Détection de motifs au sein de partitions musicales manuscrites

Assistance à la musicologie et l'analyse des compositions

Travaux de thèse de Riyadh Benammar

Direction:

Véronique Eglin, IMAGINE, LIRIS UMR 5205 Lyon

Christine Largeron, LHC UMR 5516 Saint-Etienne

Contexte

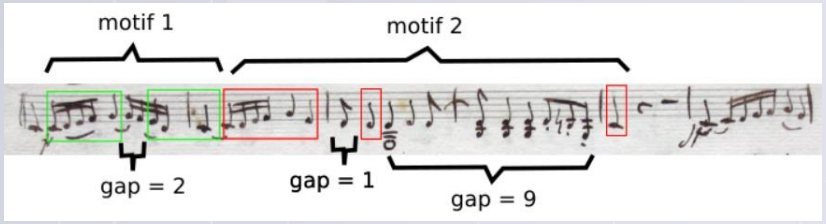
Thèse financée par une allocation doctorale de recherche de la Région Auvergne Rhône-Alpes (projet ARC5) dans la continuité du projet porté par l'ISH de Lyon « Le Magasin de Musique» (2015)

Objectifs

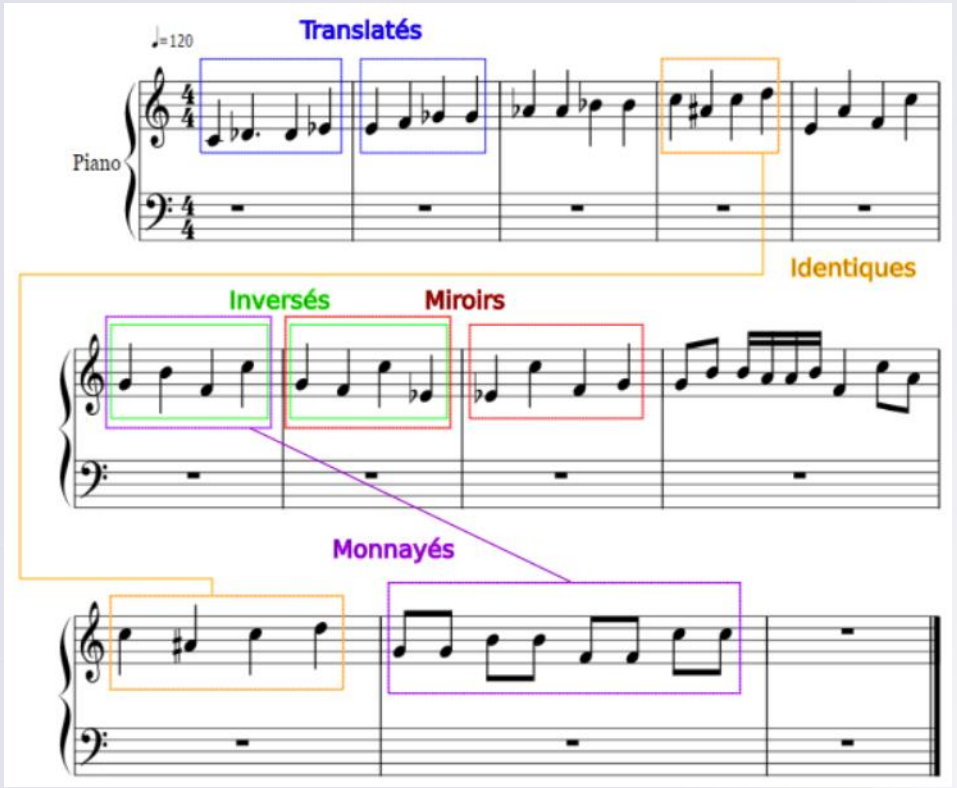
- Fournir des instruments d'assistance à l'analyse automatique de partitions musicales
  - Aide à la musicologie (grille de lecture, apprentissage des genres…)
  - Authentification de compositeurs (et des influences)
  - Recommandation d'une partition pour un auditeur
- Rechercher et identifier automatiquement des motifs musicaux (mélodiques et/ou rythmiques) sans a priori dans une partition manuscrite

Le motif musical

- Arrangement de notes contiguës ou non, mélodiques ou rythmiques, apparaissant selon une certaine fréquence dans la partition



Deux occurrences identiques contenant des gaps



Les variantes des motifs

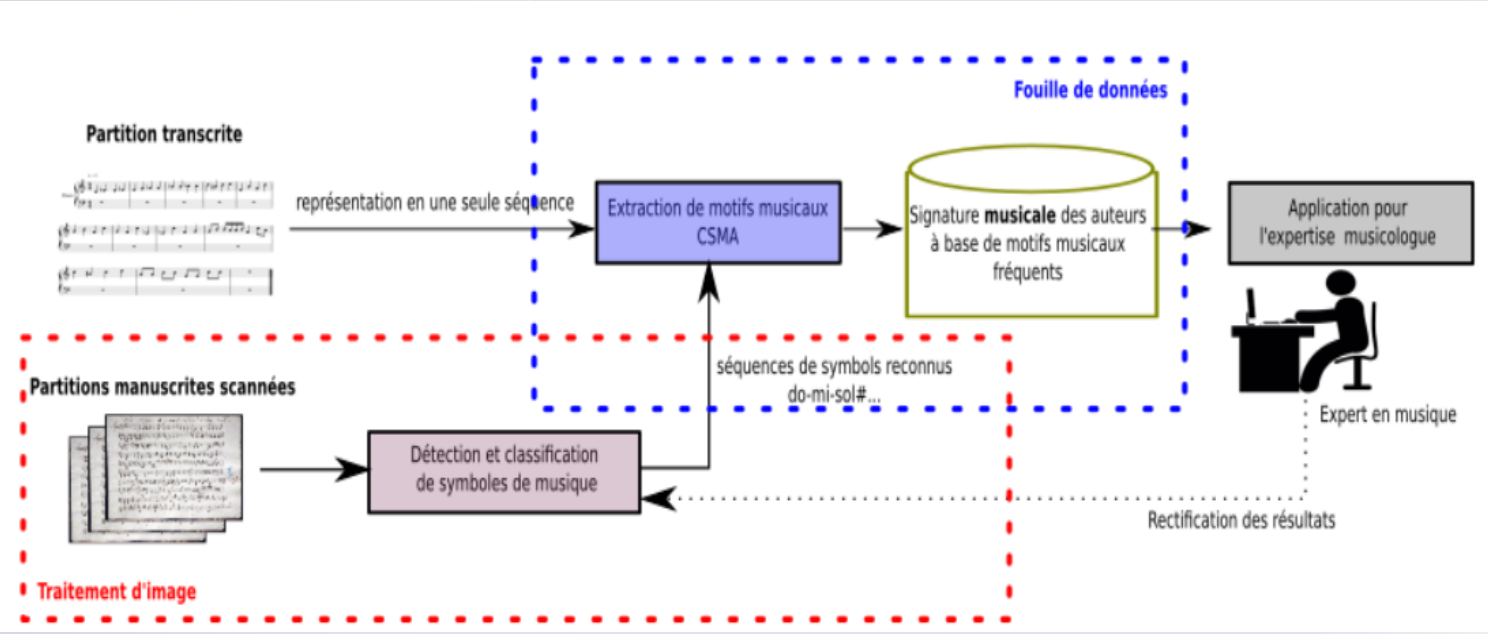
Méthodologie

- Proposer une méthodologie de **détection de motifs musicaux** issus de séquences transcrites à partir de **données brutes images** de partitions manuscrites.
- Proposer une approche générique de transformation de séquences visuelles symboliques en séquences textuelles par la reconnaissance de primitives « images » et leur **transcription symbolique**
- Exploiter les séquences et en **extraire automatiquement les motifs sans a priori** : proposer une méthodologie de caractérisation d'un auteur, d'une œuvre, d'un genre musical
- Développer une **approche ouverte à des corpus** non limités aux partitions inédites du 18ème siècle

Méthodologie

Mettre en place un système complet intégrant deux composantes principales :

- **Analyse d'images**: Réalisation d'un système de détection et de reconnaissance de primitives musicales manuscrites
- **Fouille de données**: Extraction de motifs musicaux fréquents et leurs variantes
- **Exploitation des bases de motifs** : études musicologiques, de genre et de compositions



Plan

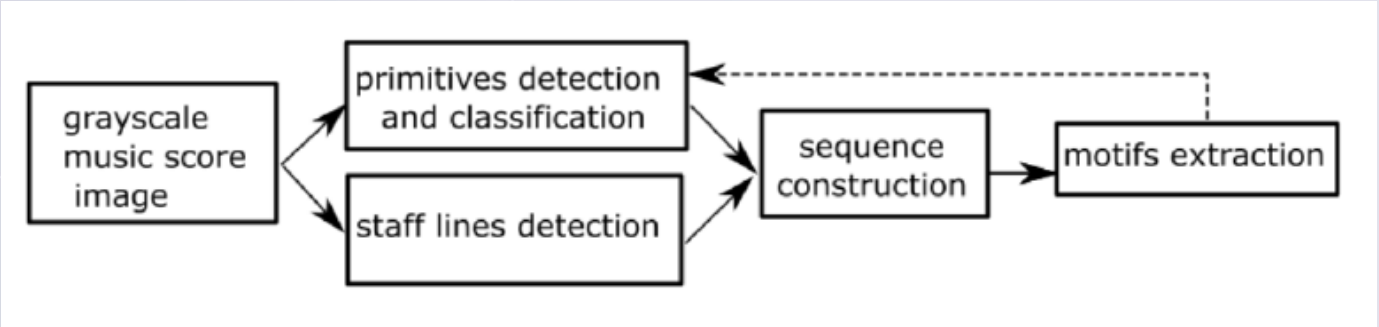
- **Traitement d'images de partitions manuscrites**
  - Détection des lignes de portée
  - Détection et classification des primitives musicales
  - Encodage des primitives
- **Algorithme d'extraction de motifs: CSMA**

  (Constrained String Mining Algorithm)
- **Evaluation du processus d'extraction de motifs**
  - Performance sur la chaîne complète : approche avec et sans gap
- **Conclusion & Perspectives**

Partie I: Extraction de primitives musicales manuscrites

Images des partitions manuscrites

- **Des symboles de grande densité et finement spatialisées** (en position relative sur les lignes de portées)
- **Un processus OMR simplifié:** prétraitement, détection/reconnaissance des primitives musicales principales, une reconstruction de séquences textuelles

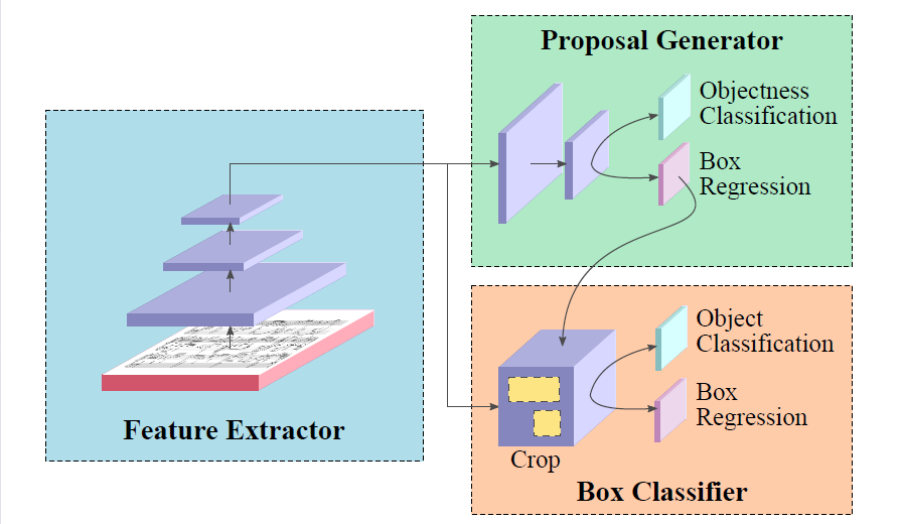


- **Pas de formalisation sémantique de reconstruction (partitions à une voie)**

- **Dataset de l'étude**
  - **MUSCIMA++ (ICDAR 2017)**: 140 partitions annotées au niveau primitives, 20 pages, 5 à 9 scripteurs
  - Des partitions à une seule voix, un seul instrument ~60 partitions
  - Le modèle peut être étendu sur d'autres données grâce à l'outil **MUSCIMarker**

FRP-CNN (Faster Region Proposal CNN)

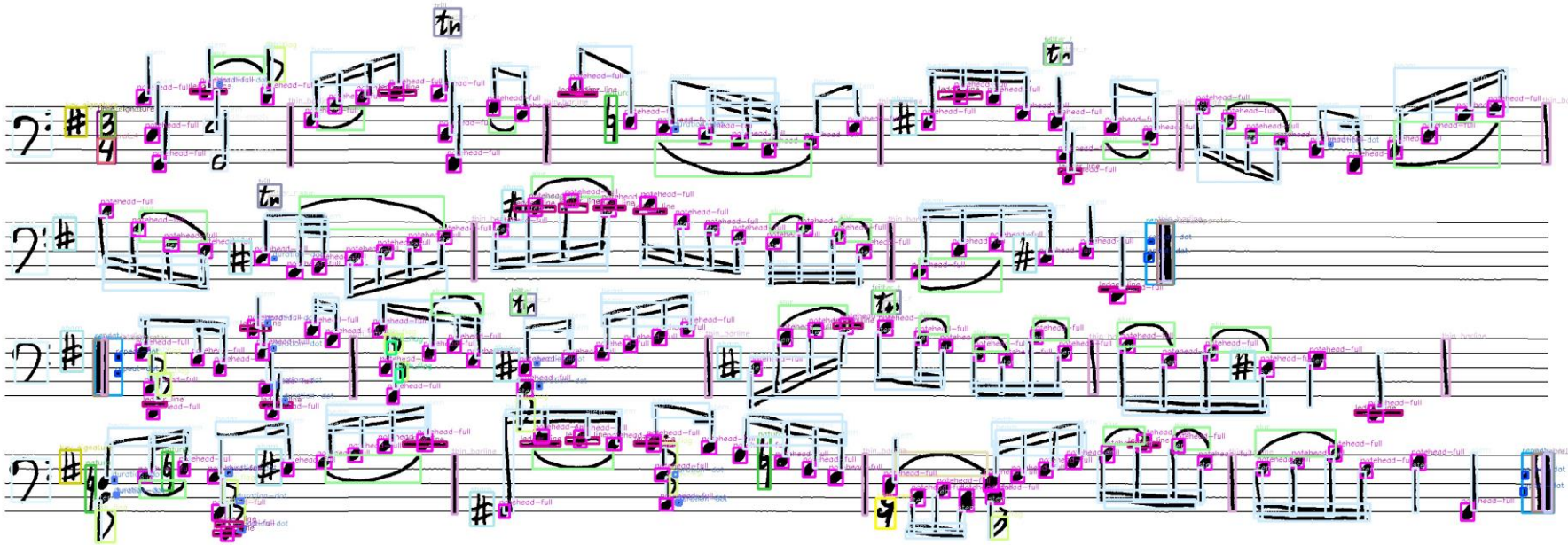
- **Partitions dégradées:** difficiles d'en détecter et d'en reconnaître les primitives musicales selon les scenarios traditionnels de reconnaissance
- **Partitions manuscrites mono-voix**: mécanismes basés HMM (*Pugin 2006*), CNN et RNN (*Chi 2015, Cavalo-Zaragosa 2017 (ISMIR), Van der Wel 2017 (ISMIR)*)
- **Détection et reconnaissance de primitives musicales** par Faster Region Proposal CNN (Fast R-CNN), *[Pacha et al., 2018]*
- **Principe d'une détection en 2 étapes (FRP-CNN)**
  - Génération de régions classifiées
  - Raffinement de la détection par classification fine
  - Partage du même réseau d'extraction de caractéristiques, Inception ResNet v2, initialisé par un réseau pré-entraîné sur ImageNet (*TensorFlow Object Detection API*)



Pacha, A., Choi, K.-Y., Couasnon, B., Ricquebourg, Y., Zanibbi, R., and Eidenberger, H. Handwritten music object detection: Open issues and baseline results. In International Workshop on Document Analysis Systems, 2018

FRP-CNN (faster Region Proposal CNN)

- **Des partitions et des données préparées:**
  - Réduction de la taille des images
  - Segmentation en lignes individuelles et réduction de format (image HxW, W~2H)
  - Jeu de données: 4700 Train/ 530 Valid/850 Test (6080 images)
  - Vocabulaire de 70 classes (Train)



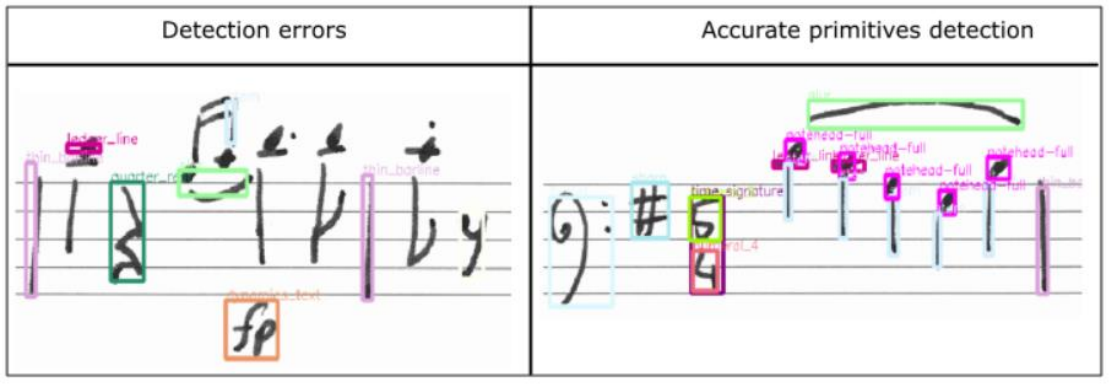
En sortie du RP-CNN, chaque primitive est définie par sa position, sa classe et ses dimensions

FRP-CNN (Faster Region Proposal CNN)

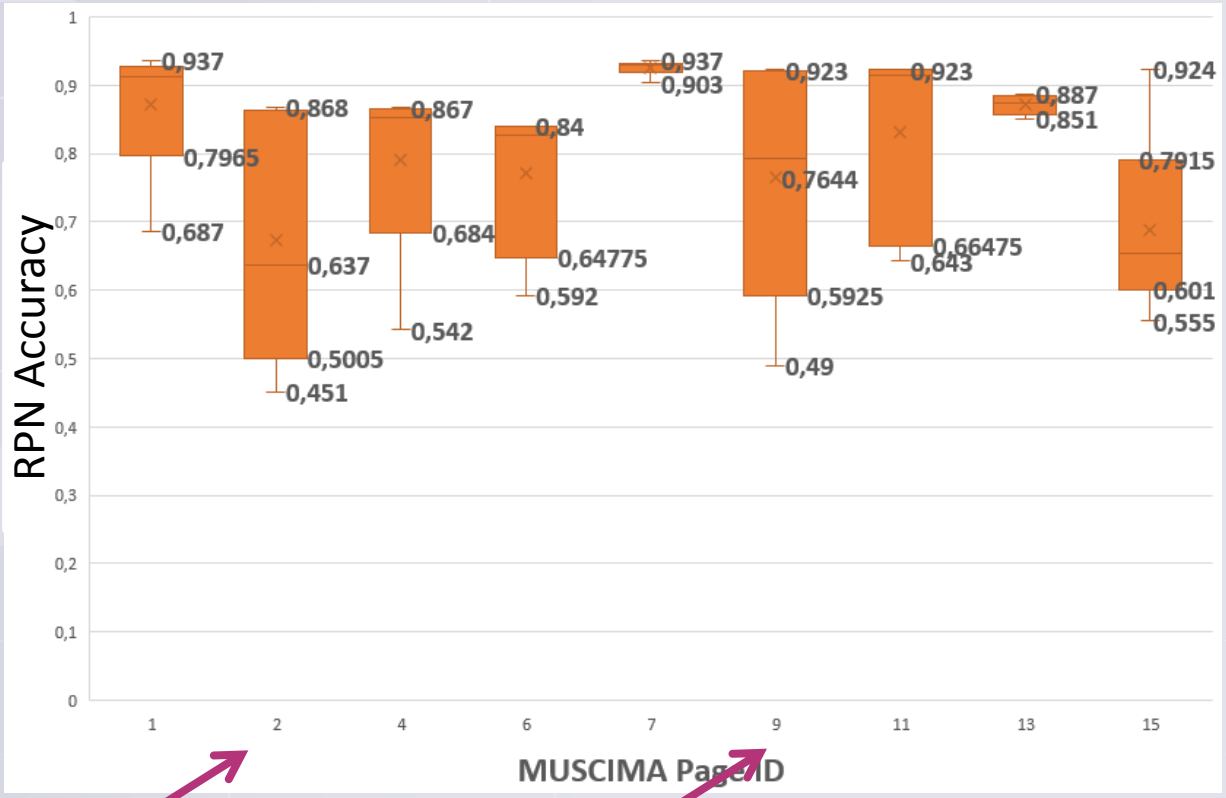
- On a choisi un sous-ensemble réduit de 11 primitives (parmi une variété de 105) couvrant l'information mélodique de la partition

mAP (mean Average Precision) du Faster R-CNN sur l'ensemble de test considérant 11 classes de primitives

Primitive	Min	avg	std deviation	max
tête de note pleine	0.426	0.888	0.188	1
tête de note vide	0,263	0,871	0,235	1
hampe	0,624	0,889	0,142	0,994
bémol	0,4	0,893	0,204	1
dièse	0,4	0,942	0,151	1
bécarre	0,25	0,88	0,217	1
Clef de Fa	1	1	0	1
clef de Sol	1	1	0	1
clef d'ut	1	1	0	1
lignes auxiliaires	0,203	0,613	0,23	0,883
barres de mesures	0,714	0,920	0,105	1

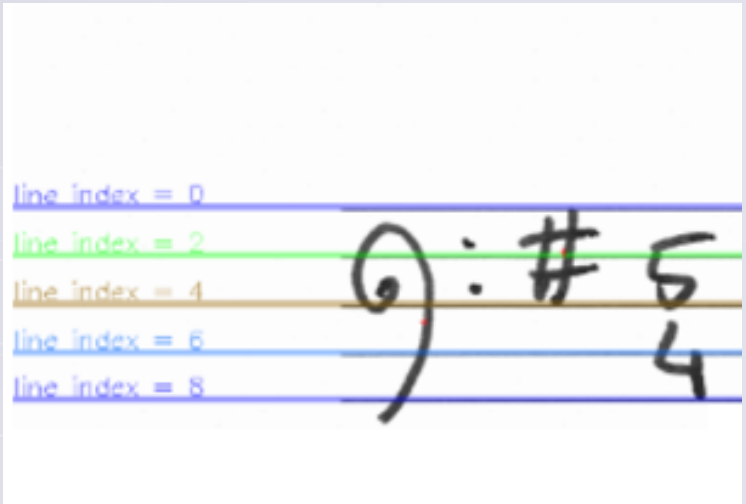


Performances du FRP-CNN sur quelques pages



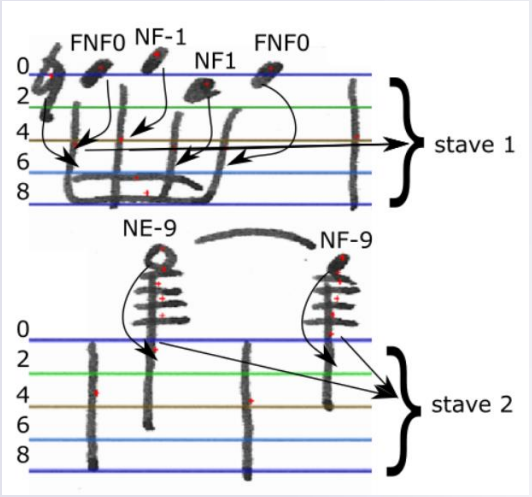
Notation des lignes de portée

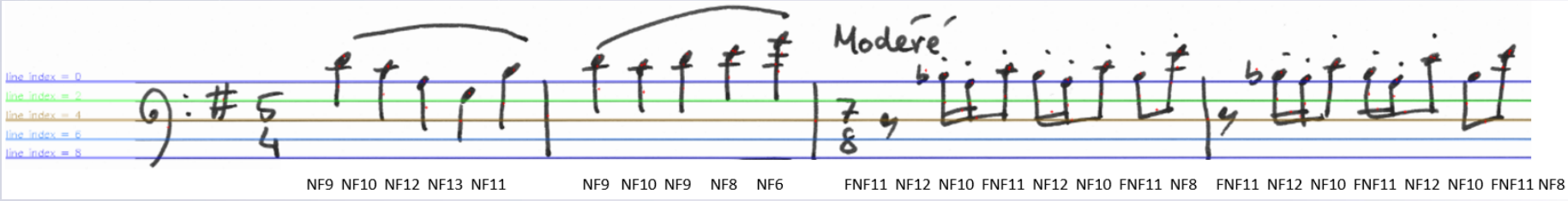
- **Détection des lignes de portée par projections**
- **Codage des lignes de portée**
  - La 1ère ligne en haut est codée 0
  - La 2ème ligne de portée est codée à 2
  - L'espace entre la 1ère et la 2ème ligne est codé à 1
  - Etc.



Encodage de la séquence de primitives

- Associer les têtes de notes et les altérations à la hampe la plus proche
- Ordonner la séquence de primitives suivant le sens de lecture (une voie)
- Encoder les primitives suivant l'expression régulière :
  ?(Alteration) (NoteType) (Position)
  - ? pour 0 ou 1 occurrence
  - L'altération peut être (F : bémol, N : bécarre, # : dièse)
  - NoteType peut être (tête de note: NF ou NE)
  - La position se réfère à l'index de la ligne de portée associée.

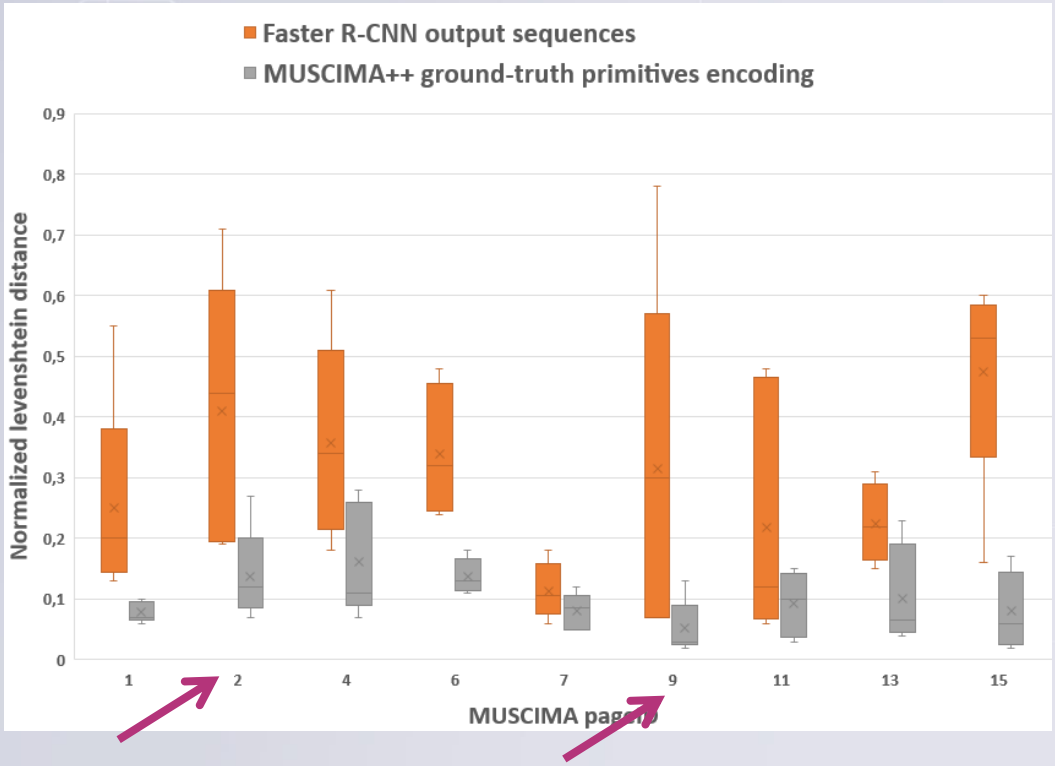




#NF0 ~ 78 Midi (noire sur la 1ère ligne accompagnée d'une altération #)

Evaluation de l'encodage

Estimation du coût d'édition entre les séquences de la vérité terrain (XML-GT) et les séquences extraites de la sortie du R-CNN (orange) puis et les séquences construites à partir des données annotées de MUSCIMA++ (gris)



Statistiques des valeurs normalisées de la distance de d'édition

En altérant la transcription, les erreurs de reconnaissance de primitives auront un impact sur la détection des motifs.

Comment contourner ces erreurs sans imposer une correction d'erreurs post-transcription ?

CSMA-Gap

Partie II: Algorithme d'extraction de motifs

CSMA - Introduction

- **Objectif**
  - Concevoir un algorithme de fouille capable de trouver des motifs musicaux à partir d'une séquence textuelle issue des alignements de primitives

- **Approche proposée**
  - CSMA (Constrained String Mining Algorithm) : découverte de motifs contigus et non contigus dans une ou plusieurs séquences
  - Prise en compte de contraintes de formes et de fréquence : fréquence minimale du motif, longueur minimale du motif, longueur maximale du motif, taille maximale de gaps et le nombre de gaps autorisés (pour les motifs non contigus)

CSMA - Algorithme

Algorithm 1: Constrained String mining Algorithm (**CSMA**)

Input : Sequence S, $minFrequency$,$maxGap$, $minLength$ and $maxLength$
Output: $\mathcal{F}$: The set of frequent patterns respecting constraints

```
1  begin
2      K = 1;
3      F_K = ∅;
       COMPUTE(1_freq); /* Compute patterns of length one then keep
        only frequent ones */
5      F_K = F_K ∪ 1_freq;
6      while F_K ≠ ∅ do
7          K = K + 1;
8          for m_i = (X, freq(X), P_i) ∈ F_{K-1} do
9              lastPosition = p_{iminFrequency} + len_{iminFrequency};
10             C = GEN_CANDIDATES(lastPosition, 1_freq);
11             for m_j = (Y, freq(Y), P_j) ∈ C do
12                 m_l = JOIN(m_i, m_j, maxGap, maxLength);
                    /*m_l = (Z, freq(Z), P_k) is a new pattern built by joining
                    m_i and m_j*/
13                 if freq(Z) ≥ minFrequency then
14                     F_K = F_K ∪ {m_l};
15                     /* We retain only frequent patterns */
16                 end
17             end
18         end
19     end
20     F = ⋃_{k≤K} F_k
21     FILTER(F, minLength);/* This function removes patterns that
        violates minLength constraint and keeps only the frequent ones */
22     return F;
23 end
```

Ensemble des candidats initiaux

Soit la séquence S=<ABABCDCABDCE> d'items de caractères

S=<**A**B**A**BCDC**A**BDCE>
Le motif **m** correspondant à **A** est défini par **m** = (**A**, freq(A) = 3, P = [(1,**1**), (3,**1**), (8,**1**)])

CSMA - Algorithme

Elagage de l'espace de recherche

Algorithm 1: Constrained String mining Algorithm (**CSMA**)

Input : Sequence S, *minFrequency*,*maxGap*, *minLength* and *maxLength*
Output: $\mathcal{F}$: The set of frequent patterns respecting constraints

1 **begin**
2 $K = 1$;
3 $\mathcal{F}_K = \emptyset$;
COMPUTE(1_freq); /* Compute patterns of length one then keep only frequent ones */
5 $\mathcal{F}_K = \mathcal{F}_K \cup 1\_freq$;
6 **while** $\mathcal{F}_K \neq \emptyset$ **do**
7 $K = K + 1$;
8 **for** $m_i = (X, freq(X), P_i) \in \mathcal{F}_{K-1}$ **do**
9 $lastPosition = p_{iminFrequency} + len_{iminFrequency}$;
$\mathcal{C}$ = **GEN_CANDIDATES**($lastPosition$, 1_freq);
11 **for** $m_j = (Y, freq(Y), P_j) \in \mathcal{C}$ **do**
12 m_l = **JOIN**($m_i, m_j, maxGap, maxLength$); /*$m_l = (Z, freq(Z), P_k)$ is a new pattern built by joining m_i and m_j*/
13 **if** $freq(Z) \geq minFrequency$ **then**
14 $\mathcal{F}_K = \mathcal{F}_K \cup \{m_l\}$;
15 /* We retain only frequent patterns */
16 **end**
17 **end**
18 **end**
19 **end**
20 $\mathcal{F} = \bigcup_{k \leq K} \mathcal{F}_k$
21 **FILTER**($\mathcal{F}, minLength$);/* This function removes patterns that violates *minLength* constraint and keeps only the frequent ones */
22 return $\mathcal{F}$;
23 **end**

Algorithm 2: *GEN_CANDIDATES*

Input : a position *position*, a set of patterns $\mathcal{M}$
Output: The set of candidate patterns $\mathcal{C}$

1 **begin**
2 $\mathcal{C} = \emptyset$;
3 **for** $m = (X, freq(X), P = [\bigcup_{i \leq freq(X)} (p_i, len_i)]) \in \mathcal{M}$ **do**
4 $i = 1$;
5 $isCandidate = False$;
6 **while** $(i \leq freq(X) \wedge isCandidate = False)$ **do**
7 **if** $p_i \geq position$ **then**
8 $isCandidate = True$;
9 **end**
10 $i = i + 1$;
11 **end**
12 **if** $isCandidate = True$ **then**
13 $\mathcal{C} = \mathcal{C} \cup \{m\}$;
14 **end**
15 **end**
16 return $\mathcal{C}$;
17 **end**

CSMA - Algorithme

Algorithm 1: Constrained String mining Algorithm (**CSMA**)

Input : Sequence S, *minFrequency*,*maxGap*, *minLength* and *maxLength*
Output: $\mathcal{F}$: The set of frequent patterns respecting constraints

1 **begin**
2 $K = 1$;
3 $\mathcal{F}_K = \emptyset$;
4 **COMPUTE**(1_freq); /* Compute patterns of length one then keep only frequent ones */
5 $\mathcal{F}_K = \mathcal{F}_K \cup 1\_freq$;
6 **while** $\mathcal{F}_K \neq \emptyset$ **do**
7 $K = K + 1$;
8 **for** $m_i = (X, freq(X), P_i) \in \mathcal{F}_{K-1}$ **do**
9 $lastPosition = p_{iminFrequency} + len_{iminFrequency}$;
10 $\mathcal{C}$ = **GEN_CANDIDATES**($lastPosition$, 1_freq);
11 **for** $m_j = (Y, freq(Y), P_j) \in \mathcal{C}$ **do**
12 m_l = **JOIN**($m_i, m_j, maxGap, maxLength$); /*$m_l = (Z, freq(Z), P_k)$ is a new pattern built by joining m_i and m_j*/
13 **if** $freq(Z) \geq minFrequency$ **then**
14 $\mathcal{F}_K = \mathcal{F}_K \cup \{m_l\}$;
15 /* We retain only frequent patterns */
16 **end**
17 **end**
18 **end**
19 **end**
20 $\mathcal{F} = \bigcup_{k \leq K} \mathcal{F}_k$
21 **FILTER**($\mathcal{F}, minLength$);/* This function removes patterns that violates *minLength* constraint and keeps only the frequent ones */
22 return $\mathcal{F}$;
23 **end**

Notion de gap: Item spécial qui peut prendre n'importe quelle valeur.

S=<ABAB**CDC**AB**D**CE>

Le motif m correspondant à C*D, où * exprime le **gap** est défini par
m=(C*D, freq(C*D)=2, P = [(5,2),(7,4)])
Avec une longueur de 4, le gap entre C et D vaut 2.

CSMA - Algorithme

Algorithm 1: Constrained String mining Algorithm (**CSMA**)

Input : Sequence S, *minFrequency*,*maxGap*, *minLength* and *maxLength*
Output: $\mathcal{F}$: The set of frequent patterns respecting constraints

1 **begin**
2 $K = 1$;
3 $\mathcal{F}_K = \emptyset$;
4 **COMPUTE**(1_freq); /* Compute patterns of length one then keep only frequent ones */
5 $\mathcal{F}_K = \mathcal{F}_K \cup 1\_freq$;
6 **while** $\mathcal{F}_K \neq \emptyset$ **do**
7 $K = K + 1$;
8 **for** $m_i = (X, freq(X), P_i) \in \mathcal{F}_{K-1}$ **do**
9 $lastPosition = p_{iminFrequency} + len_{iminFrequency}$;
10 $\mathcal{C} =$ **GEN_CANDIDATES**($lastPosition$, 1_freq);
11 **for** $m_j = (Y, freq(Y), P_j) \in \mathcal{C}$ **do**
12 $m_l =$ **JOIN**($m_i, m_j, maxGap, maxLength$);
/* $m_l = (Z, freq(Z), P_k)$ is a new pattern built by joining m_i and m_j */
13 **if** $freq(Z) \geq minFrequency$ **then**
14 $\mathcal{F}_K = \mathcal{F}_K \cup \{m_l\}$;
15 /* We retain only frequent patterns */
16 **end**
17 **end**
18 **end**
19 **end**
20 $\mathcal{F} = \bigcup_{k \leq K} \mathcal{F}_k$
21 **FILTER**($\mathcal{F}, minLength$);/* This function removes patterns that violates *minLength* constraint and keeps only the frequent ones */
22 return $\mathcal{F}$;
23 **end**

Notion de gap: Item spécial qui peut prendre n'importe quelle valeur.

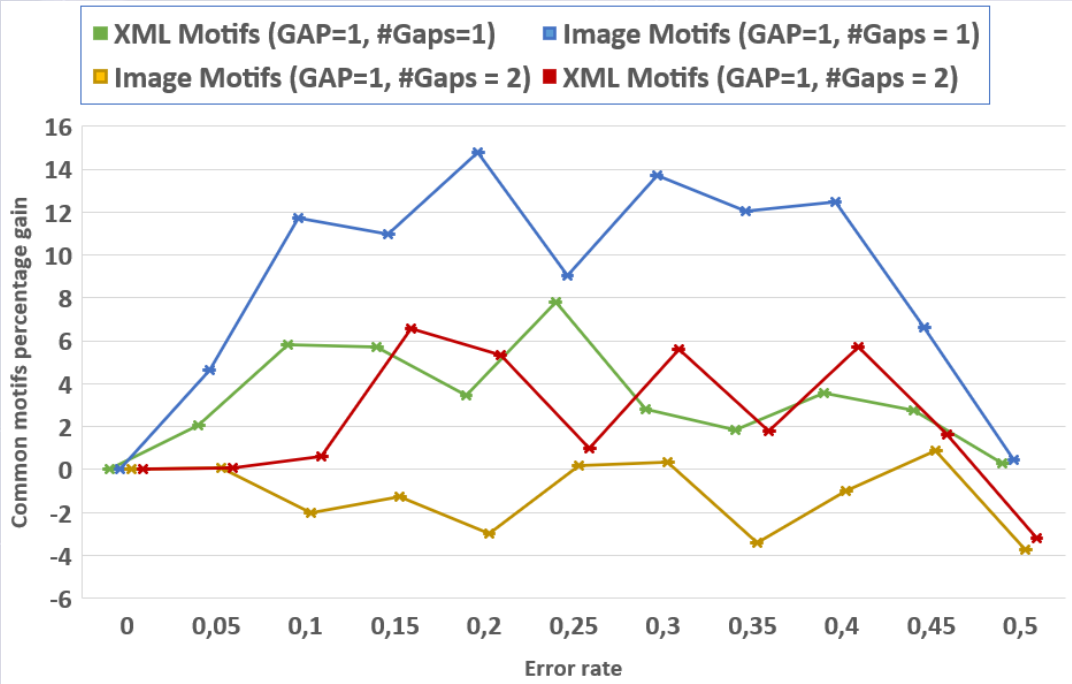
S=<ABAB**CDC**AB**D**CE>

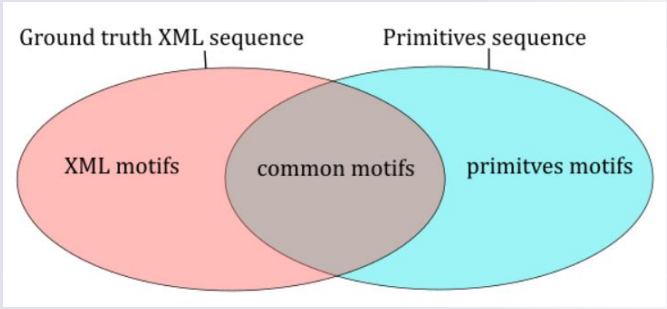
Le motif m correspondant à C*D, où * exprime le **gap** est défini par
m=(C*D, freq(C*D)=2, P = [(5,2),(7,4)])
Avec une longueur de 4, le gap entre C et D vaut 2.

Partie III: Evaluation du processus d'extraction de motifs

Evaluation du gap dans des séquences bruitées

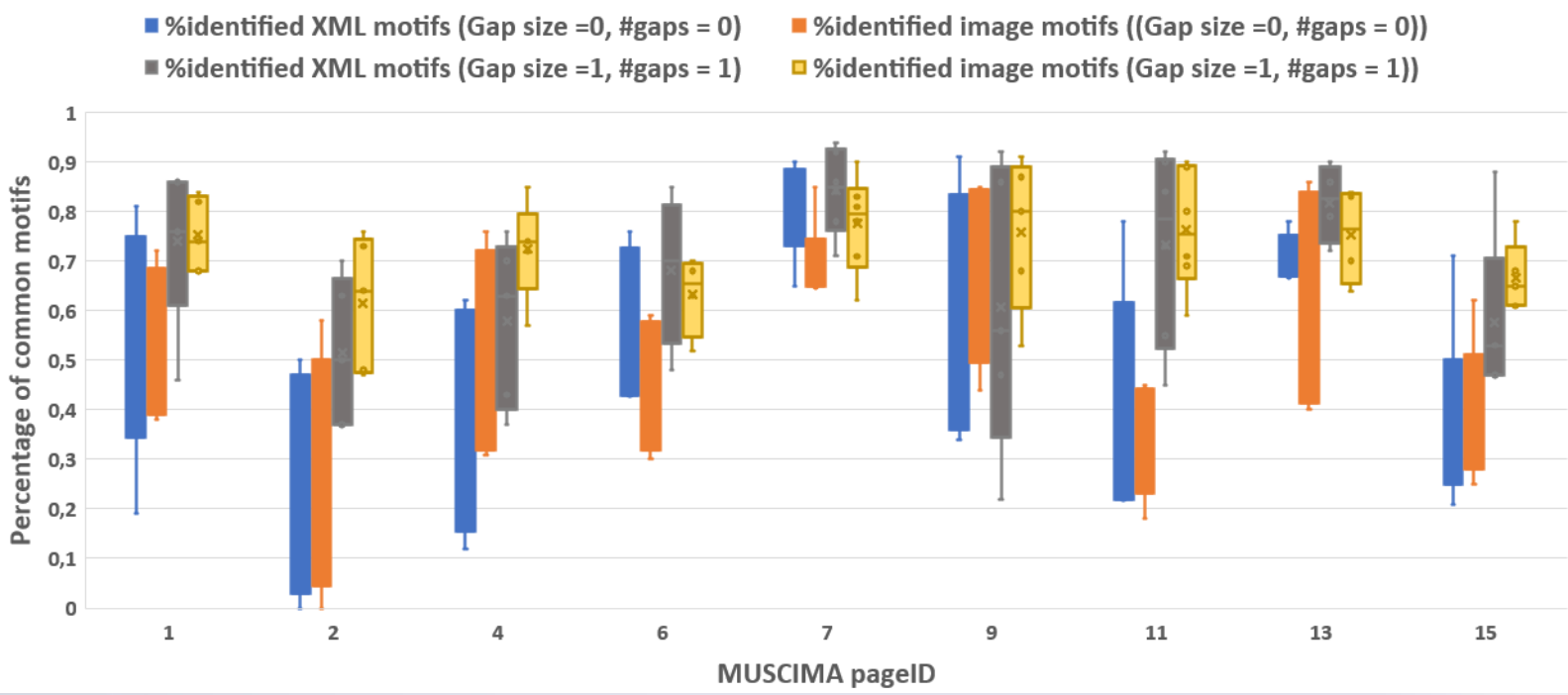
1. A partir de l'ensemble de séquences correctes XML-GT, on calcule les motifs
2. On alterne aléatoirement des valeurs des séquences XML-GT pour simuler les erreurs du R-CNN (de 0 à 50%)
3. On recalcule les motifs en faisant varier les valeurs de Gap (1 et 2) et leur nombre (1 et 2)





Evaluation du gap dans les séquences issues du FRP-CNN

- **Estimation du nombre de motifs communs** entre les séquences de la vérité terrain (XML-GT) prise comme référence et les séquences extraites de la sortie du R-CNN (bleu Gap =0 ; gris Gap=1) et entre les séquences construites à partir de la sortie du R-CNN prises comme références et les séquences XML-GT (orange Gap=0 ; jaune Gap = 1)



Conclusion & Perspectives

- Démonstration de l'utilité du gap pour réduire l'impact des erreurs du RPN sur l'extraction de motifs (~ 22% d'amélioration dans la détection des motifs corrects)
- Extension actuel du modèle sur d'autres jeux de données : les partitions inédites du 18ème avec des motifs de plus grande taille, une adaptation des processus (R-CNN, paramétrage de CSMA) et validation musicologique
- Exploitation du dispositif autour de la mise en place d'un système d'identification d'auteurs