que le principe de la reconnaissance de mots isolés peut s'appliquer directement à la reconnaissance de mots enchainés en considérant simplement que l'occurence inconnue est l'image acoustique de toute la phrase.

C'est à cette deuxième solution que nous allons nous intéresser dans la suite de cette étude. Nous verrons comment l'algorithme de programmation dynamique que nous avons utilisé auparavant peut se généraliser à la reconnaissance de mots enchainés.

CHAPITRE 1: ALGORITHME DE BRIDLE ET NAKAGAWA

En reconnaissance de mots isolés, nous avons vu que la programmation dynamique résoud efficacement le problème du recalage temporel non linéaire. Vintsyuk [Vintsyuk 71], Bridle et Brown [Bridle 82], Sakoe [Sakoe 79] ont formulé le problème de la reconnaissance de mots enchainés comme un problème d'optimisation semblable à celui des mots isolés.

Une des approches les plus prometteuses dans le domaine de la reconnaissance de mots enchainés semble donc être la programmation dynamique. Un avantage de cette méthode est le peu d'informations qu'elle nécessite: seule une forme de référence est nécessaire pour chaque mot du dictionnaire. De plus, les trois aspects- lien des mots, recalage temporel, reconnaissance- sont réalisés simultanément.

Sakoe a formalisé ainsi le problème de la reconnaissance de mots enchainés (figure 1): il s'agit de comparer une forme inconnue (soit la phrase à identifier) à l'ensemble des "superformes" de référence obtenues par la concaténation d'un nombre quelconque de mots appartenant au vocabulaire de l'application considérée. La suite de mots qui constituent la phrase prononcée est donnée par la superforme de référence qui satisfait le mieux les critères de comparaison.

Soient T la forme inconnue de longueur I en prélèvements et V un ensemble de N formes de référence Rk de longueur en prélèvements Jk. Le but de la reconnaissance de mots enchainés est donc de déterminer la superforme de référence $SR_q^n = R_{q(1)}+\ldots+R_{q(n)}$ définie par la concaténation de n formes de référence $R\hat{q}(k)$ (k variant de 1 à n) qui coincide le mieux avec la forme d'entrée T.

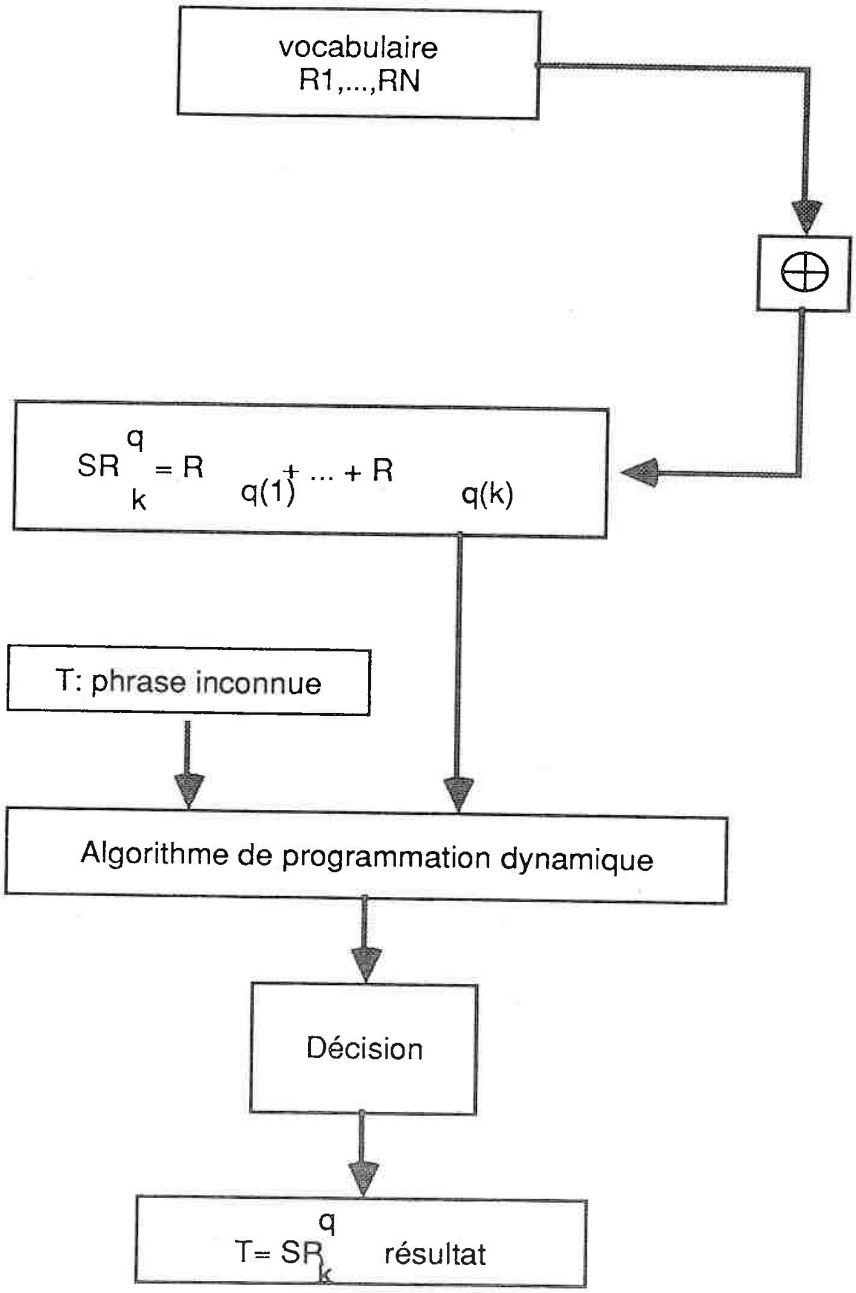


figure 1: principe de la reconnaissance de mots enchainés d'aprés Sakoe.

1. PRESENTATION GENERALE DES ALGORITHMES

Pour résoudre cette question, Sakoe [Sakoe 79] a proposé un algorithme à deux niveaux. Au premier niveau, toutes les formes de référence sont systématiquement comparées à toutes les sous sections possibles de la forme inconnue. Cette opération de comparaison est la même que pour le recalage temporel non linéaire en reconnaissance de mots isolés.

Au deuxième niveau, l'estimation optimale de la séquence inconnue est obtenue en utilisant les scores générés à l'étape précédente par la minimisation de la distance totale de toutes les séquences possibles de mots.

Plus récemment, Myers et Rabiner [Myers 81] ont proposé une autre solution pour résoudre le problème d'optimisation de la reconnaissance de mots enchainés, ils utilisent la propriété suivante: la comparaison de toutes les séquences possibles de mots peut être améliorée par la concaténation successive des formes de référence. Ils ont obtenu ce qu'ils ont appelé un algorithme de "construction par niveaux" : pour chaque niveau n, et pour chaque prélèvement i de la forme inconnue, il évalue la distance cumulée associée à la meilleure superforme de référence de n mots coincidant avec les i premiers prélèvements de la forme à tester. Toutes les distances cumulées obtenues ainsi initialisent le processus de programmation dynamique du niveau suivant.

La superforme optimale de référence est celle qui donne au dernier prélèvement de la forme inconnue la plus petite distance cumulée. L'algorithme de Myers et Rabiner, bien que constituant une amélioration de l'algorithme de Sakoe, présente cependant deux défauts majeurs qui font qu'il n'est pas réellement orienté vers le temps réel:

- le premier inconvénient est la récursion qui est introduite sur le nombre de mots pouvant être contenus dans la forme inconnue. Par conséquent, il est impératif de fixer arbitrairement le nombre maximal de mots pouvant être contenus dans la phrase à identifier afin de donner une condition d'arrêt à l'algorithme de comparaison.

- au cours d'un changement de niveau, beaucoup de distances locales qui ont déjà été évaluées doivent être recalculées à cause des retours arrière partiels réalisés sur la forme d'entrée.

Récemment, Bridle [Bridle 82] et Nakagawa [Nakagawa 83] ont indépendamment l'un de l'autre proposé un algorithme qui élimine les faiblesses de l'algorithme de Myers et Rabiner. Cet algorithme est en un seul niveau, ce qui signifie qu'il n'opère aucun retour-arrière sur la forme inconnue. Ainsi les distances locales ne sont évaluées qu'une seule fois. Tout cela laisse présager qu'il est bien orienté temps réel.

2. ALGORITHME DE BRIDLE ET NAKAGAWA

L'idée fondamentale de la reconnaissance globale de mots enchainés est la détermination du chemin optimal dans un espace de comparaison tridimensionnel tel qu'il est représenté figure 2.

Chaque point de l'espace de comparaison peut être représenté par un triplet (i,j,k) où k désigne la forme référence Rk, i le prélèvement considéré de la forme inconnue et j le jème prélèvements de Rk. Avec un tel formalisme, un chemin de recalage w est donné par une suite de points (i(k'),j(k'),k(k')) où k' varie de 1 à K, K représentant le nombre de points du chemin.

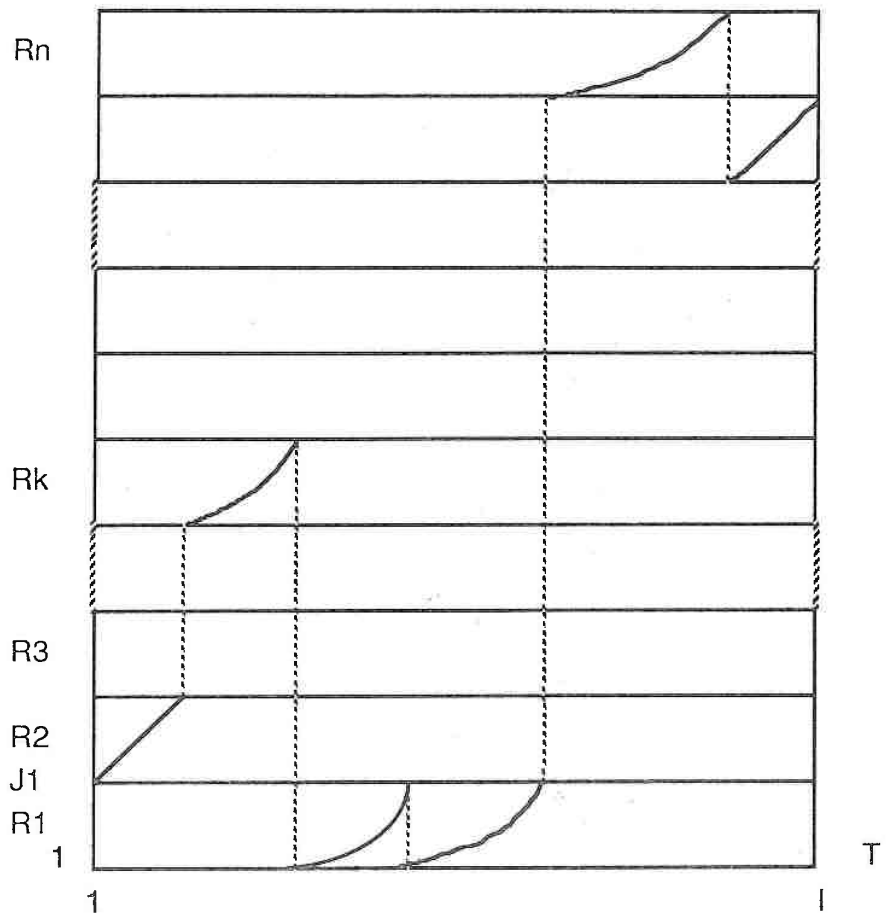


figure 2: Chemin de recalage selon Bridle et Nakagawa.

Le problème de la reconnaissance de mots enchainés se pose de la même manière que celui de la reconnaissance de mots isolés: il s'agit de trouver le chemin de recalage optimal $\hat{w}$ qui minimise la fonctionnelle D suivante:

$$D(w) = \min_{i(k'),j(k'),k(k')} \sum_{k=1}^{K} \frac{d(i(k'),j(k'),k(k')),P(k')}{N(P)}$$

où d(i(k'),j(k'),k(k')) représente la distance locale entre le i(k')ème prélèvement de la forme inconnue T et le j(k')ème prélèvement de la forme de référence $R_{k'}$.

Dans le cas d'une contrainte symétrique, le facteur de normalisation N(P) dépend de la longueur du chemin de recalage et n'est malheureusement pas une constante que l'on peut sortir de la minimisation.

Dans le cas d'une contrainte asymétrique, au contraire, N(P) représente la longueur de la forme inconnue T et est par conséquent une constante que l'on peut sortir de la minimisation.

Nous allons nous placer dans le cas général: N(P) ne peut pas être sorti de la minimisation.

Pour résoudre cette équation par programmation dynamique, il est nécessaire de définir en chaque point (i,j,k) de l'espace de comparaison deux quantités:

- D(i,j,k) qui est la distance cumulée optimale.
- L(i,j,k) qui est la longueur du chemin partiel optimal aboutissant au point (i,j,k).

Comme D(i,j,k) est la somme des distances locales, elle peut être décomposée de la même manière que le chemin de recalage en une distance cumulée le long du meilleur chemin aboutissant au point précédent(i,j,k) et la distance locale associée au point (i,j,k) lui-même.

Pour obtenir le meilleur chemin, nous devons donc sélectionner le point précédent (i,j,k) qui permet d'obtenir le rapport $\frac{D(i,j,k)}{L(i,j,k)}$ minimal.

Nous obtenons la relation suivante:

$$(i,j,k) = \underset{(i',j',k') \in V(i,j,k)}{\text{Argmin}} \left[\frac{d(i,j,k,i',j',k') + D(i',j',k')}{l(i,j,k,i',j',k') + L(i',j',k')}\right]$$

$$D(i,j,k) = D(\hat{i},\hat{j},\hat{k}) + d(i,j,k,\hat{i},\hat{j},\hat{k})$$

$$L(i,j,k) = L(\hat{i},\hat{j},\hat{k}) + l(i,j,k,\hat{i},\hat{j},\hat{k})$$

où:

- $d(i,j,k,i',j',k')$ est la distance locale entre les points (i,j,k) et (i',j',k'),

- $l(i,j,k,i',j',k')$ la longueur du chemin local joignant les points (i,j,k) et (i',j',k'),

- et $V(i,j,k)$ le voisinage du point (i,j,k) définie par la contrainte locale utilisée.

Cette équation est l'équation de programmation dynamique générale qui sera utlisée dans l'algorithme de reconnaissance des mots enchainés.

Nous n'avons pas précisé le voisinage $V(i,j,k)$ du point (i,j,k) défini par la contrainte locale utilisée. Reprenant le formalisme introduit par Ney [Ney 84], nous considérons deux types de voisinages et donc deux types de contraintes locales. Prenons par exemple la contrainte sans condition de pente (figure 3):

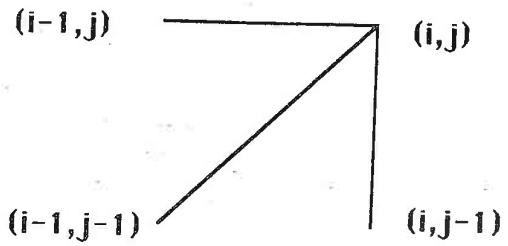


figure 3: contrainte sans condition de pente.

- contrainte interne (figure 4):

  c'est le cas où j > 1. Les trois points précédents possibles appartiennent à la même forme (dans le cas général, si μ est la profondeur de la contrainte, ceci se produit pour $j > \mu$). Alors:

  $V(i,j,k)=\{(i-1,j,k),(i-1,j-1,k),(i,j-1,k)\}$

- contrainte externe(figure 5):

  c'est le cas j=1 (ou $j \leqq \mu$ dans le cas général). Dans ce cas, certains points précédents appartiennent à une forme de référence différente (d'où le nom externe). Alors:

  $$V(i,j,k)=\{(i-1,j,k)\} \quad \bigcup_{1 \leqq k' \leqq n} \{(i-1,Jk',k')\}.$$

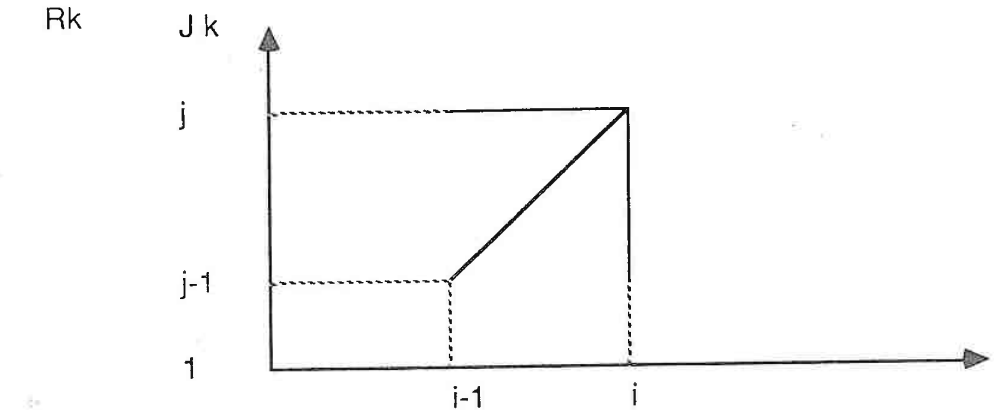


figure 4: contrainte interne.

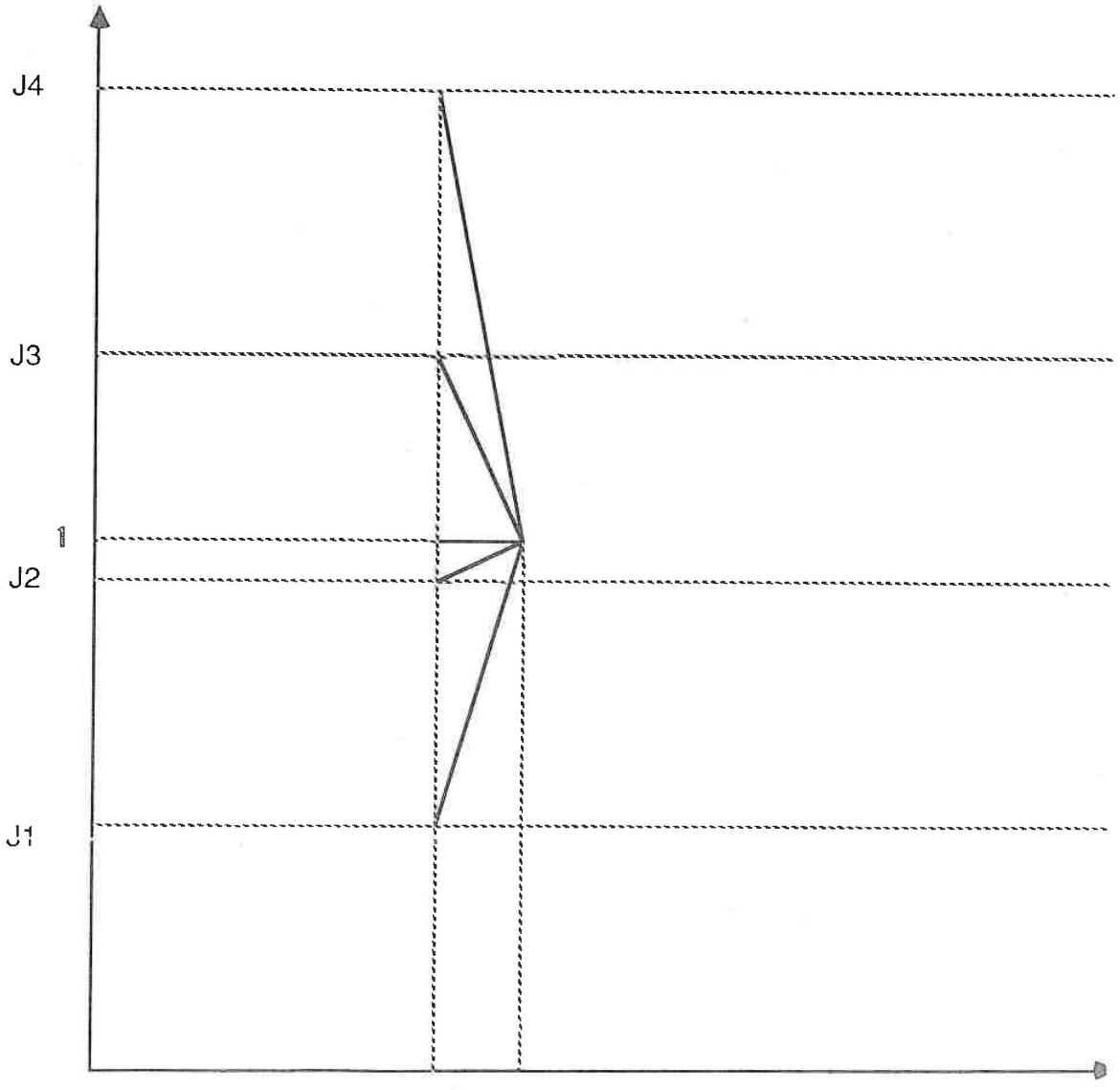


figure 5: contrainte externe.

Nous pouvons maintenant donner l'algorithme:

- 1)initialisation de toutes les variables,

- 2)détermination en tous points (i,j,k) du plan de comparaison de D(i,j,k) et L(i,j,k) comme indiqué précédemment,

- 3)décision: détermination du meilleur chemin de recalage. La séquence de mots reconnus est obtenue en "remontant" les décisions prises tout au long du déroulement de l'algorithme. Pour ce chainage arrière, nous pouvons constater que la seule connaissance des D(i,j,k) et L(i,j,k) ne suffit pas. L'information de chainage arrière doit être mémorisée dans un tableau durant l'évaluation des D(i,j,k) et L(i,j,k) par programmation dynamique. Pour le chemin de recalage aboutissant au point (i,j,k), nous remarquons qu'il y a un unique point de départ pour j = 1 dans la même forme Rk.

En conclusion, en chaque point (i,j,k) doit être définie l'origine P(i,j,k) du chemin partiel optimal inclus dans Rk et aboutissant en (i,j,k) (figure 6).

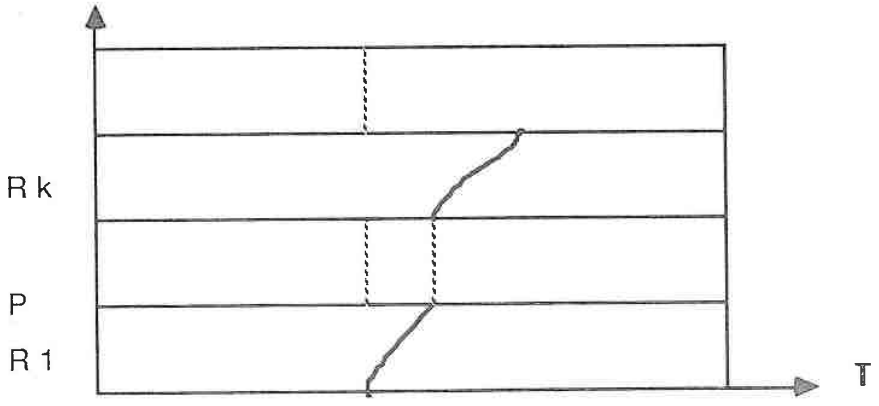


figure 6: origine du chemin partiel optimal.

Comme un point est défini par trois coordonnées dans l'espace de comparaison, trois fonctions pointeurs sont nécessaires:

- FK(i,j,k) qui rend le sous-espace Rk' auquel appartient P(i,j,k).

- FJ(i,j,k) qui rend la deuxième coordonnée de P(i,j,k) (dans notre cas, P(i,j,k) = Jk').

- FI(i,j,k) qui rend la valeur de la première coordonnée de P(i,j,k).

Nous avons donc P(i,j,k) = (FI(i,j,k),FJ(i,j,k),FK(i,j,k)).

Nous avons maintenant tous les éléments nécessaires pour spécifier l'algorithme de reconnaissance de mots enchainés.

3. SPECIFICATION DE L'ALGORITHME DE BRIDLE ET NAKAGAWA

- (* la première étape est une étape d'initialisation *)

```
pour k = 1 à n faire

        D(0,0,k)  = 0;
        L(0,0,k)  = 1(0,0,0,0,0,k);
        FI(0,0,k) = 0;
        FJ(0,0,k) = 0;
        FK(0,0,k) = 0;

        pour j = 1 à Jk faire

                D(0,j,k) = ∞;
                L(0,j,k) = 1;
        finfaire

finfaire
```

(* la deuxième étape de cet algorithme est la programmation dynamique proprement dite *)

pour i = 1 à I faire

pour k = 1 à n faire

pour j = 1 à Jk faire

$$(i,j,k) = \underset{(i',j',k')\in V(i,j,k)}{\text{Argmin}} \frac{D(i',j',k')+D(i',j',k',i,j,k)}{L(i',j',k')+L(i',j',k',i,j,k)}$$

$$D(i,j,k) = D(\hat{\imath},\hat{\jmath},\hat{k}) + d(\hat{\imath},\hat{\jmath},\hat{k},i,j,k)$$

$$L(i,j,k) = L(\hat{\imath},\hat{\jmath},\hat{k}) + l(\hat{\imath},\hat{\jmath},\hat{k},i,j,k)$$

$$si(i,j,k)\in Vi = \bigcup_{1\leq k'\leq n} (i-1,Jk',k')$$

alors

FI (i,j,k) = FI $(\hat{\imath},\hat{\jmath},\hat{k})$

FJ (i,j,k) = FJ $(\hat{\imath},\hat{\jmath},\hat{k})$

FK (i,j,k) = FK $(\hat{\imath},\hat{\jmath},\hat{k})$

sinon

FI (i,j,k) = $\hat{\imath}$;

FJ (i,j,k) = $\hat{\jmath}$;

FK (i,j,k) = $\hat{k}$;

finsi

finpour

finpour

finpour

(* la troisième étape est la décision *)

kopt= $\underset{1 \leq k' \leq n}{\text{Min}} \frac{D(I,Jk',k')}{L(I,Jk',k')}$

q(0) = kopt (* q est la table qui contiendra à la fin du retour arrière la superforme optimale en ordre inverse *)

k = 0

P(0) = (I,Jkopt,kopt)

Tant que (P_{k+1} = (FI(Pk),FJ(Pk),FK(Pk))=(0,0,0)) faire
q(k+1) = FK(Pk)
k=k+1

finfaire

nmot = k+1 (* nombre de mots reconnus *)

Pour i=1 à nmot faire

q'(i) = q(nmot-i) (* ordonne correctement la superforme *)

finfaire

(* le mot reconnu est *)

T=$R_{q'(1)}+\ldots+R_{q'(nmot)}$

fin.

Tel qu'il est spécifié, cet algorithme nécessite l'utilisation de deux tableaux à trois dimensions D(i,j,k), L(i,j,k). Nous pouvons constater que suivant la forme de la contrainte, il est parfois possible de les réduire à deux ou trois colonnes. Par exemple, dans le cas de la contrainte sans condition de pente (figure 3), deux colonnes suffisent, qui contiennent les valeurs déterminées à l'étape i-1 et celles qui sont calculées à l'étape i.

Une autre remarque permet de diminuer encore l'espace mémoire utilisé.

Pour calculer D(i,j,k) et L(i,j,k), seuls D(i-1,j,k), D(i-1,j-1,k), D(i,j-1,k), L(i-1,j,k), L(i-1,j-1,k), L(i,j-1,k) sont nécessaires. Ensuite D(i-1,j-1,k) et L(i-1,j-1,k) ne seront plus utilisées. Compte tenu de cette remarque, nous avons procédé comme l'indique la figure 7.

On détermine D(i,j,k) à l'aide des relations de programmation dynamique récurrente.

On range D(i,j-1,k) dans D(j-1,k), écrasant la valeur D(i-1,j-1,k) qui n'est plus d'aucune utilité pour la poursuite de l'algorithme.

On met enfin D(i,j,k) dans D.

La figure 8 illustre cette démarche.

Lorsque l'on a fini de calculer D(i,j,k) pour j=Jk, on range à la fois D(i,Jk-1,k) et D(i,Jk,k) dans le tableau D(j,k) (figure 9). On procède de la même manière pour le tableau L.

4. IMPLANTATION ET TEST

Cet algorithme a été écrit en langage C sur MASSCOMP. L'acquisition du corpus a été réalisée sur Masscomp à une fréquence d'échantillonage

de 16kHz. L'analyse est effectuée par un banc de filtres de 32 canaux avec une répartition en fréquences linéaire entre 0 et 8000Hz. Le vocabulaire est composé des dix chiffres français et du mot blanc qui correspond à du silence. Nous avons introduit cette référence pour permettre au locuteur de marquer des pauses lors de l'élocution d'une suite de chiffres s'il le désire. La contrainte utilisée est celle représentée figure 3. En annexe 3, sont indiquées les suites de 2, 3, 4 chiffres qui ont été prononcées.

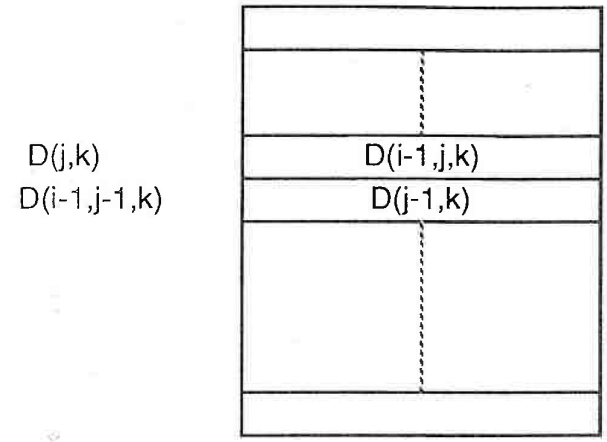
Au vue des premiers résultats, nous avons introduit deux paramètres supplémentaires:

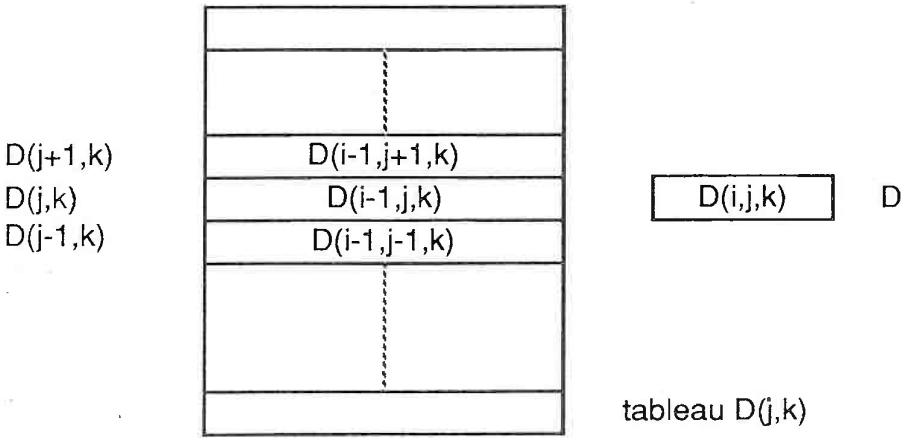
- une pénalité silence est ajoutée à la distance de Hamming entre un prélèvement silence et le prélèvement inconnu si au moins une composante de ce dernier est supérieure à un seuil donné. Ceci permet de ne pas considérer comme du silence un prélèvement qui aurait une composante importante et rien ailleurs. Cette pénalité a été fixée à la moitié de la valeur moyenne d'une composante calculée sur le corpus d'apprentissage,

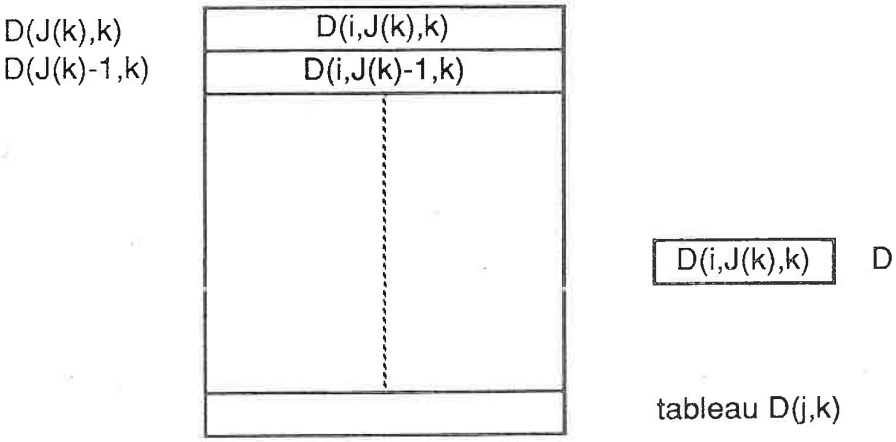
- une pénalité est ajoutée à la distance cumulée dans le cas où le chemin local choisi est l'horizontale ou la verticale. Ceci permet de limiter les chemins trop longtemps horizontaux et verticaux. Ce coefficient a été choisi comme la valeur moyenne de la distance entre deux prélèvements quelconques du corpus d'apprentissage.

Cet algorithme a été testé sur quatre locuteurs (une femme et trois hommes, deux locuteurs entrainés et deux locuteurs non entrainés). Pour chacun d'entre eux, le dictionnaire des formes de référence est constitué d'une élocution de chaque chiffre et d'un mot de silence.

Le taux moyen de bonne reconnaissance est de 90,7 %.


on désire déterminer D(i,j,k)
D(j,k)
D(i-1,j,k)
D(i-1,j-1,k)
D(j-1,k)
D(i-1,j,k)
variable
intermédiaire D
tableau D(j,k)


D(j+1,k)
D(j,k)
D(j-1,k)
D(i-1,j+1,k)
D(i-1,j,k)
D(i-1,j-1,k)
D(i,j,k)
D
tableau D(j,k)


D(J(k),k)
D(J(k)-1,k)
D(i,J(k),k)
D(i,J(k)-1,k)
D(i,J(k),k)
D
tableau D(j,k)

CHAPITRE 2: RELACHEMENT DES CONTRAINTES

Nous avons repris l'algorithme précédent pour l'adapter au cas du relâchement des frontières ([di Martino 84], [Boyer 84]). Il sera désormais autorisé à ne pas considérer un certain nombre de prélèvements en début et en fin des formes de référence Rk et test T. Notons:

- α: le nombre de prélèvements en début de forme test pouvant ne pas être pris en compte.

- β: le nombre de prélèvements en fin de forme test pouvant ne pas être pris en considération.

- $\gamma(k)$: le nombre de prélèvements en début de forme de référence k pouvant ne pas être pris en compte.

- $\delta(k)$: le nombre de prélèvements en fin de forme de référence k susceptibles de ne pas être considérés.

Un exemple de chemin de recalage est donné sur la figure 1.

Dans le cas général, la longueur du chemin de recalage n'est pas une constante comme précédemment, le problème se résoud par les relations de programmation dynamique que nous avons développées au chapitre précédent:

$$(\hat{i},\hat{j},\hat{k}) = \underset{(i',j',k')\in V(i,j,k)}{\text{Argmin}} \frac{D(i',j',k')+d(i',j',k',i,j,k)}{L(i',j',k')+l(i',j',k',i,j,k)}$$

$$D(i,j,k) = D(\hat{i},\hat{j},\hat{k}) + d(\hat{i},\hat{j},\hat{k},i,j,k)$$

$$L(i,j,k) = L(\hat{i},\hat{j},\hat{k}) + l(\hat{i},\hat{j},\hat{k},i,j,k)$$

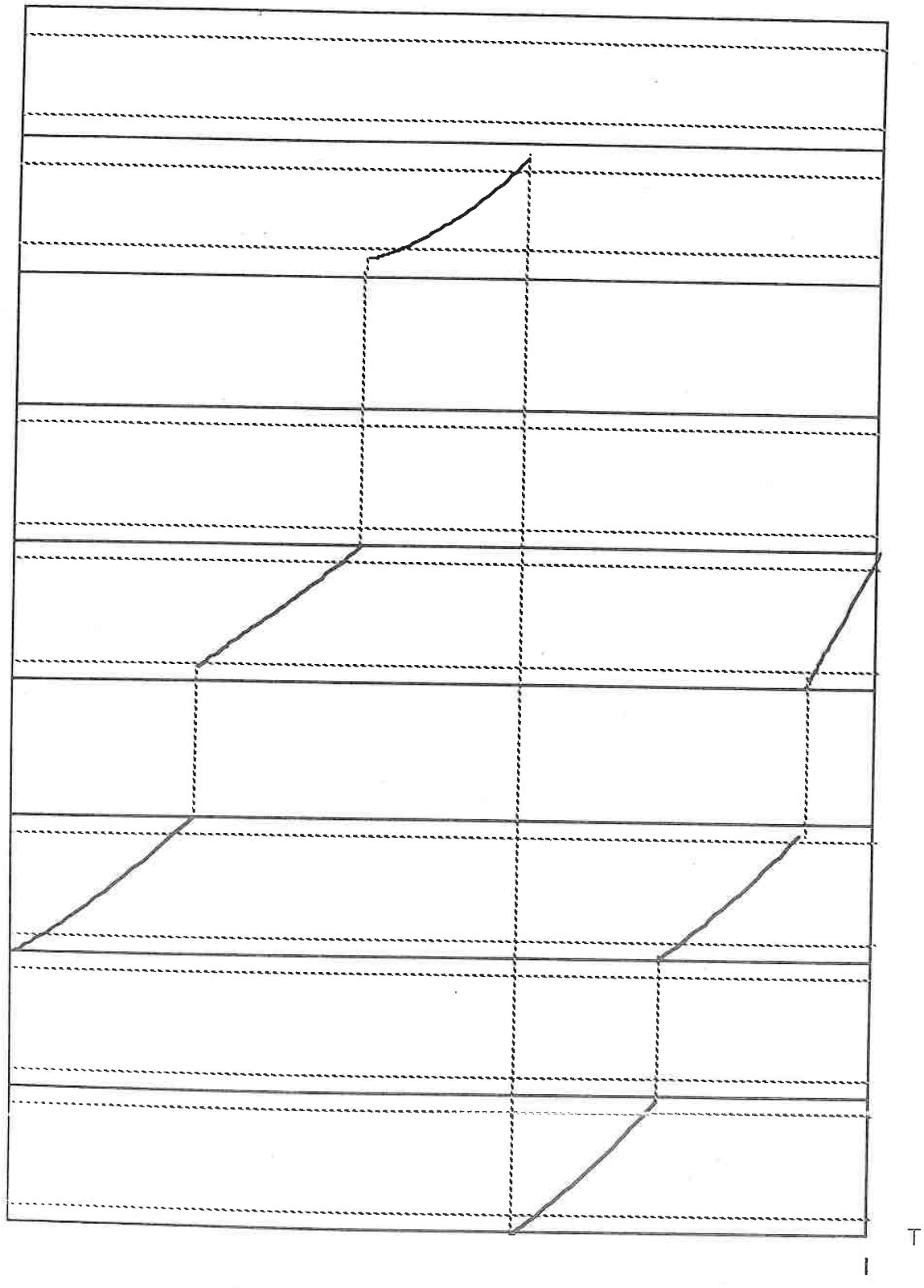


figure 1: exemple de chemin de recalage.

Mais le voisinage V(i,j,k) du point (i,j,k) défini par la contrainte locale utilisée doit être modifié. Reprenons l'exemple de la contrainte sans condition de pente.

Considérons les deux cas précédents:

- contrainte interne:

c'est lorsque (j-1) et j n'appartiennent pas à la zone de flou du début de la forme Rk (figure 2), c'est à dire $j > \gamma(k)$.

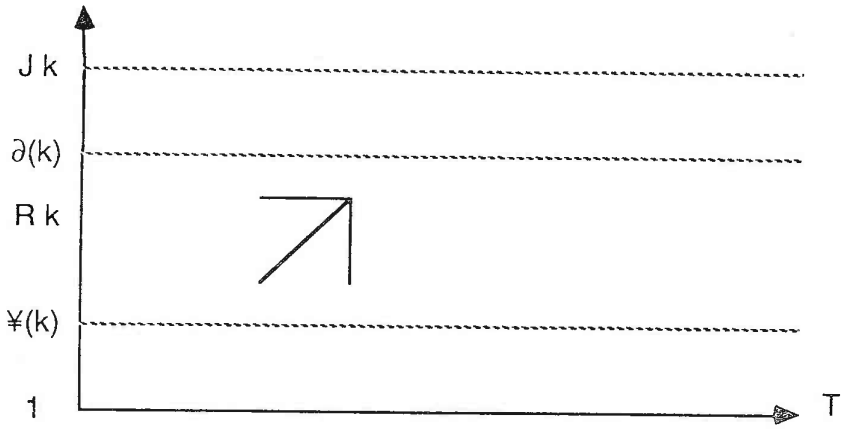


figure2: contrainte interne.

Alors, nous avons:

$V(i,j,k) = \{(i-1,j,k),(i-1,j-1,k),(i,j-1,k)\}$

- contrainte externe:

c'est lorsque (j-1) ou j appartiennent à la zone de flou du début de la forme Rk (figure 3), soit $j \leqq \gamma(k)$.

Dans ce cas, le voisinage V(i,j,k) devient:

$$V(i,j,k) = \{(i-1,j,k),(i-1,j-1,k),..,(i-1,1,k)\} \cup \{(i,j-1,k),..,(i,1,k)\}$$

$$\bigcup_{1 \leq k' \leq n} \{(i-1,Jk',k'),..,(i-1,Jk'-\delta(k'),k')\}$$

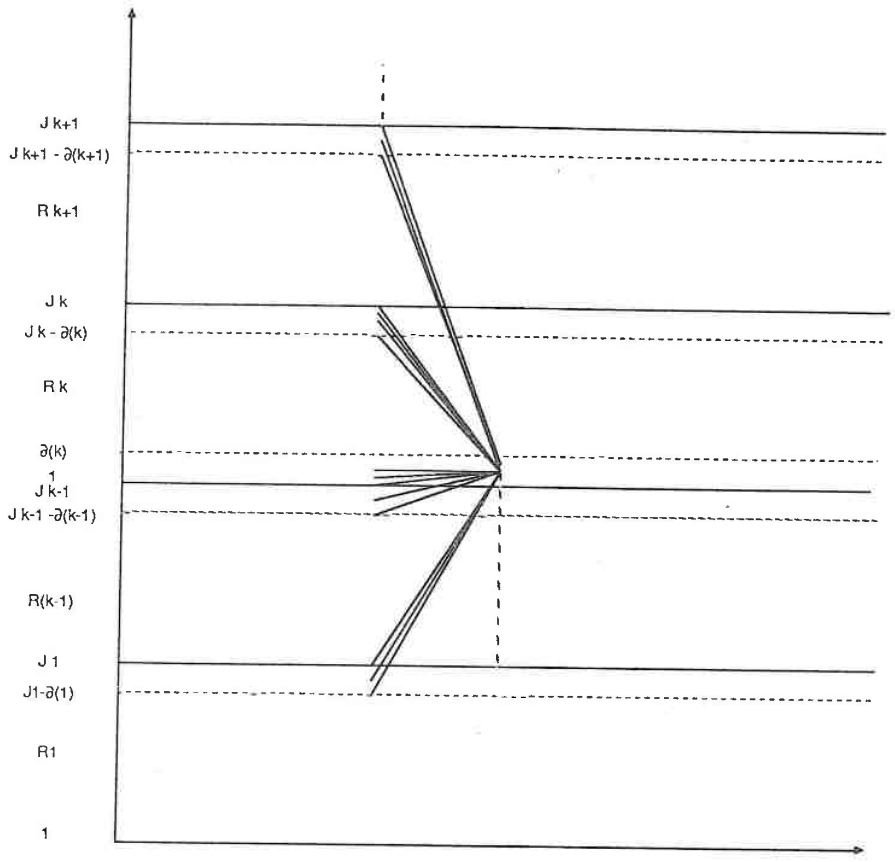


figure 3: contrainte externe.

Les différents voisinages étant définis, nous pouvons maintenant spécifier l'algorithme de reconnaissance des mots enchainés dans le cas du relâchement des contraintes.

Les trois fonctions pointeurs sont définies comme précédemment si ce n'est que cette fois la valeur FJ(i,j,k) = ĵ n'est pas uniquement Jk̂, mais est comprise entre Jk̂ - δ (k̂) et Jk̂.

L'algorithme présenté au chapitre précedent reste donc valable. la seule modification étant : $Vi = \bigcup_{1 \leq k' \leq n} (i-1, Jk', k')$ devient

$Vi = \bigcup_{1 \leq k' \leq n} (i-1, Jk', k'), \ldots, (i-1, Jk' - \delta(k'), k')$.

Cet algorithme a été implanté sur un Masscomp en langage C et testé sur le même corpus décrit au chapitre précédent.

Le taux moyen de bonne reconnaissance obtenu est de 91.3%.

Le nombre de rapports (distance sur longueur) calculés par l'algorithme de Bridle et Nakagawa décrit dans le chapitre précédent est N.I.J.3 où N est le nombre de formes du vocabulaire, I la longueur de la forme test, J la longueur moyenne d'une forme de référence, 3 est le nombre de chemins locaux autorisés par la contrainte sans condition de pente. En moyenne, nous avons J=40 et I=120, et notre vocabulaire comporte N=10 mots. Nous évaluons donc 144000 calculs divisions.

Lorsque le relâchement des contraintes aux frontières est autorisé, le nombre de rapports à calculer augmente considérablement et devient $I.N.[(J-TF).3 + TF(N.TF + 0,5.TF + 0.5)]$ où TF est la taille moyenne d'une zone de flou. Dans notre cas, TF=5 et nous évaluons donc 456000 rapports, soit trois fois plus que pour l'algorithme simple.

La charge de calculs supplémentaires imposée par le relâchement des contraintes est relativement importante mais n'est malgré tout pas prohibitive.

CHAPITRE 3: QUANTIFICATION VECTORIELLE ET RECONNAISSANCE DE MOTS ENCHAINES

Il est bien évident que le problème du stockage des données se pose de la même façon en reconnaissance des mots isolés et en reconnaissance des mots enchainés. Introduire des méthodes de quantification vectorielle présente donc un avantage certain si l'on veut réduire la place mémoire occupée par les données.

Dans ce but, nous allons généraliser les techniques de reconnaissance de mots isolés avec quantification vectorielle que nous avons étudiées dans la partie C au cas des mots enchainés.

Considérons un vocabulaire V constitué de N formes de référence notées $R_1,....,R_N$. Nous disposons d'un codebook CB_i par mot du dictionnaire. Chaque codebook CB_i est constitué de n_i prélèvements v_j^i.

Chaque forme de référence R_k est codée comme une suite de prototypes du codebook CB_k qui lui correspond.

Soit T=(t(1),...,t(I)) la forme inconnue. La reconnaissance de la forme T va être réalisée à l'aide d'un algorithme semblable à celui que nous avons décrit dans le chapitre précédent, mais qui aura été adapté au cas de la quantification vectorielle.

Un point de l'espace de comparaison est représenté par ses trois coordonnées i,j,k qui signifient qu'en ce point le ième prélèvement de la forme test T est en coincidence avec le jème prélèvement de la forme de référence k. Lorsque l'on évaluera en ce point (i,j,k) les fonctions D(i,j,k), L(i,j,k) et les trois fonctions pointeurs FI, FJ, FK, nous coderons au préalable le prélèvement t(i) à l'aide du codebook CB_k en déterminant le prototype v_i^j le plus proche au sens de la distance de Hamming. Nous allons également déterminer en ce même point une autre distance que nous noterons DB(i,j,k) qui est la distorsion de codage en ajoutant à DB($\hat{i},\hat{j},\hat{k}$) la distance entre t(i) et v_i^j (($\hat{i},\hat{j},\hat{k}$) est le meilleur point précédent (i,j,k) sur le chemin de recalage partiel optimal aboutissant au point (i,j,k)).

Pour prendre la décision finale, nous ne minimiserons pas $D(I,Jk,k)/L(I,Jk,k)$ pour toutes les formes k de V, mais $\frac{D(I,Jk,k)}{L(I,Jk,K)} + DB(I,Jk,k)$ pour toutes les références k.

Nous pouvons maintenant spécifier l'algorithme de reconnaissance de mots enchainés avec quantification vectorielle.

1. SPECIFICATION DE L'ALGORITHME DE RECONNAISSANCE

(* la première étape est l'initialisation de toutes les variables utilisées *)

```
pour k = 1 à n faire

        D(0,0,k)  = 0;
        L(0,0,k)  = 1(0,0,0,0,0,k);
        FI(0,0,k) = 0;
        FJ(0,0,k) = 0;
        FK(0,0,k) = 0;
        DB(0,0k)  = 0;


        pour j = 1 à Jk faire

                D(0,j,k)  = ∞;
                DB(o,j,k) = 0;
                L(0,j,k)  = 1;
        finfaire

finfaire
```

(* la deuxième étape de cet algorithme est la programmation dynamique proprement dite *)

pour i = 1 à I faire

pour k = 1 à n faire

$$v_k^i = \underset{v_k^j \in CB_k}{\text{Argmin}} \text{ distance}(v_k^j, t(i))$$

pour j = 1 à Jk faire

$$(i,j,k) = \underset{(i',j',k') \in V(i,j,k)}{\text{Argmin}} \frac{D(i',j',k')+D(i',j',k',i,j,k)}{L(i',j',k')+L(i',j',k',i,j,k)}$$

$$D(i,j,k) = D(\hat{i},\hat{j},\hat{k}) + d(\hat{i},\hat{j},\hat{k},i,j,k)$$

$$L(i,j,k) = L(\hat{i},\hat{j},\hat{k}) + l(\hat{i},\hat{j},\hat{k},i,j,k)$$

$$DB(i,j,k) = DB(i,j,k) + \text{distance}(t(i), v_k^i)$$

$$\text{si } (i,j,k) \in Vi = \bigcup_{1 \leq k' \leq n} (i-1, Jk', k')$$

alors

$$FI(i,j,k) = FI(\hat{i},\hat{j},\hat{k})$$

$$FJ(i,j,k) = FJ(\hat{i},\hat{j},\hat{k})$$

$$FK(i,j,k) = FK(\hat{i},\hat{j},\hat{k})$$

sinon

$$FI(i,j,k) = \hat{i} ;$$

$$FJ(i,j,k) = \hat{j} ;$$

FK (i,j,k) = k ;

finsi

finpour

finpour

finpour

(* la troisième étape est la décision *)

kopt = $\underset{1 \leq k' \leq n}{\text{Min}} \frac{D(I,Jk',k')}{L(I,Jk',k')}$ = DB(I,Jk',k')

q(0) = kopt (* q est la table qui contiendra à la fin du retour arrière la superforme optimale en ordre inverse *)

k = 0

P(0) = (I,Jkopt,kopt)

Tant que (P_{k+1} = (FI(Pk),FJ(Pk),FK(Pk))=(0,0,0)) faire

q(k+1) = FK(Pk)

k=k+1

finfaire

nmot = k+1 (* nombre de mots reconnus *)

Pour i=1 à nmot faire

q'(i) = q(nmot-i) (* ordonne correctement la superforme *)

finfaire

(* le mot reconnu est *)

$T=R_{q'(1)}+\ldots+R_{q'(nmot)}$

fin.

2. IMPLANTATION ET TEST

Les codebooks ont été déterminés comme au chapitre 1 de la partie C. Cet algorithme a été implanté sur un MASSCOMP en langage C dans le cas du relâchement des contraintes et testé sur le même corpus décrit au chapitre 1 de la partie D. Le taux de reconnaissance est de 86 %.

Lorsque la quantification vectorielle est utilisée, la place mémoire occupée par le stockage des données est divisée par quatre, comme pour la reconnaissance de mots isolés.

CHAPITRE 4: CONTRAINTES SYNTAXIQUES

Nous présentons dans ce chapitre l'algorithme de Bridle et Nakagawa avec contraintes syntaxiques [Boyer 85] [di Martino 85]. De telles contraintes, puisqu'elles permettent de grouper les mots en classe, limitent les choix possibles à chaque étape du processus de comparaison. Nous espérons ainsi augmenter la taille du vocabulaire à taux de reconnaissance égal par rapport aux méthodes sans contraintes syntaxiques. Toutefois, ces améliorations seront obtenues au prix de contraintes supplémentaires pour le locuteur qui devra obligatoirement respecter la syntaxe définie.

Dans une première partie, nous allons décrire les contraintes syntaxiques dans les algorithmes de programmation dynamique avant de décrire dans une deuxième partie un algorithme les utilisant.

1. LES CONTRAINTES SYNTAXIQUES DANS UN ALGORITHME DE PROGRAMMATION DYNAMIQUE

Soit $V = \{ R_1,....,R_N \}$ l'ensemble des N mots du vocabulaire de l'application considérée. $R_1,....,R_N$ sont des symboles terminaux qui représentent une forme vocale.

Soit $S = \{ S_1,...,S_{NS} \}$ un ensemble fini d'états.

Dans la suite de ctte étude, nous appellerons transition t ou contexte d'une forme de référence défini par S et V un élément de l'ensemble SXVXS où X est le produit cartésien sur les ensembles:

$t = (S_i,k,S_j)$ où $(S_i,S_j) \in S^2$ et $k \in V$.

La forme de référence k est dite en contexte t.

Un réseau R(S,V) est un sous ensemble de SXVXS. La figure 1 donne un exemple d'un réseau à quatre états.

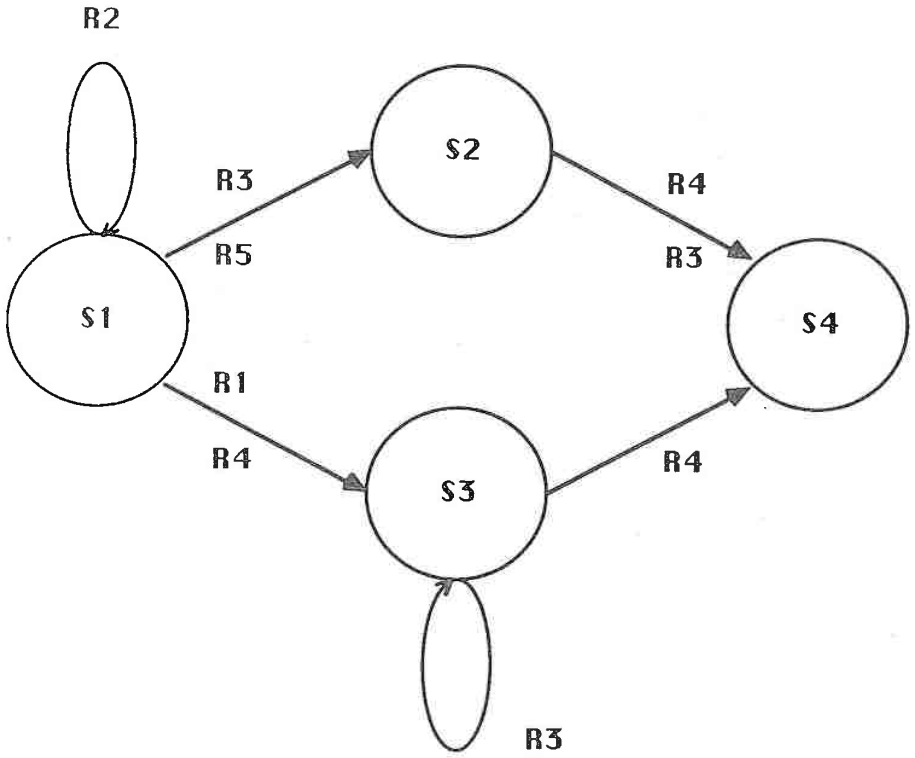


figure 1: exemple de réseau.

Les transitions de ce réseau $R(\{S_1,S_2,S_3,S_4\}),\{R_1,R_2,R_3,R_4,R_5\})$ sont:

$(S_1,R_2,S_1),(S_1,R_3,S_2),(S_1,R_5,S_2),(S_1,R_1,S_3),(S_1,R_4,S_3)$

$(S_2,R_1,S_2),(S_2,R_2,S_3),(S_2,R_4,S_4)$

$(S_3,R_3,S_3),(S_3,R_4,S_3)$

Définissons les trois projections élémentaires qui permettent d'accéder à chacun des éléments du triplet:

- projection P1:

$$R(S,V) \text{ ----> } S$$
$$t=(Si,k,Sj) \text{ ----> } Si$$

- projection P2:

$$R(S,V) \text{ ----> } V$$
$$t=(Si,k,Sj) \text{ ----> } k$$

- projection P3:

$$R(S,V) \text{ ----> } S$$
$$t=(Si,k,Sj) \text{ ----> } Sj$$

Nous dirons que $t \leqq t'$ où t et t' sont deux contextes d'un même réseau R(S,V) si P3(t)=P1(t').

Dans un réseau R(S,V), un chemin C[a,b] joignant les deux états a et b est une suite $(t_1,.....,t_m)$ de transitions qui vérifient $P1(t_1)$ =a et $P3(t_m)$=b et $t_1 \leqq t_2 \leqq ..... \leqq t_m$.

Soit CH[a,b] l'ensemble des chemins joignant a et b.

Soient Sin l'ensemble des noeuds initiaux et Sfi l'ensemble des noeuds finaux. Ces deux ensembles vérifient la propriété suivante:

$\forall S \in S$, $S_i \in Sin$, $S_f \in Sfi$, $c \in CH[S_i,S_j]$, $t \in c$, P1(t)=S ou P3(t)=S.

Pour exprimer les contraintes syntaxiques, nous avons besoin de deux fonctions:

- GAMMA qui donne tous les contextes antérieurs à un contexte donné

$$GAMMA: \quad R(S,V) \text{ ----> } P(R(S,V))$$
$$t \text{ ----> } \{t' \in R(S,V)/t' \leqq t\}$$

(P(R(S,V)) est l'ensemble des parties de R(S,V))

- PHI qui rend tous les contextes d'une forme de référence k

$$\text{PHI:} \quad V \longrightarrow P(R(S,V))$$
$$k \longrightarrow \{t \in R(S,V) / P2(t)=k\}$$

Maintenant que nous avons fixé nos notations, nous allons pouvoir déterminer les relations de programmation dynamique avec contraintes syntaxiques.

2. PROGRAMMATION DYNAMIQUE AVEC CONTRAINTES SYNTAXIQUES

Si l'on introduit des contraintes syntaxiques dans l'algorithme de reconnaissance des mots enchainés, la troisième coordonnée k d'un point (i,j,k) qui indique la forme de référence doit être remplacée par le contexte t. Dans cet espace, un point est représenté par le triplet (i,j,t) qui signifie qu'en ce point, le ième prélèvement de la forme inconnue T est mis en coincidence avec le jème prélèvement de la forme de référence qui se trouve en contexte t. La relation de programmation dynamique s'écrit de la même façon que précédemment:

$$(\hat{i},\hat{j},\hat{t}) = \underset{(i',j',t')\in V(i,j,t)}{\text{argmin}} \frac{D(i',j',t') + d(i',j',t',i,j,t)}{L(i',j',t') + l(i',j',t',i,j,t)}$$

$$D(i,j,t) = D(\hat{i},\hat{j},\hat{t}) + d(\hat{i},\hat{j},\hat{t},i,j,t)$$
$$L(i,j,t) = L(\hat{i},\hat{j},\hat{t}) + l(\hat{i},\hat{j},\hat{t},i,j,t)$$

où V(i,j,t) est le voisinage du point (i,j,t) défini par la contrainte locale utilisée.

Nous allons expliciter V(i,j,t) dans le cas où l'on n'autorise pas le relâchement des contraintes pour des raisons de clarté: il est bien évident que ces voisinages se généralisent sans difficulté au cas où les zones de flou sont permises.

Nous allons reprendre le formalisme introduit par Ney et l'adapter au cas des contraintes syntaxiques. La contrainte locale utilisée est la contrainte sans condition de pente.

Dans le cas d'une contrainte interne, (i-1,j,t),(i-1,j-1,t),(i,j-1,t) appartiennent à la même forme dans le contexte t que (i,j,t), c'est à dire j>1, le voisinage V(i,j,t) s'écrit:

$$V(i,j,t) = \{(i-1,j,t),(i-1,j-1,t),(i,j-1,t)\}$$

Dans le cas d'une contrainte externe, nous avons:

$$V(i,1,t) = \{(i-1,1,t)\} \underset{t' \in GAMMA(t)}{\cup} \{ (i-1,J_k,t') \}$$

où k est la forme de référence en contexte k'.

Les fonctions de chainage arrière doivent être redéfinies puisque nous avons généralisé une dimension de l'espace de comparaison.

Soient FI,FJ,FT les trois fonctions de chainage arrière qui sont nécessaires pour définir les trois coordonnées du point P.

Pour pouvoir remonter la meilleure superforme de référence, il est nécessaire de déterminer le point final du chemin de recalage optimal. Etant donné que nous avons introduit des contraintes syntaxiques, nous ne devons considérer lors de la minimisation de $\frac{D(I,Jk,t)}{L(I,Jk,t)}$ que les références dont le contexte t est tel que P3(t) appartient à l'ensemble Sfi des noeuds finaux.

Toutes ces considérations étant faites, nous pouvons maintenant spécifier l'algorithme de reconnaissance des mots enchainés avec contraintes syntaxiques.

3. SPECIFICATION DE L'ALGORITHME DE RECONNAISSANCE AVEC CONTRAINTES SYNTAXIQUES

```
(* initialisation *)
pour k = 1 à n faire

    pour t ∈ PHI(k) faire

        pour j=1 à J_P2(t)

            FI(1,j,t) = 0;
            FJ(1,j,t) = 0;
            FT(i,j,t) = 0;
            si j=1 et P1(t) ∈ Sin alors  D(1,j,t) = d(o,o,o,1,j,t)
                                         L(1,j,t) = l(0,0,0,1,j,t)
                                 sinon   D(1,j,t) = ∞
                                         L(1,j,t) =1
            finsi
        finfaire
    finfaire
finfaire

( * la deuxième étape de cet algorithme est la programmation dynamique
proprement dite * )

pour i = 1 à I faire

     pour k = 1 à n faire

          pour j = 1 à Jk faire

               pour t ∈ PHI(k) faire
```

$$(\hat{i},\hat{j},\hat{t}) = \underset{(i',j',t')\in V(i,j,t)}{\text{Argmin}} \frac{D(i',j',t')+D(i',j',t',i,j,t)}{L(i',j',t')+L(i',j',t',i,j,t)}$$

$D(i,j,t) = D\ (\hat{i},\hat{j},\hat{t}) + d(\hat{i},\hat{j},\hat{t},i,j,t)$

$L(i,j,t) = L\ (\hat{i},\hat{j},\hat{t}) + 1(\hat{i},\hat{j},\hat{t},i,j,t)$

si $(\hat{i},\hat{j},\hat{t}) \in Vi = \{(i-1,j,t),(i-1,j-1,t),(i,j-1,t)\}$

alors

$FI\ (i,j,t) = FI\ (\hat{i},\hat{j},\hat{t})$

$FJ\ (i,j,t) = FJ\ (\hat{i},\hat{j},\hat{t})$

$FT\ (i,j,t) = FK\ (\hat{i},\hat{j},\hat{t})$

sinon

$FI\ (i,j,t) = \hat{i}$;

$FJ\ (i,j,t) = \hat{j}$;

$FT\ (i,j,t) = \hat{t}$;

finsi

finpour

finpour

finpour

finpour

(* la troisième étape est la décision *)

$$topt = \min_{t' \in [t/P3(t) \in Sfi]} \frac{D(I, J_{P2(t')}, t')}{L(I, J_{P2(t')}, t')}$$

q(0) = P2(topt)=kopt (* q est la table qui contiendra à la fin du retour arrière la superforme optimale en ordre inverse *)

k = 0

P(0) = (I,Jkopt,topt)

Tant que (P_{t+1} = (FI(Pt),FJ(Pt),FT(Pt))=(0,0,0)) faire

q(t+1) = FT(Pt)

k=k+1

finfaire

nmot = k+1 (* nombre de mots reconnus *)

Pour i=1 à nmot faire

q'(i) = q(nmot-i) (* ordonne correctement la superforme *)

finfaire

(* le mot reconnu est *)

$T = R_{q'(1)} + \ldots + R_{q'(nmot)}$

fin.

CHAPITRE 5:LIAISONS

Lorsque Sakoe a introduit son modèle de reconnaissance de mots enchainés. il a fait l'hypothèse implicite que la coarticulation n'entraine pas de modifications importantes des formes en contexte par rapport aux références (qui résultent de l'élocution d'élocutions isolées). Or il arrive fréquemment que les zones de début et de fin des mots soient modifiées de façon significative lorsque ceux ci sont prononcés dans un continuum de parole du fait de phénomènes de phonologie et de coarticulation.

Prenons un exemple. Considérons les deux mots "DIX" et "NEUF".Or lorsque deux consonnes se suivent, la première prend le voisement de la deuxième sauf si c'est un R ou un L. Donc le s de dix qui est non voisé se voise au contact du n de neuf et devient z. Or les systèmes de reconnaissance de mots enchainés ne permettent pas de tenir compte de telles modifications, ce qui altèrent sensiblement leurs performances.

Le même problème se produit pour les liaisons entre les mots. Des phonèmes apparaissent lorsque certains mots sont prononcés à la suite, alors qu'ils ne figurent pas dans l'élocution isolée.

Prenons l'exemple de la phrase "LES ENFANTS SONT A L'ECOLE" qui sans liaisons donne la transcription phonétique suivante "leãfãsõalecol" alors que la prononciation correcte conduit à "lezãfãsõtalecol".

Si l'on veut que les algorithmes de reconnaissance de mots enchainés donnent des résultats satisfaisants, il faut qu'ils tiennent compte des problèmes de liaison.

Pour tenir compte des phénomènes de coarticulation ou de liaison, nous avons distingué trois zones dans un mot, une zone pouvant éventuellement être vide. Un mot MOT résulte de la concaténation des trois parties MOT1, MOT2, MOT3.

Il apparait clairement que MOT1 et MOT3 ne sont pas uniques mais dépendent du mot qui les précède (pour MOT1) ou qui les suit (pour MOT3).

Nous dirons plus précisément que MOT en contexte MOT' MOT" (c'est-à-dire la suite MOT' MOT MOT" a été prononcée) résulte de la concaténation de 3 parties MOT1(MOT'), MOT2, MOT3(MOT") où MOT1(MOT') est la partie de MOT modifiée par coarticulation arrière avec MOT' et MOT3(MOT") est la zone de MOT modifiée par coarticulation avant avec MOT".

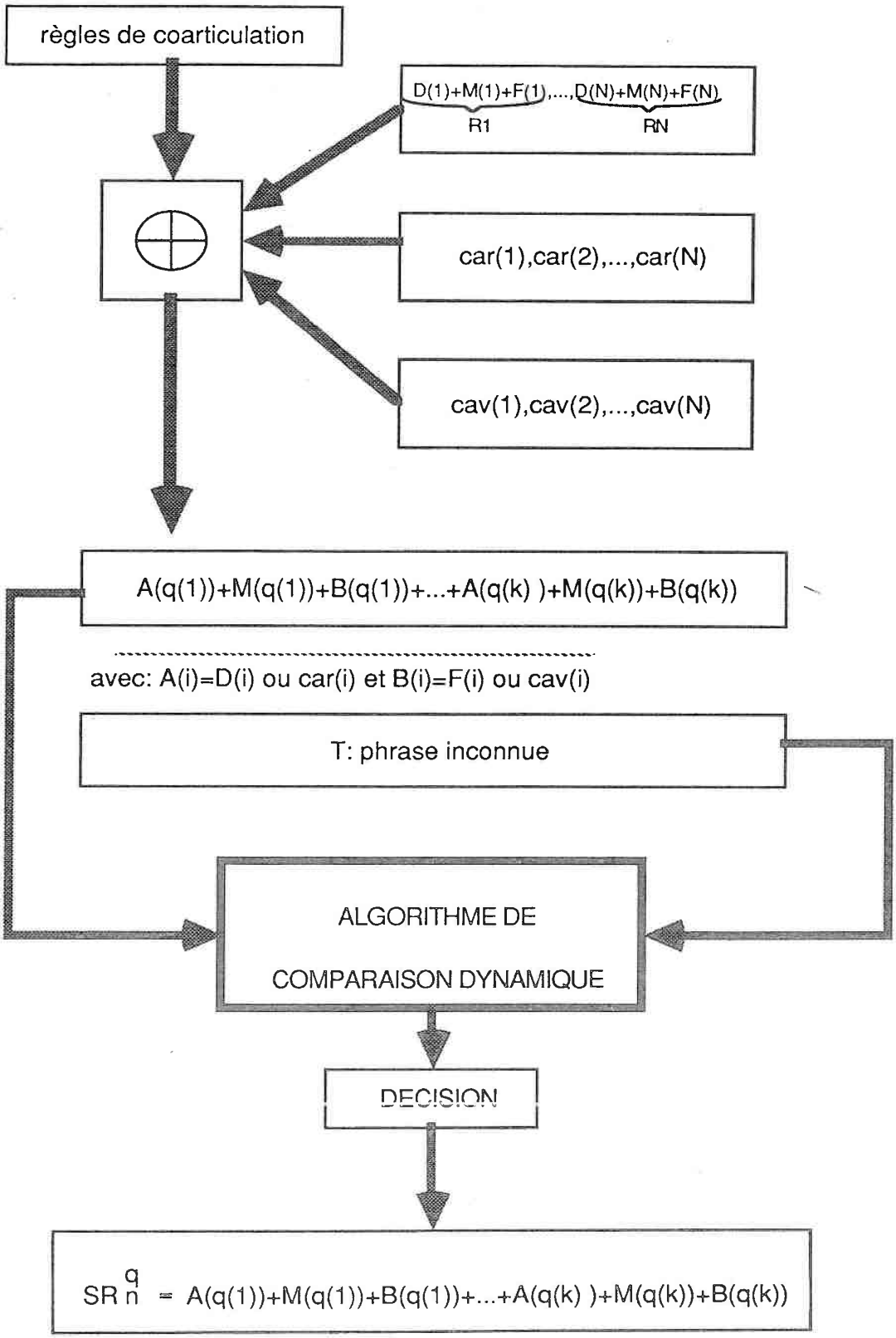
di Martino [di Martino 84] a fait l'hypothèse que les modifications introduites par coarticulation avant ou par coarticulation arrière ne dépendent pas du contexte et il n'a associé à chaque zone MOT1 et MOT3 qu'une seule forme de déformation. Cette hypothèse est réaliste dans le cas où le vocabulaire est limité mais ne cesse de l'être dans le cas d'un grand vocabulaire car elle devient alors trop restrictive.

di Martino a généralisé le modèle de Sakoe comme l'indique la figure 1. Il a utilisé l'algorithme de Meyers (décrit dans le chapitre 1 de la partie D). Il a introduit les coarticulations avant et arrière dans la comparaison dynamique de la façon suivante:

- l'algorithme de programmation dynamique décide dans un premier temps si il y a lieu de tenir compte d'un effet de coarticulation au point (i,j,k,n) (n est le niveau de construction) en vérifiant que le couple de formes constitué par la dernière forme de référence de la meilleure superforme de n-1 mots qui a permis de construire le chemin optimal aboutissant au point (i,j,k,n) et la forme k unifie une règle de coarticulation avant ou arrière,

- si la décision précédente est affirmative, l'algorithme de programmation dynamique valide ou non dans un deuxième temps l'hypothèse de coarticulation au point (i,j,k,n) en associant au chemin optimal aboutissant en ce point le minimum des distances cumulées que l'on obtient en tenant compte ou pas d'un effet dû à la coarticulation.

figure 1: modèle de Sakoe généralisé.



Comme nous l'avons déjà signalé précédemment, l'hypothèse qui considère que la modification ne dépend pas du contexte, nous parait trop restrictive dans le cas d'un vocabulaire important.

Aussi proposons nous un algorithme qui permet de considérer que la modification d'un mot par coarticulation ou par liaison dépend du contexte qui l'entoure. Pour cela, nous allons reprendre l'algorithme avec contraintes syntaxiques décrit dans le chapitre précédent et le modifier pour l'adapter aux problèmes de coarticulation et de liaison.

Un point de l'espace de comparaison est représenté par ses trois coordonnées i,j,t qui indiquent qu'en ce point le i ème prélévement de la forme inconnue T est en coincidence avec le j ème prélèvement de la forme de référence k en contexte t.

Nous appelerons réseau de coarticulation un réseau R(S,V,Re) où S est un ensemble d'états , V un vocabulaire et Re l'ensemble des règles de coarticulation et de liaisons qui peuvent s'appliquer sur les mots de V.

Un exemple d'un tel réseau est donné figure 2. Nous allons expliciter davantage cette notion .

Soient V = { R_1,..,R_N } un vocabulaire composé de N mots R_1,..,R_N et S= { S_1,S'_1,..,S_{Ns},S'_{Ns},Sin,Sfi } un ensemble d'états où Sin est l'état initial et Sfi l'état final. Soit Re = {règle1,..,règler} l'ensemble des r règles de coarticulation ou de liaisons qui peuvent s'appliquer.

Une transition aboutit à l'état S_i si le mot R_i a été prononcé sans modification du début de R_i par coarticulation arrière ou par liaison avec le mot qui précède. S_i sera appelé un état de non coarticulation.

Une transition aboutit à l'état S'_i si le mot R_i a subi une modification de sa première zone par coarticulation arrière ou liaison avec le mot qui précède. S'_i sera appelé un état de coarticulation.

Un contexte est défini par un triplet (s_i,règlek,s_j) où s_i et s_j sont deux états de coarticulation ou non et règlek est la règle de coarticulation qui s'applique lorsque le mot R_j est prononcé à la suite du mot R_i. Cette règle peut éventuellement être la règle "vide" si aucune coarticulation et aucune liaison ne se produisent.

Nous avons toujours besoin des trois fonctions projections élémentaires:

- projection P1:

```
        SXRXS          ----> S
(si,reglek,sj)         ----> si
```

- projection P2:

```
        SXRXS          ----> B
(si,reglek,sj)         ----> reglek
```

- projection P3:

```
        SXRXS          ----> S
(si,reglek,sj)         ----> sj
```

La relation $\leqq$ est celle que nous avons défini dans le chapitre précédent.

Le mot prononcé est indiqué par l'indice j de l'état d'arrivée sj.

L'algorithme de reconnaissance est identique à celui que nous avons développé au chapitre 4, si ce n'est que lors de la transition (si,reglek,sj), le mot mis en comparaison avec la forme test n'est plus $MOT1_j MOT2_j MOT3_j$ mais $MOT1_i(reglek)MOT2_j MOT3_j(reglek)$.

Signalons enfin que lorsque les règles sont bien définies pour un vocabulaire donné, le réseau se construit automatiquement.

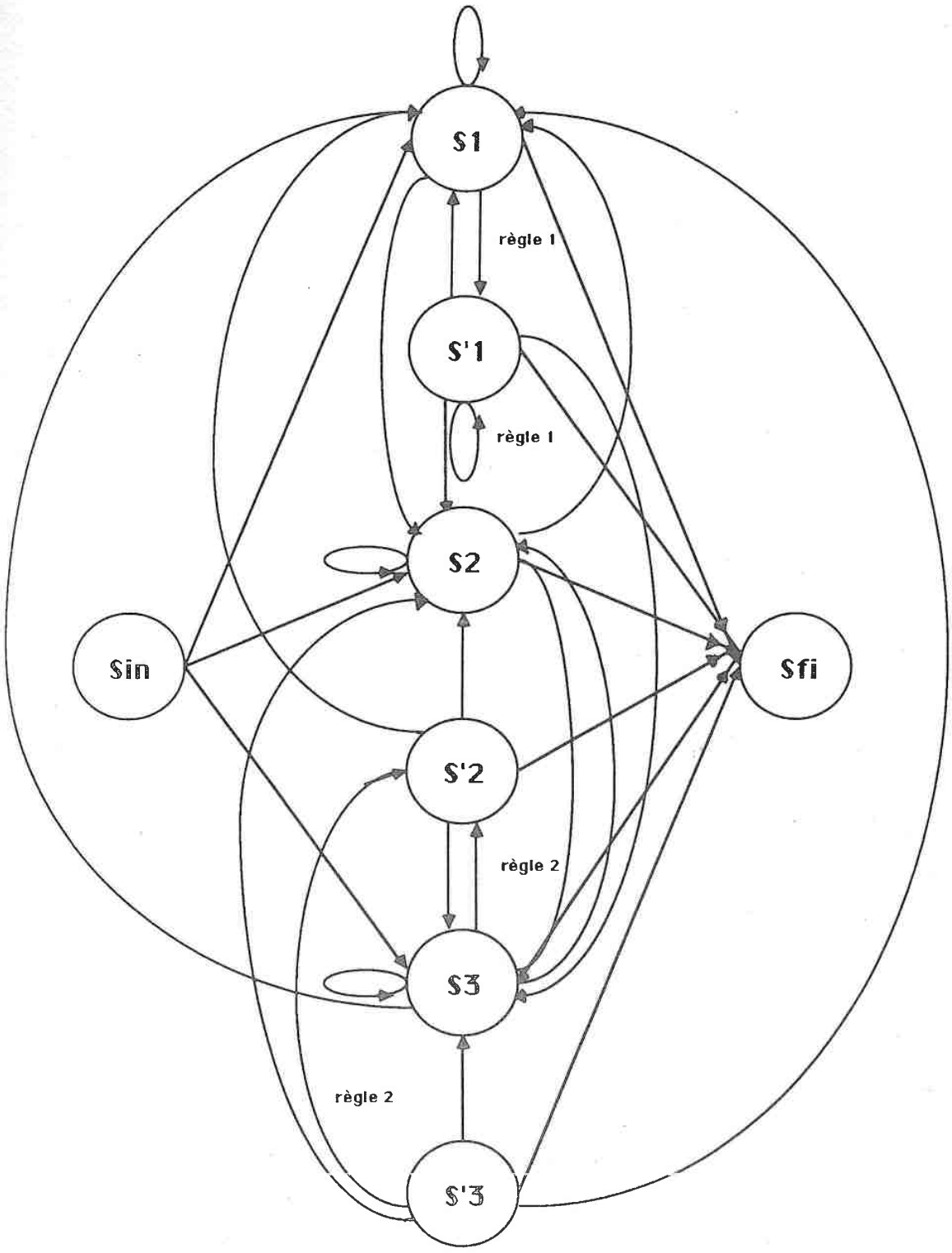


figure 2: exemple de réseau de coarticulation.

CHAPITRE 6 : APPROCHE SEMI-GLOBALE

Les algorithmes que nous avons décrits dans les chapitres précédents n'utilisent aucune information sur la structure des mots comparés. Il nous a paru intéressant de tenir compte d'informations plus fines issues directement du signal de parole et de réaliser un premier étiquetage grossier du signal en grandes classes de phonèmes. Pour cela, nous avons repris les procédures de segmentation grossière et d'étiquetage général mises au point par Fohr ([Fohr 86]) dans le cadre du projet de système expert en décodage acoustico-phonétique APHODEX. Les classes que nous avons retenues sont:

- les voyelles (détectées par NOVOCA),
- les fricatives (déterminées par FRICA),
- les autres phonèmes sont classés "inconnus".

1. PRETRAITEMENT

Nous avons modifié la partie prétraitement du système de reconnaissance de la façon suivante:

- le mot prononcé est analysé par le banc de filtres 32 canaux décrits dans le chapitre 1 de la partie D, le résultat fourni est donc une suite de vecteurs de 32 composantes,
- le signal correspondant à ce mot est ensuite étiqueté en classes générales de phonèmes grâce aux procédures NOVOCA et FRICA. A l'issue de ce traitement, une donnée supplémentaire est fournie qui indique si un prélèvement correspond à une voyelle, à une fricative ou est inconnu. Cette indication supplémentaire est ajoutée au vecteur de 32 composantes correspondant à ce prélèvement et constitue

une trente troisième composante.

Les formes de référence sont ainsi traitées et les suites de vecteurs de 33 composantes correspondantes sont rangées en mémoire.

2. ALGORITHME DE RECONNAISSANCE

Lors de la phase de reconnaissance, le mot inconnu est prononcé par le locuteur devant le microphone et le signal acoustique est numérisé puis analysé et codé par le banc de filtres. L'étiquetage en trois classes de phonèmes (voyelles, fricatives, inconnus) est réalisé et à l'issue de cette étape, le mot est représenté par une suite de vecteurs de 33 composantes.

Lors de la comparaison, l'algorithme de programmation dynamique utilisé est celui que nous avons étudié dans le premier chapitre de cette partie. Il a cependant fallu modifier le calcul de la distance locale car les vecteurs comparés ont 32 composantes qui correspondent à l'analyse banc de filtres et une dernière composante qui représente un étiquetage grossier.

La distance a été calculée de la manière suivante:

- la distance de Hamming classique a été déterminée pour les 32 premières composantes du vecteur. De plus, comme nous l'avons signalé dans le chapitre 1, si un vecteur est comparé à un prélèvement de silence, si au moins une composante du vecteur inconnu est supérieure à un seuil donné, une pénalité est ajoutée à la valeur calculée de la distance,

- si les deux prélèvements ont leur dernière composante qui diffère, une pénalité est ajoutée à la distance de Hamming. Si l'un des prélèvements est de type voyelle et l'autre non, une pénalité

"voyelle" est ajoutée à la valeur de la distance; si l'un est classé comme fricative et l'autre pas, une pénalité "fricative" est ajoutée.

Nous disposons donc maintenant de trois pénalités:

- une pénalité "silence",
- un pénalité "fricative",
- une pénalité " voyelle".

Les valeurs de ces pénalités ont été fixées ainsi:

- les pénalités "voyelle" et " fricative" ont été fixées à la valeur moyenne de la distance entre deux prélèvements quelconques du corpus d'apprentissage,
- la moitié de cette valeur moyenne a été attribuée à la pénalité "silence" .

3. IMPLANTATION ET TEST

Cet algorithme a été implanté en langage C sur Masscomp et testé sur le corpus correspondant aux deux locuteurs entrainés. Le taux de reconnaissance moyen obtenu est de 93% et constitue donc une amélioration notable par rapport à ce que ces deux mêmes locuteurs avaient obtenu pour l'algorithme donné au chapitre 1 (91,1%).

Ce résultat confirme que l'introduction d'informations plus fines dans le processus de comparaison dynamique permet d'améliorer les performances de la reconnaissance.

CONCLUSION

La reconnaissance des mots enchaînés pose plus de problèmes que la reconnaissance des mots isolés. Elle présente davantage d'interêt et ses applications pratiques sont nombreuses (éditeur oral, interrogation de bases de données....), car l'élocution sans pauses artificielles entre les mots est plus naturelle que la prononciation en mots isolés.

Nous avons pu constater que l'introduction des zones de flou améliore sensiblement les performances de la reconnaissance : en effet, le relâchement des contraintes sur les deux axes permet de tenir compte en partie des phénomènes de coarticulation à la frontière des mots.

IL est relativement aisé d'introduire des contraintes syntaxiques dans cet algorithme : l'utilisation d'une grammaire impose davantage de contraintes au locuteur qui doit respecter la syntaxe introduite, mais elle réduit le nombre de comparaisons et permet d'envisager la reconnaissance d'un langage artificiel.

L'utilisation de la quantification vectorielle permet également de réduire la quantité d'informations à stocker. Elle ne se réalise pas au détriment des performances de reconnaissance.

L'avant-dernier algorithme décrit (chapitre 5) laisse espérer une amélioration du taux de reconnaissance : en effet, il permet de tenir compte des phénomènes de coarticulation et de liaison à la frontière des mots qui étaient négligés dans les précédents algorithmes.

Le dernier algorithme présenté (chapitre 6) tient compte d'informations grossières sur la structure des mots qu'il compare. Il réalise une segmentation grossière et un étiquetage en grandes classes de phonèmes du signal inconnu et il utilise ces données lors de la comparaison. L'amélioration des performances constatée montre que ces informations permettent de corriger certaines erreurs commises lors de la comparaison dynamique.

CONCLUSION

CONCLUSION

La reconnaissance de mots isolés et de mots enchaînés est un problème complexe qui n'est encore qu'imparfaitement résolu.

Pour le traiter, nous avons employé la technique de programmation dynamique qui semble à l'heure actuelle être une solution efficace à la reconnaissance d'un vocabulaire limité (partie B). Nous avons étudié quelques algorithmes qui utilisent la programmation dynamique en reconnaissance des dix chiffres isolés et nous avons comparé leurs performances (taux de bonne reconnaissance et temps d'exécution). A temps de calcul égal, le relâchement des contraintes améliore le score de reconnaissance et semble être le mieux adapté pour reconnaitre les dix chiffres français prononcés isolément.

Nous avons ensuite introduit la quantification vectorielle (partie C) qui permet de réduire la place mémoire occupée par le stockage des données. Une rapide estimation nous a permis de constater que pour notre application, la quantification vectorielle a permis de diminuer de 75% le taux d'occupation de la mémoire par les données. Nous avons essayé diverses techniques de reconnaissance qui font appel à la programmation dynamique. Deux méthodes ont obtenu des scores de reconnaissance intéressants:

- lorsque l'algorithme de programmation dynamique est paramétré par des pondérations variables qui dépendent du couple de prototypes comparés et lorsque la décision finale s'effectue par minimisation de la somme du score de programmation dynamique et de la distorsion de codage, le taux de bonne reconnaissance s'élève à 97,5% pour deux locuteurs. Ceci améliore de 2% le score obtenu si l'on ne tient pas compte de pondérations variables, à temps d'exécution égal.

- la minimisation s'effectue sur la somme du score de programmation dynamique déterminé avec la pondération (1,2,1), de la distorsion de codage et des probabilités de présence des prototypes à la place où ils apparaissent dans le mot. Le score de bonne reconnaissance

obtenu est de 98%, ce qui constitue une amélioration de 0,5% sur le résultat obtenu par le précédent algorithme, à temps de calcul sensiblement égal.

Ces deux résultats nous permettent de constater que l'introduction de la quantification vectorielle ne diminue pas les performances de la reconnaissance.

Nous avons ensuite appliqué la technique de programmation dynamique à la reconnaissance des mots enchainés (partie D). Nous avons étudié l'algorithme proposé par Bridle et Nakagawa et vu comment il peut s'étendre au cas du relâchement des contraintes sur les deux axes d'une part et comment il peut prendre en compte des contraintes syntaxiques d'autre part. Signalons que l'introduction de zones de flou améliore sensiblement les performances de la reconnaissance. Nous avons proposé un algorithme de reconnaissance de mots enchainés qui permet de prendre en considération les phénomènes de coarticulation et de liaison qui apparaissent à la frontière de mots prononcés sans pauses intermédiaires.

La quantification vectorielle a permis de réaliser un gain de place mémoire et de temps de calcul (toutes les distances inter-prototypes étant précalculées). Nous avons constaté que ces améliorations ne s'effectuent pas au détriment des performances de reconnaissance.

Cette étude met en évidence que la programmation dynamique et la quantification vectorielle constituent une approche performante pour traiter efficacement le problème de la reconnaissance des mots isolés et des mots enchainés.

Une autre solution au problème de l'alignement temporel des formes acoustiques est obtenue en représentant chaque référence sous la forme d'un graphe où les noeuds représentent les états d'un processus de Markov. Nous avons constaté que les trois transitions possibles d'un modèle de Markov (retour d'un état vers lui même, transition vers l'état suivant avec émission d'un prélèvement, transition vers l'état suivant sans émission de prélèvement) correspondent aux trois transitions

locales de la contrainte sans condition de pente que nous avons utilisée dans nos algorithmes de programmation dynamique.

Ainsi les coefficients (a,b,c) de la contrainte sans condition de pente peuvent être associés à un état d'un modèle de Markov -il y a dans ce cas autant d'états que de triplets- qui lui même peut être associé à une liste de vecteurs prototypes. Cette dernière solution modèlise plus convenablement la structure temporelle des formes vocales, structure temporelle qui est perdue lorsqu'une liste de prototypes est associée à un état d'un modèle de Markov.

Les résultats obtenus par l'algorithme à pondérations variables qui associe à chaque couple de prototypes (i,j) un triplet (a(i,j),b(i,j),c(i,j)) montre une amélioration du taux de reconnaissance par rapport à celui réalisé par le même algorithme avec des pondérations constantes.

Nous envisageons dans la poursuite de ce travail de représenter chaque mot du vocabulaire par un treillis de prélèvements. Ceci permettra de stocker davantage de formes de références à place mémoire équivalente.

Les performances obtenues par l'algorithme décrit au chapitre 6 de la partie D montrent qu'il est intéressant d'utiliser des procédures de segmentation grossière (fournies par exemple par le système APHODEX [Fohr 86]) afin de segmenter le signal inconnu en classes phonétiques.

Une solution au problème de la reconnaissance de grands vocabulaires par exemple est que la forme inconnue, dont le patron phonétique grossier aura été ainsi déterminé, ne soit comparée par programmation dynamique qu'aux formes de référence possédant un patron phonétique analogue.

Une deuxième approche peut être dans le cas d'un vocabulaire difficile de sélectionner par programmation dynamique un certain nombre de candidats à la reconnaissance. Par exemple, dans le cas du vocabulaire constitué par les lettres de l'alphabet français, le b et le d ne pourront pas être distingués par programmation dynamique. Les deux formes

candidates seront alors b et d. Une analyse plus fine, sur le burst dans ce cas, sera alors réalisée sur la forme inconnue et sera discriminante. En général, une seule analyse fine suffit et elle sera d'autant plus facile que, dans le cas du b et du d par exemple, elle ne cherche pas à reconnaitre les phonèmes "b" et "d" dans tous les contextes mais uniquement en contexte "e".

L'utilisation complémentaire d'une analyse fine (telle qu'elle est pratiquée dans une approche analytique) et de la programamtion dynamique semble prometteuse pour la reconnaissance de mots appartenant à un vocabulaire difficile.

BIBLIOGRAPHIE

- [Adoul 85]: Adoul J.P., La quantification vectorielle d'ondes: approche algébrique,Séminaire GALF, pp.1-52, Paris 1985.

- [Baum 1972]: Baum L.E., An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process, Inequalities, 3, 1-8, 1972.

- [Bellman 57]: Bellman J., Dynamic programming, Princeton University Press, 1957.

- [Benani 84]: Benani M., Adjonction d'E/S vocales et graphiques en E.A.O.: amélioration du dialogue et nouveaux champs d'application, Thèse de 3 ème cycle, Université de Nancy 1,1984.

- [Bogert 63]: Bogert B.P., Healy M.J.R., Tuckey J.W., The frequency analysis of time-series for echoes, Proc. Symp. Time Series Analysis, M. Rosenblatt editor, chap. 15, pp 209-243, 1963.

- [Bonneau 85]: Bonneau H., Eskenazi M., Mariani J., Utilisation du codage vectoriel en reconnaissance de mots isolés, Séminaire GALF, pp.169-176, Paris 1985.

- [Boyer 84]: Boyer A., Etude d'algorithmes de reconnaissance de mots enchainés, Rapport de DEA, Université de Nancy 1, 1984.

- [Boyer 85]: Boyer A., Haton J.P., di Martino J., Un algorithme de reconnaissance de mots enchainés avec contraintes syntaxiques, 14 ème JEP, Paris 1985.

- [Boyer 86]: Boyer A., Haton J.P., di Martino J., Reconnaissance de la parole multilocuteur par la programmation dynamique, 15 ème JEP, Aix en Provence 1986.

- [Bridle 82]: Bridle J.S., Brown MD., Chamberlain R.M., An algorithm for connected word recognition, ICASSP, pp.899-902, Paris, 1982.

- [Burton 83]: Burton D.K., Shore J.E., Buck J.T., A generalization of isolated word recognition using vector quantization, ICASSP, Boston,pp. 1021-1024, 1983.

- [Burton 84]: Burton D.K., Buck J.T., Shore J.E., Parameter selection for isolated word recognition using vector quantization, ICASSP, San Diego,1984.

- [Burton 85]: Burton D.K., Applying matrix quantization to isolated word recognition, ICASSP, Tampa, pp.29-32, 1985.

- [Caelen 79]: Caelen J., "Un modèle d'oreille. Analyse de la parole continue. Reconnaissance phonémique", thèse d'état, Toulouse, 1979.

- [Charpillet 85]: Charpillet F.,"Un système de reconnaissance de parole continue pour la saisie de textes lus",Thèse de l'Université de Nancy 1, 1985.

- [Das 80]: Das S.K., Some experiments in discrete utterance recognition, ICASSP, Denver, pp.178-181, 1980.

- [Davis 80]: Davis S.B., Mermelstein P., Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, IEEE Trans. on acoustics, speech and signal proc., vol. ASSP-28, no 4, août 1980.

- [Divoux 85]: Divoux P., Haton J.P., Classification hiérarchique et reconnaissance multi-locuteur,Congrès AFCET-RFIA 1985, Grenoble, pp. 309-318.

- [Dumas 85]: Dumas J., rapport de stage de fin d'étude ISIN, 1985.

- [Ferretti]: Ferretti M., Cinare F., Synthèse, reconnaissance de la parole, Editest 1983.

- [Flanagan 72]: Flanagan J.L., Speech analysis,synthesis and perception, 2nd. ed., Springer Verlag, New-York, 1972.

- [Fohr 86]: Fohr D., Aphodex: un système expert en décodage acoustico-phonétique de la parole continue, Thèse de l'Université de Nancy 1, 1986.

- [Forney 73]: Forney G.D.,Jr., The Viterbi algorithm, Proc. IEEE., Vol. 61, N.3, 1973.

- [Gauvain 82]: Gauvain J.L.,Mariani J., A method for connected word recognition and word spotting on a microprocessor, ICASSP, pp.891-894,Paris, France, 1982.

- [Gauvain 85]: Gauvain J.L., Techniques de reconnaissance globale, Séminaire GALF, Paris, pp. 106-135, 1985.

- [G.C.P.-SFA 87]: Groupe Communication Parlée de la SFA, La parole et son traitement automatique, à paraitre dans la collection CNET-ENST.

- [Gong 86]: Gong Y.,Haton J.P., Time domain harmonic matching pitch estimation using timer dependent speech modeling, à paraître dans IEEE Trans. A.S.S.P, 1987.

- [Guibert 79]: Guibert J., La parole: compréhension et synthèse par les ordinateurs, PUF, 1979.

- [Haltsonen 84]: Haltsonen S., An endpoint relaxation method for dynamic time warping algorithms, ICASSP,San Diego, pp 981- 984, 1984.

- [Haton 74]: Haton J.P., Contribution à l'analyse, la paramétrisation et la reconnaissance de la parole, Thèse d'état, Université de Nancy I, 1974.

- [Haton 85]: Haton M.C., Contribution à l'éducation vocale assistée par ordinateur: étude des voix et réalisation du système SIRENE, Thèse d'état, Université de Nancy I, 1985.

- [Hess 83]: Hess W.J., Pitch determination of speech signals- algorithms et devices , Springer Verlag , Berlin, 1983.

- [Itakura 75]: Itakura F., Minimum production residual principle applied to speech recognition, IEEE Trans. ASSP Vol.23, pp 67-72, février 1975.

- [Jelinek 76]: Jelinek F, Continuous speech recognition by statistical methods, Proc. IEEE vol 64 no 4, avril 1976.

- [Klatt 80]: Klatt D.H., Speech perception: a model of acoustic-phonetic access; in Perception and production of fluent speech, L. Erlbaum Assoc., Hillsdale, 1980.

- [Kopec 85]: Kopec G., Bush M., Network-based isolated digit recognition using vector quantization, IEEE Trans. ASSP, Vol 33 no 4, août 1985.

- [Lazrek 83]: Lazrek M., Décodage acoustico-phonétique en compréhension automatique de la parole continue, Thèse de troisième cycle, Université de Nancy I, 1983.

- [Lee 85]: Lee K.F., Incremental network generation in template-based word recognition, CMU-CS-85-181, 1985.

- [Levinson 83]: Levinson S.E., Rabiner L.R., Sondhi M.M., speaker independant isolated digit recognition using hidden Markov models, ICASSP, Boston, pp 1049-1052, avril 1983.

- [Levinson 84]: Levinson S.E, Liberman M., La reconnaissance de la parole par ordinateur, pp 87-100, dans L'intelligence de l'informatique, Pour la science, Belin, 1984.

- [Lienard 77]: Lienard J.S., Les processus de la communication parlée, Editions Masson, 1977.

- [Linde 80]: Linde Y., Buzo A., Gray R.M., "An algorithm for vector quantizer design", IEEE Trans. Commun. Technol., vol COM-28, janvier 1980.

- [Lloyd 57]: Lloyd S.P., Least squares quantization in PCM, IEEE Trans. IT, vol 28, mars 1982.

- [Lonchamp]: Lonchamp F., Cours à l'Université de Nancy 2, 1985.

- [Mari 85]: Mari J.F., Reconnaissance de mots enchainés à l'aide de modèles markoviens discrets, 5 ème congrès AFCET-RFIA, pp 859-867, Grenoble, 1985.

- [Markel 76]: Markel J.D.,Gray A.H., Jr., Linear production of speech, Springer Verlag, Berlin, 1976.

- [di Martino 83]: di Martino J., Haton J.P., Haton M.C., Evaluation d'algorithmes en reconnaissance automatique de la parole, 11 th international congress on acoustics, Paris, France, pp 192-202, juillet 1983.

- [di Martino 84]: di Martino J., Contribution à la reconnaissance globale de la parole: mots isolés et mots enchainés, Thèse de docteur ingénieur en informatique, Université de Nancy I, 1984.

- [di Martino 85]: di Martino J., Dynamic time warping algorithms for isolated and connected word recognition, in Nato Asi series, vol F16, New Systems and architectures for automatic speech recognition and synthesis, R. de Mori, C.Y. Suen editors, Springer verlag, Heidelberg 1985.

- [Miclet 85]: Miclet L., Dabouz M., Un vocodeur à classification: transmission de la parole à très faible débit par quantification vectorielle du spectre, Séminaire GALF, Paris, pp 53-90, février 1985.

- [Mollier 84]: Mollier J., La signature numérique des messages chiffrés, pp 101-108, dans L'intelligence de l'informatique, Pour la science, Belin, 1984.

- [Moore 85]: Moore R., Systems for isolated and connected word recognition, Nato ASI Seies, Vol.F16,pp 73-143,
New systems and architectures for automatic speech recognition and synthesus, edited br R. De Mori and C.Y. Suen, Springer Verlag Berlin Heidelberg 1985.

- [Moore 83]: Moore R., Russel M.J., Tomlinson M.J., The discriminative network: a mechanism for focusing recognition in whole word pattern matching, Proc. IEEE Conf. Acoustics, Speech and Signal processing, 1983, pp 1041-1044.

- [Myers 80]: Myers C.S., Rabiner L.R., Rosenberg A.E., Performance tradeoffs in dynamic time warping algorithms for isolated word recognition, IEEE Trans. ASSP, vol 28,no 6, decembre 1980.

- [Myers 81]: Myers C.S., Rabiner L.R., A level building dynamic time warping algorithms for connected word recognition, IEEE Trans. ASSP. vol 29, no 2, avril 1981.

- [Nakagawa 83]: Nakagawa S., A connected spoken word recognition method by O(n) dynamic programming pattern matching algorithm, ICASSP, Boston, pp 296-299, avril 1983.

- [Ney 82]: Ney H., Connected utterance recognition using dynamic programming, in proc third congress FASE, DAGA, Goettingen, Germany, septembre 82, pp 915-918.

- [Ney 84]: Ney H., The use of a one-stage dynamic programming algorithm for connected word recognition, IEEE Trans. ASSP, vol 32, no. 12, avril 1984.

- [Rabiner 71]: Rabiner L.R., Scafer R.W., Design of digital filter banks for speech analysis, Bell Syst. Tech. J., vol. 50, no 10, pp 3097-3115, decembre 1971.

- [Rabiner 78]: Rabiner L.R., Rosenberg A.E., Considerations in dynamic time warping algorithms for discrete word recognition, IEEE Trans. ASSP, vol 26, no 6, decembre 1978.

- [Rabiner 81]: Rabiner L.R., Levinson S.E., Isolated and connected word recognition-theory and selected applications, IEEE Trans. Comm., vol 29, no 5, 1981, pp 621-659.

- [Rabiner 85]: Rabiner L.R., Soong F.K., Bergh A.F., Pan K.C., An efficient vector-quantization preprocessor for speaker independant

isolated word recognition, ICASSP85, vol-4, pp874-877, Tampa.

- [Saito 66]: Saito S., Itakura F., The theoretical consideration of statistically optimum methods for speech spectral density, report no 3107, Electrical Communication Laboratory, N.T.T., Tokyo, 1966.

- [Sakoe 78]: Sakoe H., Chiba S., Dynamic programming algorithms optimization for spoken word recognition, IEEE Trans. ASSP, vol26, no 1, 1978.

- [Sakoe 79]: Sakoe H., Two-level DP-matching. A dynamic programming based pattern matching algorithm for connected word recognition, IEEE Trans. ASSP, vol 27, no 6, 1979.

- [Sambur 76]: Sambur M.R., Rabiner L.R., Statistical decision approch to the recognition of connected digits, IEEE Trans. ASSP, vol 24, 1976.

- [Tassy 85]: Tassy A., Miclet L, Quantification vectorielle et reconnaissance de mots multilocuteur, pp 585-592, 5 ème congrès AFCET RFIA, Grenoble, novembre 1985.

- [Tassy 86]: Tassy A., Miclet L., Reconnaissance multilocuteur des chiffres français par association d'un prétraitement fondé sur la QV avec des modèles de Markov cachés, pp 251-254, 15 ème JEP, Aix en Provence, mai 1986.

- [Tou 74]: Tou J.T., Gonzalez R.C., Pattern recognition principles, Addison-Wesley, 1974.

- [Vintsyuk 68]: Vintsyuk T.K, Reconnaissance de mots par programmation dynamique, Kibernetica nO 1, pp 81-88, 1968.

- [Vintsyuk 71]: Vintsyuk T.K., Element-wise recognition of continuous speech composed of words from a specified dictionary, Kibernetica no 2, pp 133-143, mars-avril 1971.

ANNEXE 1

Les plages de fréquence du vocoder utilisé sur le Motorola sont:

- 200
- 247
- 306
- 378
- 468
- 579
- 716
- 886
- 1095
- 1355
- 1676
- 2073
- 2564
- 3171
- 3922
- 4850
- 6000

ANNEXE 2

Les erreurs rencontrées pour le locuteur masculin sont:

- une fois, 9 a été reconnu comme 8,
- huit fois, 0 a été reconnu comme 8,
- une fois, zero a été reconnu comme 7,
- une fois, 5 a été reconnu comme 1.

ANNEXE 3

LES mots de deux chiffres prononcés sont:

- 00, 01, 02, 03, 04, 05, 06, 07, 08, 09
- 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
- 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
- 30, 31, 32, 33, 34, 35, 46, 37, 38, 39
- 40, 41, 42, 43, 44, 45, 46, 47, 48, 49
- 50, 51, 52, 53, 54, 55, 56, 57, 58, 59
- 60, 61, 62, 63, 64, 65, 66, 67, 68, 69
- 70, 71, 72, 73, 74, 75, 76, 77, 78, 79
- 80, 81, 82, 83, 84, 85, 86, 87, 88, 89
- 90, 91, 92, 93, 94, 95, 96, 97, 98, 99

Les mots de trois chiffres prononcés sont:

- 000, 567, 142, 876, 292, 146, 735, 654, 989,633, 777, 201, 143, 666, 227, 506

Les mots de quatre chiffres prononcés sont:

- 0370, 7189, 7172, 5017, 1007, 0335, 2101, 8638, 1051, 2461, 7617, 1100, 0976, 0193, 3159, 8929, 8862, 2865, 1755, 8228, 1550, 1524, 3301, 0085, 8916, 9231, 2000, 1157, 2394.

ANNEXE 4

Nous avons réalisé des tests de reconnaissance de mots isolés sur le vocabulaire des dix chiffres, les formes comparées étant acquises à diverses fréquences d'échantillonnage. Deux locuteurs ont répété chacun huit séries des dix chiffres isolés, les résultats obtenus sont:

- avec une fréquence d'échantillonnage de 8kHZ, 12 erreurs de reconnaissance ont été rencontrées,
- à 10 kHZ, nous avons eu 5 erreurs,
- à 12 kHz, 5 erreurs ont été constatées,
- à 16 kHZ, un seule erreur a été détectée,
- à 20 kHz, une erreur a été décelée.

A l'issue de ces tests, nous avons décidé de choisir 16 kHz comme fréquence d'échantillonnage.

NOM DE L'ETUDIANT : *Mademoiselle BOYER Anne*

NATURE DE LA THESE : *Doctorat de l'Université de NANCY I en Informatique*

VU, APPROUVE ET PERMIS D'IMPRIMER

NANCY, le 26 MARS 1987

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

R. MAINARD

RESUME

Nous nous intéressons à la reconnaissance des mots isolés et des mots enchainés par la technique de programmation dynamique.

En reconnaissance de mots isolés, nous avons étudié un algorithme de programmation dynamique qui, tout en conservant le caractère optimal de la recherche, permet de relâcher les contraintes aux frontières assujettissant les chemins de recalage temporels afin de compenser certaines distorsions dues par exemple à une mauvaise détection parole non parole.

Nous introduisons la quantification vectorielle afin de réduire la place mémoire occupée par les données nécessaires à la reconnaissance (une ou plusieurs références par mot du vocabulaire). Nous proposons diverses méthodes de reconnaissance multilocuteur des mots isolés utilisant la programmation dynamique et nous comparons leurs performances (taux de bonne reconnaisssance, temps de calcul...).

Nous étudions ensuite l'algorithme de reconnaissance de mots enchainés par programmation dynamique proposé par Bridle et Nakagawa et nous montrons comment il est possible d'intoduire le relâchement des contraintes aux frontières ainsi que des contraintes syntaxiques.

Nous proposons un algorithme de programmation dynamique qui tient compte des effets de coarticulation ou de liaison qui apparaissent à la frontière de mots prononcés de façon continue.

Mots-clefs: mots isolés, mots enchainés, programmation dynamique, quantification vectorielle, relâchement des contraintes aux frontières, contraintes syntaxiques, coarticulation.