Formalisation grammaticale de la connaissance *a priori* pour l'analyse de documents : application aux partitions d'orchestre*

Bertrand Coüasnon
IRISA / INSA-Département Informatique
20, Avenue des buttes de Coësmes
35043 Rennes Cedex, France
Tél : +33 99 84 74 11
couasnon@irisa.fr

Résumé - La reconnaissance optique de partitions musicales est une forme d'analyse de document pour laquelle la connaissance a priori est particulièrement importante. Bien que la notation musicale soit régie par un ensemble de règles, les systèmes existants n'arrivent pas à utiliser correctement cette connaissance. Dans le cas de partitions complexes, ces systèmes ne peuvent régler les problèmes classiques de segmentation de la reconnaissance de document.

Nous proposons une nouvelle méthode de reconnaissance, utilisant une grammaire pour formaliser les règles syntaxiques et ainsi représenter le contexte. Cependant, lorsque des objets se touchent, il existe une différence entre la manière dont la connaissance (la grammaire) décrit un objet, et la manière de le reconnaître, puisque les objets se touchant doivent être en premier lieu segmentés correctement. Après avoir présenté la grammaire, nous introduirons deux opérateurs permettant de modifier la façon dont l'analyse de l'image s'effectue, afin de pouvoir, par exemple, segmenter correctement des objets se touchant (une altération touchant une tête de note).

Mots clés - Reconnaissance syntaxique, Segmentation, Grammaire, Reconnaissance optique de partitions musicales

1 Introduction

La connaissance a priori est très importante pour améliorer la qualité de la reconnaissance des documents à forte syntaxe. Cependant, dans les systèmes existants, la connaissance n'est généralement pas suffisamment utilisée et lorsqu'elle est utilisée, elle est introduite à de nombreux points du programme. Il en résulte un manque d'unité de la connaissance qui crée de réelles difficultés pour augmenter les capacités du système et pour modifier la connaissance utilisée. Ce manque d'unité et d'utilisation de la connaissance peut être expliqué par la différence qui existe entre la manière dont la connaissance décrit un objet et la manière dont cet objet doit être reconnu. Cette différence est généralement liée à des problèmes de segmentation.

En reconnaissance de partitions musicales, les problèmes de segmentation débutent avec les lignes de portée qui connectent tous les objets musicaux. Ensuite, les partitions complexes contiennent généralement un grand nombre de notes et sont souvent polyphoniques (différentes voix sur une même portée), ce qui génère une forte densité de l'information, débouchant sur d'autres problèmes de segmentation (par exemple des objets connectés alors que syntaxiquement ils ne devraient pas l'être). Or ces difficultés ne sont pas résolues par les systèmes existants [2], principalement parce qu'ils n'utilisent pas suffisamment les avantages syntaxiques de la notation musicale (*i.e.* la forte syntaxe et la connaissance a priori sur l'écriture musicale).

Nous présentons dans cet article, une méthode où la connaissance a priori sur la notation musicale contrôle l'ensemble du processus de reconnaissance, en particulier la segmentation et l'étiquetage des objets. Cette méthode permet d'améliorer et de fiabiliser le processus de reconnaissance. Fujinaga indiquant [8] que la notation musicale peut être formalisée par une grammaire, nous proposons d'utiliser une grammaire afin de contrôler l'ensemble de la reconnaissance. Le contexte introduit par cette formalisation, nous permet de tenter une segmentation adaptée à l'objet en cours de reconnaissance.

Dans les systèmes ascendants, les primitives sont généralement étiquetées dès qu'elles sont extraites (à l'aide d'informations très locales) et ces étiquettes ne sont jamais remises en question. Il est, dans ce cas, fort probable que des erreurs de reconnaissance se produisent sans qu'il soit possible de les corriger. La méthode que nous proposons permet au système de n'étiqueter définitivement une primitive que lorsqu'une règle s'applique, c'est-à-dire que l'étiquetage est réalisé à l'aide de la plus large information contextuelle.

Cependant, cette formalisation grammaticale n'est qu'une description d'une partition. Comme nous l'avons déjà indiqué, cette description ne permet pas, dans tous les cas, de reconnaître des objets pour lesquels il existe des difficultés de segmentation (par exemple une altération touchant une barre de groupe). Afin de pouvoir segmenter des objets se touchant et en même temps de conserver l'unité de la connaissance, nous proposons une méthode pour transformer automatiquement la descrip-

*. Dans les actes du dixième congrès Reconnaissance des Formes et Intelligence Artificielle, Rennes, France, janvier 1996.

tion grammaticale d'un objet, en une manière de le reconnaître.

Cette méthode peut être appliquée à d'autres domaines où la syntaxe est importante comme la reconnaissance d'expressions mathématiques, de dessins techniques, etc. La grammaire sur la notation musicale que nous proposons est suffisamment générale pour permettre de reconnaître des partitions d'orchestre. Cette grammaire a été validée par un musicien professionnel.

Nous débuterons cet article par l'étude des méthodes existantes, puis nous présenterons la grammaire (modélisant des partitions d'orchestre) qui constitue le coeur de notre système. Nous verrons ensuite comment en appliquant deux opérateurs sur la grammaire, il est possible de modifier en cours d'analyse la structure analysée et l'ordre d'analyse, afin de résoudre certains problèmes de segmentation liés aux objets connectés.

2 Méthodes existantes

La reconnaissance de partitions nécessite une phase de détection des lignes de portée sinon les objets logiquement séparés apparaîtraient comme un seul objet. Cette phase de détection permet, de réaliser une première segmentation des objets après suppression des lignes (technique utilisée par la plupart des auteurs), et surtout de pouvoir calculer par la suite la hauteur des notes.

Cette détection est réalisée à l'aide de différents traitements mais le plus souvent en considérant la ligne de portée comme une droite, ce qui permet d'utiliser des projections horizontales [9, 10] ou des recherches de cordes les plus longues [11, 14]. Seul Carter [4] propose une méthode à base du graphe des lignes adjacentes permettant de détecter des lignes de portée non-parfaitement rectilignes (cas assez fréquent).

Les objets, segmentés à la suite de la suppression des lignes de portée, sont ensuite classés selon différentes méthodes (certains auteurs en utilisent plusieurs) : préclassification sur la taille des boites englobantes[13, 5] ; projections horizontales et verticales [9] ; extraction de primitives (avec des techniques d'érosion, de squelettisation, etc.) comme tête de note, hampe, etc. Il n'y a généralement pas de remise en cause de cet étiquetage [1] de primitives sauf pour Kato [10].

D'après Blostein [2] et Carter [4], les principaux problèmes restant à résoudre sont les suivants :

- la majorité des systèmes développés sont des prototypes adaptés à des partitions assez simples. Le passage à des partitions complexes pose des difficultés (les mêmes techniques ne pouvant plus être utilisées). Par exemple la difficulté de passer d'un système monophonique à un polyphonique ;
- reconstituer un objet segmenté par le bruit ;
- le problème des objets connectés alors que, syntaxiquement, ils ne devraient pas l'être.

Le manque d'utilisation de la connaissance a priori est l'une des raisons pour lesquelles ces problèmes sont encore ouverts, bien que les différents systèmes de reconnaissance de partitions intègrent tous des règles de l'écriture musicale. Mais le plus souvent, ces règles sont réparties de manière ponctuelle au sein du processus de reconnaissance et manquent généralement de formalisation précise. Certaines solutions utilisent des règles formalisées par une grammaire mais elles sont alors principalement utilisées en vérification ou en correction d'erreurs et, éventuellement, en calcul des hauteurs finales des notes (variant avec les clés, altérations, etc.). Cependant, elles n'intègrent pas les règles existant au niveau graphique et restent limitées à des partitions très simples.

Seule la solution d'Andronico [1] tend vers ce type de formalisation jusqu'au niveau graphique en utilisant une grammaire « naturelle » et un ensemble de grammaires « bidimensionnelles », mais ces grammaires n'étaient adaptées qu'à des partitions simples et monophoniques. Fujinaga [8] définit la grammaire de la notation musicale comme étant algébrique et LL(k), ce qui permet au musicien (équivalant à un analyseur descendant) de lire cette notation de façon efficace. On peut donc s'orienter vers la définition d'une grammaire formalisant les règles de l'écriture musicale (que l'on peut rencontrer sur les partitions complexes), contrôlant tout le processus de reconnaissance afin d'y utiliser au maximum le contexte. Afin d'éviter de futurs problèmes d'extension de prototype, nous allons fixer les objectifs et, en fonction de ceux-ci, déterminer le système de reconnaissance.

3 Objectifs

Nous voulons développer un système capable de :

- reconnaître les partitions d'orchestre ;
- reconnaître les partitions complexes (polyphonie) ;
- avoir le moins possible d'erreurs de reconnaissance, et quand il y en a, les détecter afin d'éviter de vérifier «manuellement» toute la partition ;
- être adaptable aux partitions manuscrites ;
- augmenter son vocabulaire facilement ;
- reconnaître uniquement la notation «classique», la notation en musique contemporaine étant trop complexe.

Afin de définir une grammaire permettant de représenter une partition, nous allons dans un premier temps étudier le type d'informations que l'on peut y rencontrer.

4 Informations rencontrées sur une partition

Nous pouvons, comme l'a montré Martin [11], séparer en deux classes les objets se trouvant sur une partition :

- les *construits* : nommés « iconiques » par Martin (notes, groupes de notes, fractions de soupir, soufflets, etc.), constitués de segments (hampe, barre de groupe, etc.) et éventuellement de têtes (de notes, de fraction de soupirs) ou de crochets. De plus, un ensemble de règles d'assemblage de ces segments

1. Associer un nom à un objet graphique

et têtes permet de décrire les nombreuses formes de ces *construits*. Les barres de mesure, par leurs longueurs diverses, entrent également dans cette classe ;

– les *symboliques* : par exemple les clés, altérations, pauses, etc. peuvent être considérés comme des caractères. Les techniques utilisées en reconnaissance de caractères pourront donc leur être appliquées.

Comme les objets construits possèdent des formes diverses mais suivent des règles précises, il est nécessaire pour les reconnaître que la grammaire définisse ces règles. La grammaire devra également pouvoir manipuler la seconde classe d'objets : les *symboliques*.

Sur une partition, on peut constater une indépendance entre la manière de regrouper les notes et la sémantique musicale (si l'on ne s'intéresse qu'à la reconnaissance et non à l'interprétation). En effet, on peut considérer un groupe de notes comme étant syntaxiquement complètement équivalent à une suite de croches simples. Or l'information de groupe est importante pour faciliter la reconnaissance, mais rend plus difficile la gestion de la simultanéité des notes dans une partition d'orchestre. Il nous faut donc distinguer deux types d'informations que l'on peut rencontrer sur une partition :

– les informations physiques : correspondant à la manière dont les notes et leurs attributs (altérations, signes d'attaques, etc.) sont formés et agencés sur la partition ;

– les informations logiques : correspondant à la manipulation des notes de manière syntaxique dans l'écriture musicale. Ces notes sont alors indépendantes des regroupements, et possèdent en attribut toutes les informations les concernant : hauteur, durée, altération, etc. En effet, à ce niveau syntaxique, il n'est pas nécessaire de savoir, par exemple, où doit se trouver un dièse par rapport à la note, mais simplement que cette note est altérée par un dièse.

  Cette concentration de l'information musicale autour de la note permet entre autre au musicien de vérifier la synchronisation des notes entre les différentes portées d'une partition d'orchestre.

Cette séparation est également nécessaire pour reconnaître par exemple une clé au milieu d'un groupe de notes. En effet, cette clé ne peut être reconnue en même temps que le groupe car les modifications dues à la clé s'appliquent à toute la portée et non pas seulement au groupe.

La grammaire devra donc modéliser cette séparation de l'information musicale en deux niveaux.

5 Terminaux

La grammaire doit permettre de décrire un document et de définir un contexte utilisable dans le processus de reconnaissance. Pour ce faire, les éléments du vocabulaire terminal doivent suivre deux contraintes : d'une part, être les entités de base de la description du document et d'autre part, être reconnaissables sans information contextuelle, afin d'avoir une information sur laquelle le processus de reconnaissance puisse s'appuyer.

D'après les objets rencontrés sur une partition (les *symboliques* et les *construits*) on peut considérer comme vocabulaire terminal:

$V_T = \{$ segment $\} \cup$ ensemble des *symboliques*

En effet, les *construits* peuvent être modélisés à l'aide du segment et de têtes ou crochets (des symboles). Les *symboliques* eux, sont naturellement représentés par des symboles.

De plus, l'information segment est suffisamment simple et locale pour être reconnue sans contexte. Les symboles, qui peuvent être vus comme des caractères, sont reconnaissables sans contexte en utilisant les techniques d'OCR (*Optical Character Recognition*), à condition qu'ils soient correctement segmentés. Cette condition est réalisée suffisamment souvent après la suppression des lignes de portées pour que le système puisse, dans le cas d'erreurs de segmentation, les corriger à l'aide du contexte.

6 Description de la grammaire

6.1 Structure

La structure de la grammaire dépend bien sûr de ce que l'on souhaite reconnaître. D'après les objectifs que nous nous sommes fixés, nous devons, entre autres, reconnaître des portées polyphoniques et réaliser de la détection d'erreurs. Celle-ci nécessite de contrôler par exemple le nombre de temps dans une mesure. Or cette vérification n'est possible que voix par voix. Il faut donc que le système soit capable de répartir correctement les notes dans les voix. En conséquence, la grammaire devra intégrer cette notion de voix. On aura donc jusqu'à 3 voix par portée. Cependant la portée double peut comporter, par exemple dans une partition pour piano, une voix passant de la portée supérieure à la portée inférieure. Il est donc nécessaire de ne pas considérer ces doubles portées comme deux portées séparées, mais bien comme une seule portée comportant 11 lignes et autorisant jusqu'à 5 voix simultanées. La grammaire doit donc faire la distinction entre les portées simples (les non-terminaux débutant par les lettres « sp ») et leurs équivalents pour les portées doubles (débutant par « db »).

Afin de simplifier le traitement, et la vérification du nombre de temps dans une mesure se faisant au niveau de la mesure, la portée sera naturellement décomposée en mesures dans la grammaire.

Nous avons vu précédemment les deux niveaux physiques et logiques que l'on trouve sur une partition. Ces deux niveaux doivent se retrouver dans la grammaire. En effet, par exemple, pour vérifier le nombre de temps dans une mesure, il n'est pas nécessaire de connaître la forme d'une note, ni où sont placés ses attributs. Mais seulement qu'une note existe et possède des attributs. Nous allons donc séparer la reconnaissance en deux phases :

– une graphique (correspondant au niveau physique) qui reconnaît les notes en décrivant leur forme et le positionnement relatif de leurs attributs (altérations, points, signes d'attaque) ;

– une syntaxique (correspondant au niveau logique) qui reconnaît tous les symboles se rattachant à une voix comme les liaisons, nuances, etc., ou à une portée comme les clés, alta, bassa, nuances, liaisons, etc. C'est également à ce niveau que les notes sont réparties voix par voix et que sont contrôlés le nombre de temps par voix par mesure et l'alignement vertical des notes au sein d'une portée polyphonique ou d'un système d'orchestre.

6.2 Syntaxe, opérateurs

Avant de pouvoir écrire la grammaire, il faut définir une syntaxe et des opérateurs adaptés aux particularités du domaine. En effet, la grammaire devant décrire une image, c'est à dire un espace à 2 dimensions, il est nécessaire d'introduire des opérateurs particuliers permettant de définir le positionnement relatif des éléments. Ces opérateurs permettent en outre de s'affranchir des problèmes de non connexité en manuscrit, où par exemple les hampes le plus souvent ne touchent pas les têtes de notes. Ces opérateurs de position possèdent la sémantique suivante :

A P B

signifie A, et à la position P par rapport à A, on trouve B.

Cet opérateur permet de définir avec beaucoup de souplesse un lien entre 2 objets en spécifiant leurs positions relatives jusqu'à éventuellement, la zone de l'objet A en relation avec l'objet B. Cette souplesse est offerte par la diversité des opérateurs et on trouvera dans la grammaire, pratiquement un opérateur différent par lien entre objets.

Nous allons utiliser pour la grammaire la syntaxe suivante :

::=	le constructeur de règle
en italique	les terminaux
en gras sans sérif	les opérateurs de position
en sans sérif	les non-terminaux

Nous ajoutons à cette syntaxe les abréviations classiques des expressions régulières :

A^*	répéter A de 0 à n fois
A^+	répéter A au moins une fois
$A^?$	répéter A 0 ou 1 fois
A^{n-p}	répéter A de n à p fois
|	ou

L'utilisation des opérateurs de position nécessite de faire référence à la variable grammaticale à gauche de l'opérateur pour positionner la variable grammaticale de droite. Mais, il peut être nécessaire qu'une même variable grammaticale serve de référence à plusieurs opérateurs de position afin de placer d'autres variables grammaticales. Ceci est possible grâce à l'utilisation d'une notation de factorisation :

A(P1 B:P2 C)

signifie A P1 B et A P2 C

Parmi les objectifs, nous nous sommes fixés la reconnaissance de partitions d'orchestre. Afin de réaliser cet objectif le plus facilement possible, nous définissons un opérateur Syst permettant d'appliquer une règle adaptée à une portée simple, à l'ensemble des portées d'un système (c'est à dire à une liste de portées), générant ainsi de manière automatique la règle pour un système de portées. Cet opérateur Syst s'occupe bien évidemment de gérer de manière transparente la circulation des attributs qui par exemple doivent circuler « horizontalement » sur une même portée simple. La syntaxe de Syst est la suivante :

Syst(R)

pour définir l'extension d'une règle R pour une portée simple, à la règle adaptée au système complet.

Cet opérateur permet une extension simple de la grammaire pour portée simple vers une grammaire adaptée aux partitions d'orchestre.

Bien que la grammaire offre une approche descendante de l'écriture musicale, nous allons, pour une meilleure compréhension, réaliser la présentation de la grammaire de manière ascendante. Nous verrons ainsi en premier la partie graphique, puis syntaxique et enfin la vue globale de la grammaire. Nous ne pourrons présenter ici la grammaire complète car elle représente 5 pages de règles.

6.3 Présentation de la partie graphique

Afin d'illustrer le niveau graphique (voir figure 2) de la grammaire nous allons présenter (voir figure 1) un ensemble de règles très simplifiées définissant un groupe de croches. Ainsi, il n'y a qu'une seule barre de groupe reliant les notes. Les accords, les points d'allongement de durée d'une note, les signes d'attaque et les liaisons rythmiques ne sont pas définis dans cet exemple.

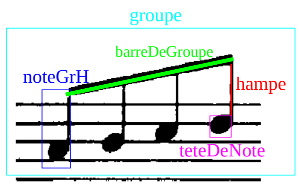

Fig. 1 – *Représentation graphique de la règle* groupe

Les non-terminaux suivants représentent :

fracSoupir	demi-soupir, quart de soupir, etc.
noteGr	note graphique
noteGrH	note graphique à hampe orientée vers le haut
noteGrB	note graphique à hampe orientée vers le bas
têteDeNote	tête de note avec tous ses attributs
tête	pouvant être une tête de noire ou de blanche
attributNote	limité ici aux altérations (pas de points ou de signes d'attaque)
altération	pouvant être bécarre, bémol ou dièse

Les opérateurs de positionnement utilisés dans cet exemple ont les significations suivantes :

extrémitéGauche	à l'extrémité gauche
procheV	proche selon la verticale

extrémitéDroite	à l'extrémité droite
procheGExtBas	proche à gauche de l'extrémité basse
procheGmêmeLigne	proche à gauche sur à la même hauteur

Les règles sont :

groupe ::= barreDeGroupe
  (extrémitéGauche noteGr :
  [procheV [noteGr | fracSoupir]]* :
  extrémitéDroite noteGr)
noteGr ::= noteGrH | noteGrB
têteDeNote ::= tête attributNote
attributNote ::= (procheGmêmeLigne [altération]$^?$:
  ...)
hampe ::= *segment*
barreDeGroupe ::= *segment*

Les règles, barreDeGroupe et hampe provoquent un étiquetage des segments leur correspondant.

On peut donc ainsi traduire la règle noteGrH par :

> On a une noteGrH si : on a une hampe, et proche à gauche de l'extrémité basse de la hampe, on trouve une têteDeNote.

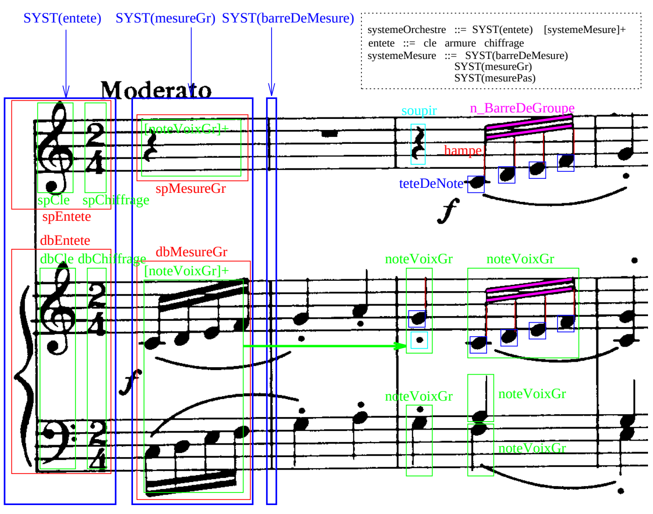

FIG. 2 – *Représentation graphique du niveau partition et de* mesureGr

L'intérêt de la notation de factorisation est illustrée par la règle groupe. Cette règle signifie qu'un groupe est constitué d'une barre de groupe et qu'à chaque extrémité de cette barre il y a une note graphique et qu'entre ces deux notes il peut y avoir d'autres notes graphiques. Ainsi, chaque opérateur de positionnement (extrémitéGauche, procheV et extrémitéDroite) fait référence au non-terminal barreDeGroupe. Ceci est utile lorsque plusieurs objets sont positionnés par rapport à un seul autre.

Voyons maintenant la partie graphique de la grammaire de manière plus globale à l'aide de la figure 2. Nous pouvons constater qu'à ce niveau, il n'y a pas de prise en compte de la polyphonie. Les notes sont simplement reconnues de manière indépendante et ce n'est que dans la partie syntaxique que ces notes seront réparties voix par voix.

Dans cette partie graphique, seuls les attributs des notes sont reconnus avec les notes, c'est-à-dire : les altérations, les signes d'attaques et les points d'allongement de durée. Les autres signes, comme les nuances, qui se rapportent à une voix ou à la partie entière seront reconnus par la suite dans la partie syntaxique.

6.4 Présentation de la partie syntaxique

Cette partie utilise les résultats de la partie graphique, c'est-à-dire l'ensemble des notes reconnues et de leurs attributs. Par rapport à ces notes, sont définis les positionnements des attributs de voix ou de portée comme les liaisons de phrasées, les clés en cours de mesure, (et donc éventuellement au milieu d'un groupe), les nuances, les alta et bassa, etc. Les notes sont en plus représentées voix par voix ce qui permet de réaliser la vérification de la cohérence entre le nombre de temps dans la mesure et le chiffrage.

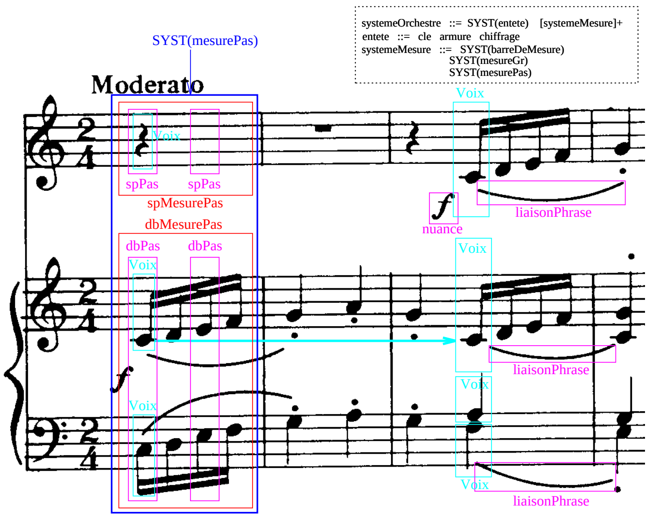

FIG. 3 – *Représentation graphique de* mesurePas

Afin de réaliser une vérification de l'alignement vertical des notes, aussi bien entre les portées d'une partition d'orchestre, mais aussi entre les voix d'une portée polyphonique, nous proposons la notion de Pas (figure 3). Un Pas correspond à un découpage temporel de la partition, ce qui graphiquement donne un découpage vertical de la partition. La durée d'un Pas est en fait la plus petite durée dans une colonne de notes verticalement alignées (c'est-à-dire débutant en même temps) sur l'ensemble de la partition. La méthode pour que le système détecte lui-même ses erreurs, afin d'améliorer la fiabilité, est présentée dans [7].

6.5 Présentation du niveau partition

Nous avons vu dans la description de la structure d'une partition (section 6.1) qu'il était nécessaire de considérer les portées de type piano, non pas comme deux portées indépendantes, mais comme une seule portée à 11 lignes. Ceci afin de permettre à une voix de passer d'une portée à l'autre. Les règles de grammaire s'attachant à une portée seront donc toujours décomposées en une règle pour les portées simples et une pour les portées dites doubles, comme dans cet exemple :

mesureGr ::= spMesureGr | dbMesureGr

Lorsqu'il y a notion de voix dans une règle, les portées simples sont limitées (d'après les musiciens) à 3 voix et les portées doubles à 5 voix. Nous pouvons également voir sur les figures 2 et 3 l'utilisation de l'opérateur SYST afin de passer automatiquement d'une règle pour une portée seule (simple ou double) à une règle pour l'ensemble des portées d'un système d'orchestre.

7 Description et reconnaissance

Nous venons de présenter une grammaire permettant de modéliser de manière générique une partition d'orchestre à portées polyphoniques ainsi que la plupart de ses attributs. Cette grammaire constitue donc une description de la connaissance de l'écriture musicale.

Cette description est de plus, séparée du programme ce qui permet une modification facile et une adaptation beaucoup plus souple de la connaissance utilisée. Cependant cette modélisation grammaticale de la connaissance ne fait que décrire un ensemble d'objets (la partition), alors que nous cherchons un moyen pour reconnaître cet ensemble d'objets. En effet, il y a une différence entre la manière dont la connaissance décrit un objet et la manière dont cet objet peut être reconnu. Cette différence est plus ou moins marquée selon la segmentation des objets.

Ainsi, s'il n'y a pas de problèmes de segmentation, c'est-à-dire si les objets sont séparés les uns des autres, il est possible d'appliquer de manière indépendante chaque règle représentant un objet *symbolique* ou *construit*. En effet, puisque un objet est reconnu à l'aide d'une règle, et que cet objet est isolé des autres, sa règle associée est indépendante des autres. Étant indépendante, l'ordre d'évaluation importe peu si ce n'est pour un positionnement relatif des objets. Il est donc possible d'utiliser l'ordre d'évaluation simplement défini par une application LL des règles de la grammaire. Nous avons dans ce cas idéal une adéquation entre modélisation de la connaissance et méthode de reconnaissance.

Cependant, ce cas ne correspond pas à la réalité. En effet, lorsque l'on s'intéresse aux partitions complexes, des problèmes de segmentation liés à la densité de l'information, se présentent. Ces problèmes peuvent se répartir selon deux classes :

- celle mettant en jeu uniquement des objets décrits par un seul terminal (en l'occurence les *symboliques*) ;
- celle mettant en jeu au moins un objet décrit par un non-terminal (en l'occurence les *construits*).

La première classe de problèmes correspond au cas où des *symboliques* touchent d'autres *symboliques* ou bien au cas où un *symbolique* est segmenté en plusieurs morceaux. Seuls les terminaux étant en cause, on ne peut résoudre ces problèmes de segmentation qu'en modifiant ce qui dans la grammaire est lié aux terminaux, c'est-à-dire la structure de donnée analysée. Cette modification n'est possible que grâce à l'utilisation du contexte modélisé par la grammaire.

La seconde classe de problèmes correspond au cas où un *construit* touche un autre *construit* ou un *symbolique*. Des non-terminaux (représentés par des règles) étant alors en cause, il faut pour résoudre les problèmes de segmentation, modifier dans la grammaire ce qui est lié aux non-terminaux, c'est-à-dire les règles grammaticales. Le seul moyen d'intervenir sur une règle sans modifier sa définition (afin de conserver l'unité de la connaissance) est de changer son ordre d'évaluation.

7.1 Modification de la structure analysée

Cette modification est nécessaire pour régler les problèmes de segmentation concernant uniquement les *symboliques* (les objets décrits par un seul terminal) :

- connexion *symbolique-symbolique* ;
- *symbolique* segmenté en différents morceaux.

Cependant, il est très difficile, par exemple, de séparer 2 altérations se touchant (figure 4) sans connaître le type des objets que l'on cherche à séparer. En effet, on tombe dans ce cas sur le paradoxe que pour reconnaître, il faut segmenter correctement, mais pour segmenter correctement, il faudrait avoir déjà reconnu. Un moyen pour contourner ce paradoxe est d'utiliser le contexte (les objets déjà reconnus), afin d'en déduire le type d'objet que l'on peut trouver dans une composante connexe non reconnue. Ainsi, si le système cherche à trouver une altération devant une tête de note, si de plus, à cet endroit se trouve une matrice de pixels non reconnue, il est alors possible de tenter une segmentation de la composante en sachant que l'on recherche une altération. Grâce à cette information, on connaît les caractéristiques géométriques de l'objet recherché, ce qui permet d'effectuer une segmentation adaptée (voir exemple en section 7.1.1). Cette segmentation est ensuite validée en présentant les nouvelles matrices de pixels au reconnaisseur. Si celui-ci reconnaît un *symbolique*, ces matrices remplacent la matrice de pixels initiale dans la structure de donnée, puis l'analyse se poursuit. Si le reconnaisseur rejette ces matrices, la segmentation est refusée, la structure ne change pas et une autre règle pourra par la suite tenter une autre segmentation.

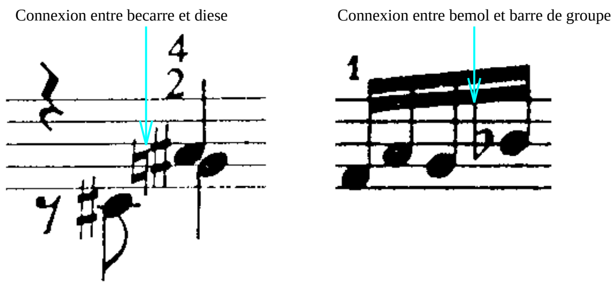

FIG. 4 – *Exemples d'objets connectés à tort*

De la même manière, pour les *symboliques* sur-segmentés, il est possible d'effectuer des essais de regroupement adaptés au type de *symbolique* recherché. En effet, les *symboliques* risquant d'être sur-segmentés le sont généralement de la même manière (par ex. la clé de fa), ce qui permet de savoir où retrouver les morceaux manquants du *symbolique*. Selon la même méthode que la séparation des *symboliques* se touchant, le reconnaisseur vérifie ensuite le regroupement ; s'il est valide, le *symboliques* est ajouté dans la structure à la place de ses composantes, sinon la structure reste la même et un autre regroupement ou une autre séparation seront essayés.

Ces essais de séparation et regroupement sont d'autant plus implémentables que le nombre de *symboliques* généralement touchés par ces problèmes de segmentation est relativement faible.

Nous proposons donc un opérateur supplémentaire pour la grammaire :

RECHERCHE(C)

où C est une classe de *symbolique* (altération par exemple) ou bien un *symbolique* particulier (clé de fa par exemple).

Cet opérateur recherche dans la structure analysée, à la position définie par les opérateurs de position, un *symbolique* ou une classe de *symbolique* en effectuant le traitement adapté à l'objet, c'est-à-dire soit une séparation, soit un regroupement, soit une combinaison des deux.

7.1.1 Opérateur RECHERCHE

Afin de mieux cerner le principe de fonctionnement ce cet opérateur RECHERCHE, nous allons voir un exemple de segmentation à l'aide du contexte. La règle :

têteDeNote ::= tête attributNote
attributNote ::= (procheGmêmeLigne [altération]$^?$:
...)

s'écrit avec l'opérateur RECHERCHE (connaissant la propension qu'ont les altérations à être connectés à tort):

têteDeNote ::= tête attributNote
attributNote ::= (procheGmêmeLigne
RECHERCHE(Altération)
[altération]$^?$:
...)

Dans le cas du bécarre touchant le dièse (figure 4), au moment de l'appel à l'opérateur RECHERCHE, la composante connexe contenant les deux *symboliques* n'est pas étiquetée. En revanche, elle se trouve correctement placée par rapport aux contraintes données par l'utilisateur de position procheGmêmeligne. L'opérateur RECHERCHE peut segmenter cette composante grâce à l'information contextuelle représentée par l'argument de RECHERCHE.

En effet, en utilisant les caractéristiques des altérations, la segmentation devient beaucoup plus simple. Ainsi, la composante connexe est séparée en deux, par exemple sur un critère de largeur moyenne d'une altération. Chacune des deux nouvelles composantes est ensuite présentée au classifieur afin de vérifier si elle représente un *symbolique*. Dans l'exemple, le bécarre et le dièse sont reconnus car ils sont maintenant correctement segmentés. A la sortie de l'opérateur RECHERCHE, les deux composantes étiquetées remplacent la composante initiale non étiquetée, dans la structure analysée. L'analyse se poursuit ensuite exactement de la même manière que si les deux altérations avaient été segmentées correctement depuis le début. Chaque altération étant attribuée à la bonne tête de note.

7.1.2 Fonctionnement général

D'une manière plus générale et plus précise, nous pouvons définir l'opérateur RECHERCHE par :
RECHERCHE(C) :

- appelé au sein d'une règle ;
- en fonction de la zone de recherche (modifiée par les opérateurs de position), sélectionne la ou les matrices de pixels non étiquetées ;
- applique RECHERCHESURMATRICE(C) à chaque matrice sélectionnée.

où RECHERCHESURMATRICE(C) :

- réalise une recherche adaptée à la classe d'objet C ;
- **si** Boite englobante de la matrice < Boite englobante de C **alors** REGROUPEMENT(C)
sinon SEPARATION(C) ;
- **si** au moins un *symbolique* est trouvé **alors** modification de la structure analysée.

L'opérateur RECHERCHE s'appliquant dans les cas où l'extraction des *symboliques* fondée sur la connexité n'est pas valide, il est nécessaire que les opérations de regroupement et de séparation ne soient pas liées à cette connexité. Le principe de ces deux opérateurs est de considérer la matrice de pixels courante comme étant la zone de référence, la recherche s'effectuant dans l'image, indépendamment des composantes connexes :

REGROUPEMENT **(et** SEPARATION **éventuelle)**
: Pour chaque classe (ou objet) C, on peut considérer qu'il existe au moins une composante connexe qui possède une forme stable quels que soient les problèmes de sur-segmentation. Nous appellerons ces composantes connexes des « zones d'appui » à partir desquelles il est possible de positionner une boite englobante de taille correspondant à celle de C.

La matrice de pixels (indépendante de la connexité) au sein de cette boite englobante est présentée au classifieur qui détermine si il y a ou non un *symbolique* connu.

SEPARATION **(et** REGROUPEMENT **éventuel)**
: Selon le même principe que le regroupement, sa séparation est effectuée en fonction de la classe (ou l'objet) C, en positionnant la boite englobante par rapport à des zones d'appui. Celles-ci peuvent être, par exemple pour les altérations, des segments verticaux détectés à l'aide de l'extracteur de segment, une position par rapport à la portée, etc. La matrice de pixels segmentée est ensuite présentée au classifieur.

La modification de la structure analysée n'a lieu que lorsque au moins un *symbolique* a été détecté par le regroupement ou la séparation. C'est en fait le contexte (modélisé par la grammaire) qui permet de contourner le paradoxe qu'il faut segmenter pour reconnaître, mais que pour segmenter il faudrait avoir reconnu. En effet, le contexte permet de considérer que l'on a reconnu une classe C (ou un *symbolique*), lorsqu'un *symbolique* est reconnu, on peut alors le segmenter correctement à l'aide d'une segmentation adaptée. Cette segmentation est réalisée en appliquant un masque adapté au *symbolique* reconnu dans l'image des pixels restant à reconnaître.

La structure analysée devant être isomorphe avec l'image, toutes les matrices de pixels (non reconnues) connexes à la boite englobante du *symbolique* sont remplacées par le *symbolique* reconnu et les composantes connexes calculées après suppression du *symbolique* reconnu de l'image.

7.2 Modification de l'ordre d'évaluation

Cette modification est nécessaire lorsqu'un *construit* touche un autre *construit* ou un *symbolique*. Ces deux types de connexions provoquent les mêmes difficultés et se résolvent de la même manière. Nous allons donc étudier le cas où un *symbolique* touche un *construit*.

Un *symbolique* peut toucher un *construit* au niveau segment (une hampe ou une barre de groupe) ou au niveau tête (tête de note ou tête de fraction de soupir) (figure 5).

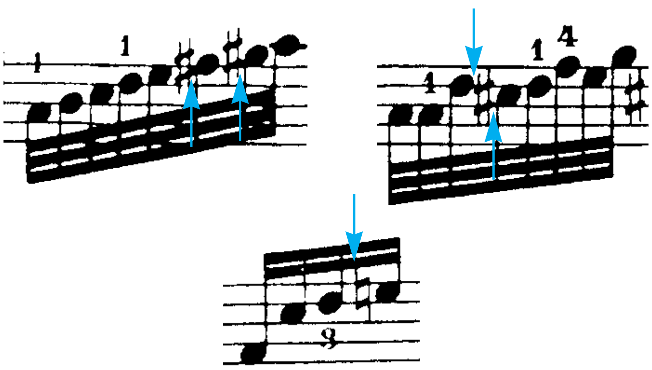

Fig. 5 – *Exemples de symboles touchant des construits*

Pour reconnaître ces *symboliques*, le système doit dans un premier temps les segmenter correctement. Une manière de réaliser cette segmentation est de reconnaître en premier les segments et les têtes, puis de les supprimer de l'image. De cette façon, les *symboliques* sont correctement (« naturellement ») séparés.

Cette segmentation est tout à fait possible car le détecteur de segments (présenté dans [12]) peut extraire les segments même s'ils sont connectés à d'autres objets. En effet, l'information segment est suffisamment locale pour être reconnue au bas niveau même si les segments ne sont pas correctement segmentés. Cependant, la description d'un *construit* (par exemple la règle grammaticale groupeDeNote) nécessite de reconnaître les *symboliques* (par exemple les altérations) au même moment que les segments. En effet, pour pouvoir reconnaître la barre de groupe d'un groupeDeNote, il est nécessaire de reconnaître en premier lieu toutes les noteGr, or pour reconnaître chaque noteGr il faut reconnaître en même temps : la hampe (un segment), la tête et l'éventuelle altération (un *symbolique*). Lorsque l'altération touche la barre de groupe, la séparation « naturelle » (en supprimant les segments reconnus) ne peut être appliquée car la barre de groupe n'est pas encore reconnue, puisqu'elle nécessite la reconnaissance de toutes les noteGr. On rencontre ici une sorte de cercle vicieux entre la description d'un objet et la manière dont il peut être reconnu.

Une solution pourrait être de séparer la description hampe-tête des *construits* de celle des attributs de note. Le processus de reconnaissance consisterait alors à appliquer les règles des *construits* (pour reconnaître les segments et les têtes) et ensuite les règles sur les attributs. Une difficulté serait alors de redistribuer les bons attributs aux bonnes notes. Cependant, la plus importante serait que de cette manière, on perd l'unité de la connaissance, et cette connaissance a priori ne peut alors être séparée du programme. Or cette séparation est très importante si l'on souhaite un système pouvant augmenter ses capacités grâce à la possibilité d'adapter, modifier les règles facilement. Cette difficulté de séparer la connaissance et le programme provient du cercle vicieux existant entre la description d'un objet et le moyen de le reconnaître. C'est sans doute l'une des raisons pour lesquelles les autres systèmes utilisent seulement un peu de connaissance et qui est directement insérée dans le programme (donc non séparée de celui-ci).

Nous proposons afin de résoudre ce problème, d'introduire un opérateur Gel. Celui-ci pourra être capable de changer radicalement la manière dont une règle est évaluée, en ne changeant pratiquement pas l'énoncé de la règle décrivant un objet. Ainsi, par exemple, l'énoncé de le règle groupeDeNote restera pratiquement le même, mais l'opérateur Gel permettra de façon transparente, de réaliser la segmentation (en supprimant les segments reconnus) en modifiant automatiquement l'ordre d'évaluation.

7.2.1 Opérateur Gel

Afin de conserver l'unité de la grammaire, nous introduisons un opérateur Gel qui modifie de façon transparente l'ordre d'évaluation des règles.

Ainsi, la règle noteGrH (simplifiée car sans les accords pour une meilleur compréhension) :

noteGrH ::= hampe procheGExtBas teteDeNote
têteDeNote ::= tête attributNote

s'écrit en fait avec l'opérateur Gel :

noteGrH ::= hampe procheGExtBas teteDeNote
têteDeNote ::= tête Gel(attributNote)

L'opérateur Gel va systématiquement stopper l'évaluation de la règle où il se trouve (dans l'exemple, la règle têteDeNote) au niveau de la règle en argument (dans l'exemple, la règle attributNote), jusqu'à la fin de l'analyse de la partie graphique de la mesure (mesureGr). Le système doit effectivement attendre la fin de la mesure, car un *symbolique* peut toucher un segment appartenant à un *construit* différent de celui auquel le *symbolique* se réfère. Afin d'éviter une explosion combinatoire, les conditions minimales d'application de la règle en cours (ici têteDeNote) doivent être présentes avant l'utilisation du GEL. Pour une tête de note, ces conditions sont la présence d'une tête. C'est pour cette raison que la règle têteDeNote réussit simplement si tête détecte une tête (noire ou blanche) à la position définie par l'opérateur de position.

L'opérateur GEL ne fait qu'ajouter dans un argument de têteDeNote (une liste de toutes les règles déjà stoppées, qui circule dans toutes les règles de la grammaire), la règle en argument (attributNote), ses attributs et la position courante dans l'analyse. À la fin de la mesure, les têtes et les segments reconnus comme hampe ou barre de groupe, sont supprimés de l'image, réalisant ainsi une segmentation « naturelle » entre les *symboliques* et les *construits* se touchant. Les composantes connexes sont ensuite calculées, générant ainsi une nouvelle structure de mesure à analyser. Chaque règle stoppée (stockée dans la liste des règles) est appliquée à la nouvelle structure, dans laquelle les *symboliques* touchant les *construits* sont segmentés correctement. Grâce à cette segmentation correcte, les *symboliques* peuvent être reconnus à l'aide d'un classifieur utilisant des techniques classiques de reconnaissance de caractères (figure 6).

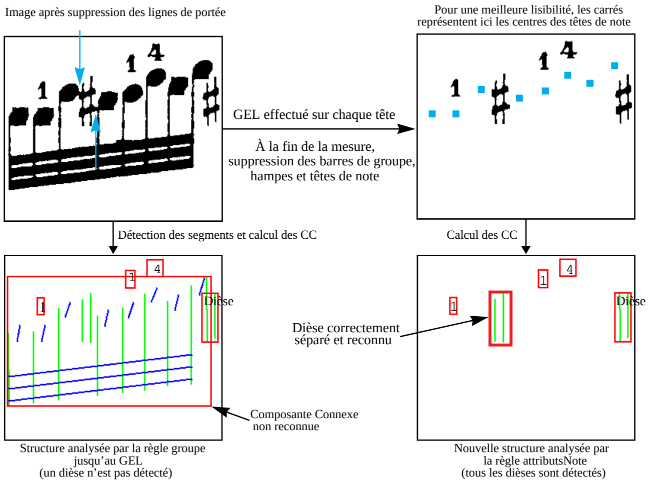

FIG. 6 – *Structure avant et après* GEL

La position courante de l'analyse ayant été également stockée lors du GEL, chaque règle est en mesure d'associer les bons attributs (altérations, etc.) à la bonne tête.

7.2.2 Fonctionnement général

D'une manière plus générale et plus précise, nous pouvons définir l'opérateur GEL par :
GEL(R) :

- appelé au sein d'une règle ;
- ajoute à la liste de toutes les règles déjà gelées : la règle R, ses arguments et la position courante dans l'analyse ;

EVALGEL est appelé à la fin de la mesure graphique (mesureGr) et effectue :

- le calcul de la nouvelle structure de donnée après avoir supprimé de l'image les segments et têtes reconnues ;
- l'application de chaque règle stoppée (stockées dans la liste des règles gelées) à cette nouvelle struture.

8 Résultats

La grammaire est actuellement entièrement définie pour des partitions d'orchestre, avec différentes voix sur une même portée, des accords dans une voix, des signes d'attaque, n-olets (triolet, etc.), pause, octava, nuances, liaisons rythmiques et de phrasé, et appogiatures. Les abréviations, ornements (sauf les appogiatures) et les paroles ne sont pas encore inclus dans la grammaire.

La plupart des outils nécessaires pour implémenter la grammaire ainsi que les opérateurs de position ont été écrits. Nous avons développé un compilateur de grammaire permettant, partant de la grammaire, de produire automatiquement l'analyseur associé. Grâce à cette séparation de la connaissance a priori et du programme, il est possible de tester le système sur un sous-ensemble de la grammaire sans avoir de problèmes lors du passage à la grammaire complète.

Nous avons donc développé le système en utilisant un sous-ensemble de grammaire permettant de reconnaître les partitions d'orchestre polyphoniques avec les clés, chiffrages, armures, blanches, noires, croches simples, groupes de notes, altérations, points d'allongement de durée, pauses et barres de mesure. Les opérateurs RECHERCHE et GEL sont implémentés. Cependant, seuls les traitements de SEPARATION associés aux classes Têtes et Crochet ont été réalisés. Les autres classes de SEPARATION et l'opérateur REGROUPEMENT n'ont pu être ajoutés au système car le classifieur de *symboliques* est actuellement trop simple (uniquement pour un problème de temps) pour permettre un fonctionnement avec SEPARATION et REGROUPEMENT. En revanche, grâce à l'opérateur GEL, ce système peut, en utilisant le contexte, résoudre les problèmes de segmentation créés lorsqu'un *symbolique* est connexe à un groupe de notes. Le système peut également détecter et éventuellement corriger, la plupart des erreurs temporelles grâce à l'alignement vertical et le nombre de temps dans une voix par mesure.

L'analyseur [6] est implémenté en λProlog [3], un dialecte typé de Prolog intégrant l'unification d'ordre supérieur. Le choix de ce langage nous a permis de produire relativement rapidement un prototype efficace au vu de la complexité du problème. L'ordre supérieur de λProlog nous a offert la possibilité d'implémenter l'opérateur GEL. Nous avons également écrit un compilateur de grammaire qui, à partir du formalisme grammatical génère automatiquement l'analyseur.

En exemple, pour analyser une image de 1900x900 pixels, comportant une partition d'orchestre à 3 portées possédant 78 éléments reconnus (sur 282 éléments analysés), sur un SUN Sparc 10, il faut 6s pour le traitement d'image (extraction des segments, détection et suppression des lignes de portée, construction de la structure à analyser) et 5s pour l'analyse au niveau grammatical.

9 Conclusion

Nous avons présenté un système de reconnaissance de partitions musicales entièrement contrôlé par une grammaire. Cette grammaire a été définie pour pouvoir reconnaître des partitions d'orchestre, avec différentes voix sur une seule portée, des accords, des signes d'attaque, des liaisons, des nuances, etc.

Contrairement aux autres systèmes, notre approche formalise toutes les règles utilisées pour la reconnaissance et produit une intégration maximum du contexte. Habituellement, la syntaxe est utilisée pour vérifier un étiquetage. Ici, c'est la syntaxe qui contrôle tout le processus de reconnaissance produisant ainsi un meilleur étiquetage. Généralement, les méthodes grammaticales ne sont utilisées qu'en vérification (à un haut niveau), alors que notre système retourne au niveau image pour produire une segmentation adaptée au contexte et ainsi obtenir une meilleur reconnaissance.

Un autre avantage de cette formalisation grammaticale, est la séparation entre la partie dynamique du système (le programme) et la définition des règles musicales (la grammaire). Cette séparation nous permet de modifier facilement une règle ou d'adapter le système à un autre type de document structuré. En effet, il suffit de définir une nouvelle grammaire pour construire un nouveau système de reconnaissance. Cette séparation est possible grâce à l'introduction des opérateurs RECHERCHE et GEL qui modifient, l'un la structure analysée et l'autre l'ordre d'évaluation, afin de résoudre des problèmes de segmentation : *symboliques* sur ou sous segmentés, *symboliques* touchant des *construits*. Ces opérateurs peuvent également être utilisés sur d'autres types de documents structurés où l'unité de la connaissance a priori est très importante. Cette unité peut être conservée grâce à ces opérateurs même lorsqu'il existe une différence entre la description d'un objet et la manière de le reconnaître.

De plus, cette méthode est une réponse possible aux problèmes non résolus décrits par Blostein [2] et Carter [4] : reconstruction d'objets séparés et segmentation d'objets se touchant. Elle permet également de résoudre les difficultés de passage d'un prototype à un système complet. Le système est actuellement capable de reconnaître des partitions d'orchestre polyphoniques. Grâce au nombre de temps dans une mesure et à l'alignement vertical des notes, l'analyseur peut détecter et corriger des erreurs sur la durée des notes, afin de produire des résultats plus fiables.

Références

[1] Andronico (A.) et Ciampa (A.). – On automatic pattern recognition and acquisition of printed music. *In : ICMC, International Computer Music Conference*, pp. 245–278. – Venice, Italy, 1982.

[2] Blostein (D.) et Baird (H.). – A critical survey of music image analysis. *In : Structured Document Image Analysis*, éd. par Springer-Verlag, pp. 405–434. – Eds. H.S. Baird, H. Bunke, K. Yamamoto, 1992.

[3] Brisset (P.) et Ridoux (O.). – *The Compilation of λProlog and its execution with MALI.* – Rapport technique n° 687, IRISA, 1992. ftp://ftp.irisa.fr/local/lande.

[4] Carter (N. P.) et Bacon (R. A.). – Automatic recognition of printed music. *In : Structured Document Image Analysis*, éd. par Springer-Verlag, pp. 456–465. – Eds. H.S. Baird, H. Bunke, K. Yamamoto, 1992.

[5] Carter (N.P.), Bacon (R.A.) et Messenger (T.). – The acquisition representation and reconstruction of printed music by computer : A review. *Computers and the Humanities*, vol. 22, 1988, pp. 117–136.

[6] Couasnon (B.), Brisset (P.) et Stephan (I.). – Using logic programming languages for optical music recognition. *In : International Conference on the Practical Application of Prolog*, pp. 115–134. – Paris, France, avril 1995.

[7] Couasnon (B.) et Retif (B.). – Using a grammar for a reliable full score recognition system. *In : International Computer Music Conference.* – Banff, Canada, septembre 1995.

[8] Fujinaga (I.). – *Optical Music Recognition using projections.* – Montreal, Canada, Thèse, McGill University, Faculty of Music, 1988.

[9] Fujinaga (Ichiro), Alphonce (Bo), Pennycook (Bruce) et Boisvert (Natalie). – Optical recognition of music notation by computer. *Computers in Music Research 1*, 1989, pp. 161–164.

[10] Kato (H.) et Inokuchi (S.). – A recognition system for printed piano music using musical knowledge and constraints. *In : Structured Document Image Analysis*, éd. par Springer-Verlag, pp. 435–455. – Eds. H.S. Baird, H. Bunke, K. Yamamoto, 1992.

[11] Martin (Philippe). – *Réseaux de neurones artificiels: Application a la reconnaissance optique de partitions musicales.* – Grenoble, France, Thèse de PhD, IMAG, 1992.

[12] Poulain d'Andecy (V.), Camillerapp (J.) et Leplumey (I.). – Kalman filtering for segment detection: Application to music scores analysis. *In : ICPR, 12th International Conference on Pattern Recognition (IAPR)*, pp. 301–305. – Jerusalem, Israel, octobre 1994.

[13] Prerau (D. S.). – Do-re-mi: A program that recognizes music notation. *Computer and the Humanites*, vol. 9, n° 1, janvier 1975, pp. 25–29.

[14] Roach (J. W.) et Tatem (J.E.). – Using domain knowledge in low-level visual processing to interpret handwritten music : an experiment. *Pattern Recognition*, vol. 21, n° 1, 1988, pp. 33–44.