

Traitement d'une partition

COLOMBO Pierre - MULLER Paul

6 novembre 2016

Rapport projet long 2A
Traitement d'une partition



Table des matières

1	Introduction	5
2	Cahier des charges	6
3	Prétraitemet de la partition	7
3.1	Introduction	7
3.2	Redressement de la partition	9
3.3	Recherche des portées et des caractéristiques	10
3.3.1	Le cas des portées	10
3.3.2	Calcul de l'épaisseur des lignes de portée	11
3.4	Subdivision de la partition en sous partitions	12
3.4.1	Séparation de l'image en sous image représentant chacune une partition	12
3.4.2	Calcul des coordonnées de début et de fin pour chaque sous image	13
3.4.3	Bilan	13
3.5	Effacement des portées	13
3.6	Mise en boite des notes	14
3.6.1	Introduction	14
3.6.2	Carte des empans	14
3.6.3	Mise en place des boites englobantes	16
4	Analyse des boites englobantes	18
4.1	Reconnaissance des Symboles	18
4.1.1	Présentation des différentes Méthodes	18
4.1.2	Reconnaissance de notes	19
5	Améliorations et Optimisations	28
5.1	Multithreading	28
5.2	Programmation GPU	29
5.3	Amélioration pour la Robustesse	29
5.4	Développement de l'interface pour la présentation des résultats	30
5.4.1	L'interface console	30
5.4.2	L'interface graphique	30
5.5	Conclusions	31
6	Choix du C++	32
7	Test	32
7.1	Le problème de la clef de sol et de l'armure	32
7.2	La partition : Le caillou	34
7.3	Test sur une sous-portée	35

7.4	Test sur toute la partition : Le Caillou	36
7.4.1	Les résultats	36
7.4.2	Analyse des résultats	38
7.5	Analyse de l'image test	38
7.6	Analyse à l'aide de Matrices de Confusion	40
7.6.1	Matrice de confusion de "Je Regrette"	40
7.6.2	Matrice de confusion "Le Caillou"	42
7.6.3	Matrice de confusion de "Frère Jacque"	43
7.6.4	Conclusion	45
8	Conclusions	46
9	Annexe 1 : Répertoire des fonctions codées	47
9.1	Le main	48
9.2	Correction du biais	51
9.2.1	Correction du biais multithreadé sans fft	51
9.2.2	Correction du biais avec fft	51
9.3	Detection de l'interligne	51
9.4	Recherche des Xis	52
9.5	Sous Image Portee	52
9.6	Effacement Portee	52
9.7	Histogramme Largeur Ligne	52
9.8	Fusion verticale	52
9.9	Carte Empans	52
9.10	Carte Finale des Empans	52
9.11	Selection Unique Empan	52
9.12	Selection Plusieurs Empans	52
9.13	Emboitage Image	53
9.14	Reconnaissance Notes	53
9.15	Affichage avec texte	53
9.16	Interface console	53
10	Annexe 2 : Délivrables	54
11	Remerciements	55

Table des figures

1	Partition en couleur	7
2	Partition convertie avec notre algorithme en moyenne quadratique	8
3	Image de travail	8
4	Schéma de principe	9
5	Image de travail corrigée	10
6	Profils verticaux	11
7	Première portée	12
8	Seconde portée	12
9	Troisième sous portée	13
10	Première sous image sans lignes de portées	14
11	Carte des empans phase quatre de la sous image 1	15
12	Carte finale des empans de la sous image 1	16
13	Boites englobantes	17
14	Croche 1	18
15	Croche 2	18
16	Ébauche de l'Interface Graphique - QtDesigner	31
17	Test : Mise en évidence du problème de la clef	33
18	Sous image du Test : Mise en évidence du problème de la clef	33
19	Partition de test : Le Caillou	35
20	Sous image de la partition de test : Le Caillou	35
21	Partition de test : "Je regrette"	38
22	Partition de test : Le Caillou	40
23	Partition de test : Le Caillou	42
24	Partition de test : "Frère Jacque"	43
25	Diagramme résumant la démarche du code	47

1 Introduction

L'un des problèmes majeurs de notre Société depuis les années 1980 est le passage d'un monde de données principalement sur Papier à une base de données Numérisée, passage toujours non-accompli. La numérisation des données utilisées présente des disparités de domaines : l'administration gouvernementale, par exemple, utilise principalement les systèmes numériques, les copies papiers assurant la liaison individu / administration. La musique est un des domaines dans lequel la numérisation n'est que partielle, car très contraignante : il faut rentrer les bonnes notes, les bons temps, au bon endroit : le travail à fournir pour un Homme est titanique, et ce pour une simple partition. En numériser une grande quantité demande un temps et un effort que toute personne raisonnable ne veut ni ne peut fournir.

C'est dans cette optique que nous, Paul Müller et Pierre Colombo, avons décidé de créer un logiciel effectuant cette numérisation automatiquement : l'opérateur humain n'a qu'à fournir une image de la partition, quelques renseignements, et le logiciel la lui numérisera. Nous profiterons aussi du fait d'avoir une partition numérisée pour proposer à l'utilisateur des services utiles, comme par exemple la lecture audio de la partition, ou la transposition en une autre tonalité, travail extrêmement fastidieux.

Notre logiciel permettra donc de simplifier et d'accélérer le travail de tout musicien, débutant ou confirmé, souhaitant jouer une partition sur papier. Nous attendons de très bons résultats de notre travail, une forte utilisation, et visons un très large public, que nous accèderons via une application pour smartphones.

2 Cahier des charges

Notre projet est la création d'une application pour smartphones permettant la lecture interactive de partitions fournies par scan, ou prise de photos. Notre cahier des charges comporte donc deux parties distinctes : une concernant le logiciel de reconnaissance en lui-même, une autre concernant l'application.

Cahier des charges du Logiciel de Reconnaissance :

1. Lecture d'autant de formats d'images que possible (.bmp, .jpeg, .png ...)
2. Reconnaissance des notes et autres symboles dans une partition.
3. Reconnaissance de l'ordre d'enchainement de ces notes dans la partition globale.
4. Robustesse de la reconnaissance : Le taux d'erreur doit être le plus faible possible.
5. Capacité de modifier les notes, de transposer et de faire jouer la partition par le système.
6. Opérations menées aussi rapidement que possible avec un encombrement mémoire minimal.

Cahier des charges de l'Application :

1. Effectue le lien entre le logiciel et l'utilisateur, et doit donc comporter toutes les fonctionnalités du logiciel.
2. Possibilité de prendre en photo une partition à partir du logiciel.
3. Possibilité de choisir des partitions sur Internet.
4. Possibilité de choisir des partitions dans la mémoire du téléphone.
5. Menus intuitifs et peu envahissants permettant à l'utilisateur de facilement lire et jouer la partition lui-même.
6. Mode d'édition et de lecture.
7. Possibilité de scanner page par page une longue partition, et de lier entre elles les pages dans un même fichier.

Ce cahier des charges est voué à être étendu au fil de l'avancement du projet.

3 Prétraitemet de la partition

3.1 Introduction

Dans cette partie nous allons expliquer le prétraitemet de la partition. Cette partie est principalement présentée à l'aide d'images exposant nos résultats. Pour l'aspect mathématique (produits de convolution, choix des filtres et calcul matri-ciel) il suffit de se référer au code.

Le code a tout d'abord été développé sous le logiciel de Matlab puis ensuite en C++.

La première étape de notre traitemet est de convertir des images en couleur en images bicolores noir et blanc. Pour cela nous avons écrit notre propre algorithme de correction. Cet algorithme est basé sur un calcul de moyenne quadratique des couleurs avec insertion d'un seuil arbitraire. Cela nous permet de mener la suite de notre travail sur des partitions en noir et blanc. Dans la suite nous travaillerons sur des image bicolores sous forme de matrices contenant des 0 (blanc) et des 1 (couleur noir). Nous supposerons que les portées et les symboles sont en blanc.

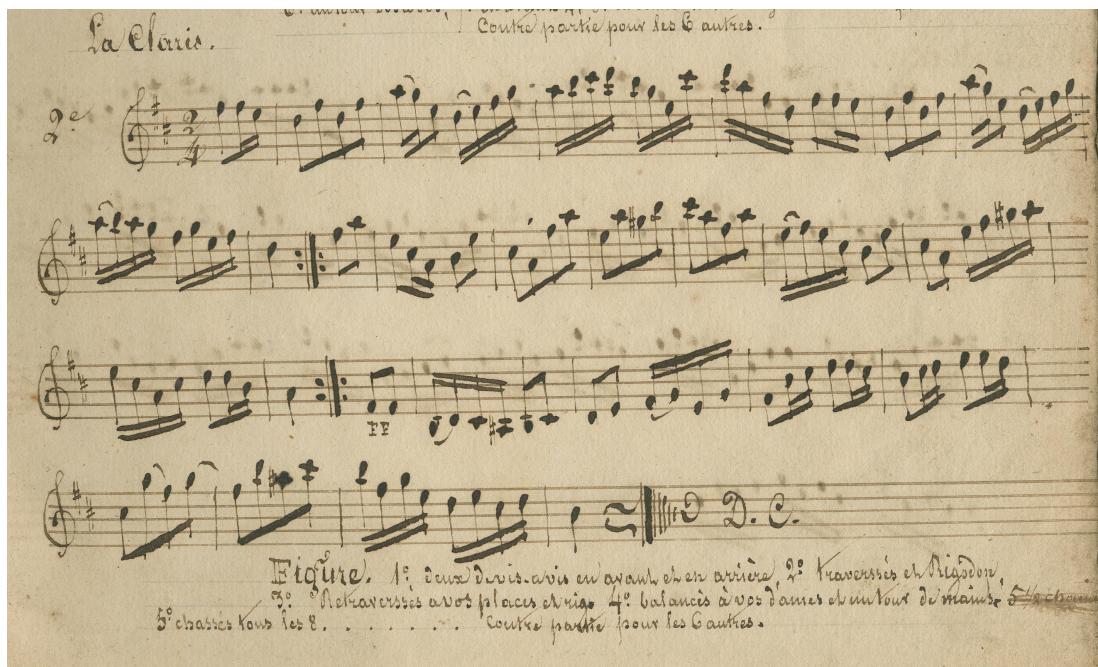


FIGURE 1 – Partition en couleur



FIGURE 2 – Partition convertie avec notre algorithme en moyenne quadratique

Pour illustrer nos propos et présenter nos résultats ; nous avons choisi de les illustrer à l'aide de tests nous avons choisis pour cela d'utiliser une image plus simple. Dans toute la suite nous travaillerons sur la partition suivante volontairement tournée d'un angle de 15° :

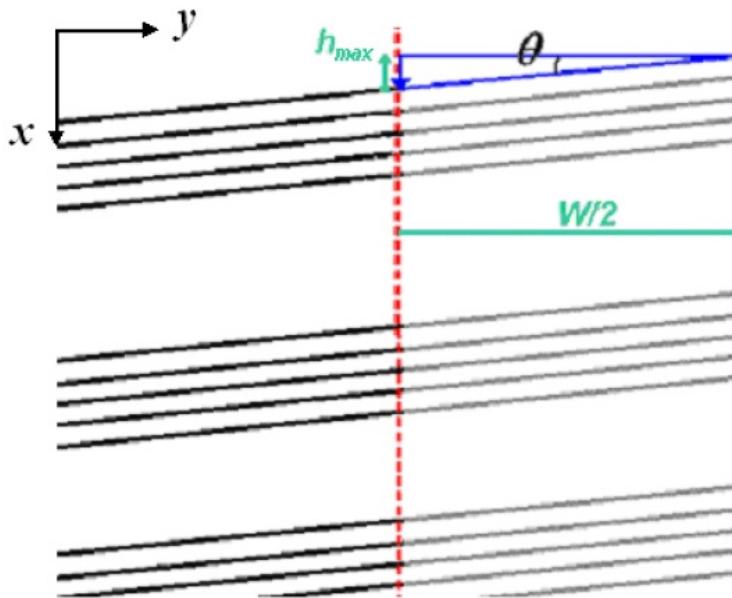


FIGURE 3 – Image de travail

3.2 Redressement de la partition

Dans un premier temps nous avons tenu compte de l'inclinaison éventuelle de la partition. En effet, dans le cas d'un scan ou d'une photo, il n'est pas impossible que la partition soit un peu inclinée ce qui pourrait nuire à la suite du traitement. Pour cela on note que

$$\theta = \arctan\left(\frac{2 * h_{max}}{\omega}\right) \quad (1)$$



de l'image.jpg –

FIGURE 4 – Schéma de principe

En remarquant que le premier pic non nul de l'intercorrélation entre la première moitiée et entre la seconde moitiée de l'image nous fournit h_{max} . On peut très facilement en déduire θ .

Il suffit ensuite d'appliquer une rotation, que l'on assimilera ici à une transformation linéaire, à l'image d'origine pour se ramener à une image sans aucun biais et parfaitement droite (on fait l'hypothèse de petits angles)

$$I(x, y) = I(x - 2 * h_{max} * y, y) \quad (2)$$

On obtient donc l'image corrigée suivante



FIGURE 5 – Image de travail corrigée

On avait, en première implémentation de l'intercorrélation utilisée pour trouver hmax, utilisé une implémentation naïve de l'intercorrélation, c'est à dire une double boucle for représentant la double somme de la formule définissant l'intercorrélation en 2D :

$$\sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(n, m)g(x - n, y - m)$$

Il existe en fait des algorithmes beaucoup plus performants utilisant l'équivalence produit/convolution induit par la transformée de fourier. Néanmoins, ces algorithmes requièrent une connaissance en programmation à un extrêmement bas niveau (problèmes d'adressage), et il nous a été impossible de finir de les implémenter correctement avant la fin du projet.

3.3 Recherche des portées et des caractéristiques

3.3.1 Le cas des portées

Pour rechercher les lignes nous faisons la somme horizontalement des pixels ce qui nous permet d'obtenir le profil ci-dessous :

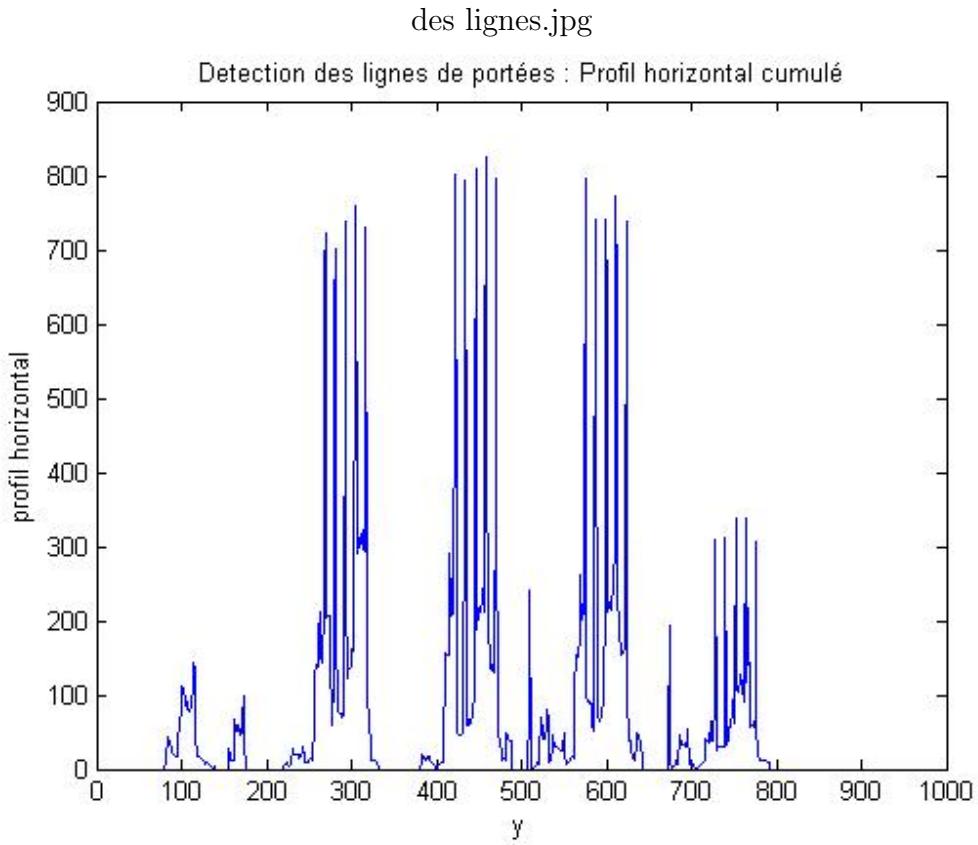


FIGURE 6 – Profils verticaux

On remarque que le profil vertical de la partition nous permet de déterminer avec exactitude la position des portées (on repère les groupes de 5 maximaux "pas trop espacé"). En repérant les différents pics, on peut repérer la position des lignes. Nous notons qu'il faut mettre un seuil arbitraire : nous avons choisis $2/3$ du plus grand maximum. De plus nous notons que dans le cas de notre partition la dernière portée, trop courte, ne sera pas détecté. Pour palier à ce problème nous comptons faire intervenir l'utilisateur dans notre application.

3.3.2 Calcul de l'épaisseur des lignes de portée

Nous nous attaquons ensuite au calcul de l'épaisseur des portées. Pour se faire on cherche le maximum de l'histogramme des longueurs des empans noirs. De plus nous avons un avantage : la localisation des portées est déjà connue ce qui nous permet de nous restreindre au voisinage de la troisième ligne de portée. Cette détermination va nous aider pour la suite des traitements des partitions.

3.4 Subdivision de la partition en sous partitions

3.4.1 Séparation de l'image en sous image représentant chacune une partition

Nous allons dans cette partie subdiviser l'image de la partition en sous images chacune d'entre elle correspondant à une et une seule portée. Cela nous permettra plus de liberté et d'efficacité :

- traiter chaque sous portée indépendamment va nous permettre de minimiser les erreurs
 - éventuellement nous pourrons faire les traitements suivants sur plusieurs coeurs ou plusieurs threads

The musical score for "Je Regrette" consists of a single staff of music in common time, 2/4 time, or 4/4 time. The key signature is one sharp (F#). The melody is composed of eighth and sixteenth notes. Measure numbers 2, 3, 4, 5, 6, 7, 8, and 9 are indicated above the staff. The first measure begins with a sixteenth note followed by a eighth note.

FIGURE 7 – Première portée

A musical score page showing measures 6 through 12. The score consists of two staves. The top staff uses a treble clef and a 6/8 time signature, starting with a B-flat. The bottom staff uses a bass clef and a 6/8 time signature, starting with a C. Measure 6: Treble staff has eighth-note pairs (B-flat, A), (G, F), (E, D), (C, B-flat). Bass staff has eighth notes (F, E, D, C, B-flat, A). Measure 7: Treble staff has eighth-note pairs (G, F), (E, D), (C, B-flat). Bass staff has eighth notes (D, C, B-flat, A, G, F). Measure 8: Treble staff has eighth-note pairs (E, D), (C, B-flat). Bass staff has eighth notes (B-flat, A, G, F, E, D). Measure 9: Treble staff has eighth-note pairs (C, B-flat), (A, G). Bass staff has eighth notes (A, G, F, E, D, C). Measure 10: Treble staff has eighth-note pairs (A, G), (F, E). Bass staff has eighth notes (F, E, D, C, B-flat, A). Measure 11: Treble staff has eighth-note pairs (F, E), (D, C). Bass staff has eighth notes (D, C, B-flat, A, G, F). Measure 12: Treble staff has eighth-note pairs (D, C), (B-flat, A). Bass staff has eighth notes (B-flat, A, G, F, E, D). Measure 13: Treble staff has eighth-note pairs (B-flat, A), (G, F). Bass staff has eighth notes (G, F, E, D, C, B-flat).

FIGURE 8 – Seconde portée

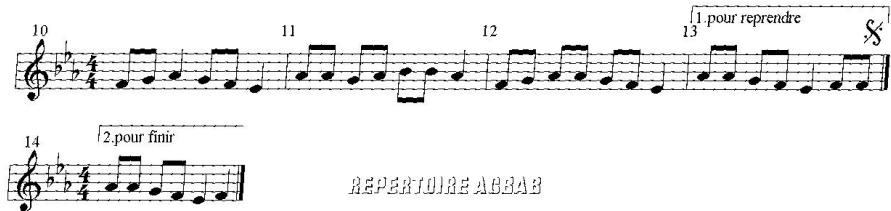


FIGURE 9 – Troisième sous portée

Remarquons que le découpage que nous avons effectuée est assez optimal : en effet il n'y a pas de recouvrement des sous images entre elles et chaque image isole bien une et une seule portée.

Remarquons aussi que comme signalé précédemment la detection de la dernière portée (la quatrième) est problématique et est effectivement ignorée par notre algorithme.

3.4.2 Calcul des coordonnées de début et de fin pour chaque sous image

On réalise une projection verticale des zones de l'image situées autour des zones correspondant à chaque ligne de portée. Sur chaque zone la somme des pixels est environ égale à cinq fois l'épaisseur moyenne. A l'aide de ce critère on peut donc déterminer quelles sont les coordonnées de début et de fin de chaque jeu de portées.

3.4.3 Bilan

A ce stade nous connaissons de nombreux paramètres géométriques de notre partition. Dans la suite nous cherchons à isoler précisément les symboles de la partition caractérisés par des empans verticaux.

3.5 Effacement des portées

L'effacement des portées est une étape indispensable pour nous permettre d'isoler finement les différents symboles présents sur la partition. C'est ici une phase assez délicate puisque nous devons effacer le maximum de pixels noirs pour faciliter la suite du traitement tout en gardant les formes géométriques reconnaissable (les ovales des notes ne doivent pas être tronqués etc...). Pour faciliter le traitement nous avons mis en place un algorithme de poursuite

des portées permettant de tenir compte des distorsions de l'image.
Le résultat obtenu est bien conforme à nos attentes :

sans porte.jpg



FIGURE 10 – Première sous image sans lignes de portées

3.6 Mise en boite des notes

3.6.1 Introduction

La détection des segments verticaux doit surmonter deux difficultés majeures : les ruptures de segment et le biais (légère rotation résiduelle de la partition). Les ruptures de segment, c'est-à-dire les interruptions durant quelques pixels, sont très fréquentes dans les documents originaux, et sont parfois introduites par la numérisation. Nous allons maintenant décrire en détail les différentes phases de la méthode. Tout d'abord nous allons créer une carte des empans en 5 phase. Ensuite la carte des empans va nous permettre grâce entre autre à des algorithmes de croissance de trouver des boites.

3.6.2 Carte des empans

Nous cherchons à avoir une carte précise des empans. On distinguera dans la suite le terme "empan vertical", qui désigne une colonne de pixels noirs connexes, et le terme "segment vertical", qui fait référence à une ligne verticale d'épaisseur supérieure ou égale à 1, donc constituée d'empans verticaux contigus. On peut également voir le segment comme une succession d'empans noirs horizontaux, de faible longueur, et verticalement alignés.

Les symboles caractérisés par un segment vertical sont les notes qui possèdent une hampe (toutes les notes exceptées les rondes), les altérations et les appogiatures.

Première phase : La première phase consiste à parcourir toute l'image, colonne par colonne, et à créer une carte codant la longueur des empans verticaux noirs détectés.

Seconde phase : Dans la seconde phase, les empans horizontaux susceptibles d'appartenir à un segment vertical sont extraits par convolution du négatif de l'image et intersection de l'image résultat avec l'image source.

Troisième phase : Dans la troisième phase, on réalise une fermeture verticale d'ordre 2. Afin d'éviter de connecter des objets qui doivent être effectivement bien séparés, il faut vérifier que les deux empans concernés sont de type ligne. La règle est la suivante : deux empans de la colonne y, séparés de 1 ou 2 pixels blancs, sont connectés si cela est pertinent.

Quatrième phase : Dans la quatrième phase, on recherche dans chaque colonne de la carte l'empan le plus long.

Nous obtenons donc la carte des empans suivante :

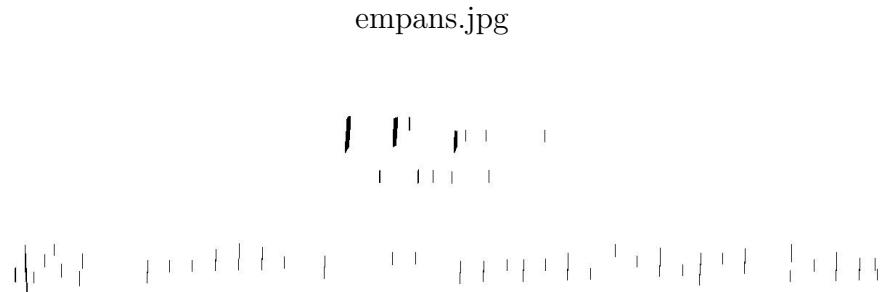


FIGURE 11 – Carte des empans phase quatre de la sous image 1

Nous pouvons remarquer qu'en haut de l'image nous n'avons pas un seul empan pour le titre mais bien un empan qui est trop large (on va traiter ce cas dans la cinquième phase). Néanmoins nous pouvons remarquer que nous reconnaissions bien les différents empans des notes et des barres de mesures.

Cinquième phase : La cinquième phase permet de retenir un empan unique par segment vertical. Pour cela, une fenêtre d'analyse de largeur adaptative parcourt horizontalement l'image

terminale.jpg

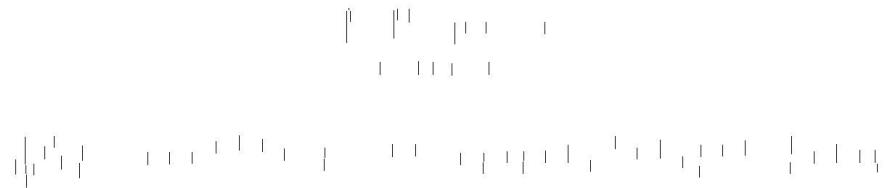


FIGURE 12 – Carte finale des empans de la sous image 1

Nous notons que nous avons une carte des empans performante (avec un empan à chaque fois) et que toutes les notes sont bien détectées.

3.6.3 Mise en place des boites englobantes

Mise en boite naïve L’empan principal de chaque segment ayant été précisément localisé, il peut servir de germe pour un algorithme de croissance de région, qui agglomère tous les pixels noirs connexes à cet empan, au sens des 8-voisins. On peut ainsi délimiter le symbole par une boîte englobante. Comme certains symboles sont encore connectés malgré l’effacement des lignes de portée, en particulier les notes reliées par des barres de groupe, on limite la croissance de part et d’autre de l’empan à 1.5 interligne. La largeur d’un symbole est en effet toujours inférieure à cette valeur, de chaque côté du segment vertical. L’autre condition d’arrêt de la croissance de région est l’absence de pixels noirs connexes à la région détectée.

Séparation des boites Les symboles isolés sont correctement délimités par la boîte englobante. En revanche, pour les groupes de notes, les limites de chaque rectangle dépendent du degré de proximité des notes. Lorsque deux notes consécutives sont très proches, alors les deux boîtes englobantes se chevauchent. Il faut donc ajouter une étape permettant de les séparer.

Suppression des boites Il reste certains cas où deux boîtes contiennent deux fois le même symbole. On supprime les doublons.

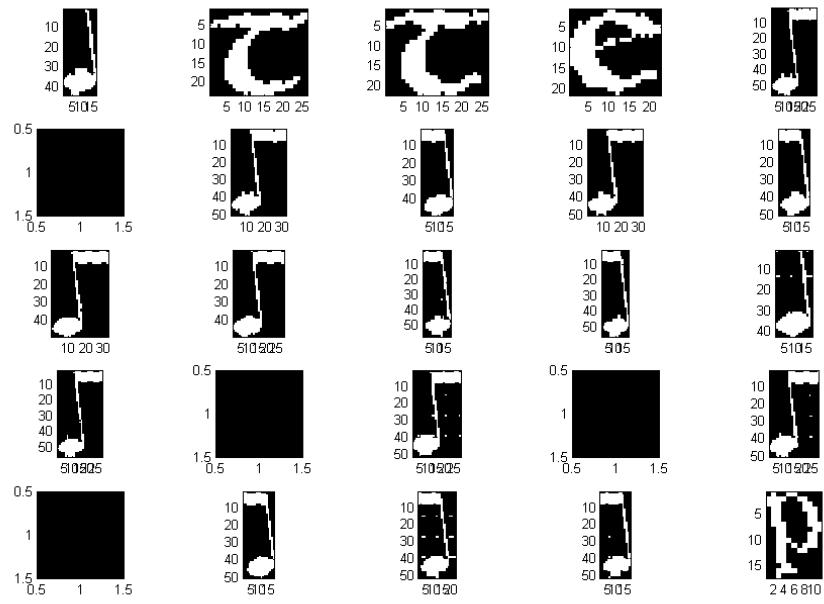


FIGURE 13 – Boites englobantes

On remarque que certaines boites sont vides ; cela ne posera pas de problèmes pour la suite de la détection car elles seront ignorées et supprimées en post-processing.

4 Analyse des boites englobantes

4.1 Reconnaissance des Symboles

4.1.1 Présentation des différentes Méthodes

Nous avons envisagé plusieurs méthodes pour la reconnaissance des notes. Il y a deux problèmes bien différents à distinguer : le problème de reconnaissance de la hauteur de la note et le problème de la reconnaissance du rythme.

Le problème de la reconnaissance de la note est assez simple à résoudre : en effet, il "suffit" de repérer un disque par template matching dans la boîte pour trouver la hauteur de la note.

Le second problème est bien plus complexe du fait de la forte variabilité dans la représentation d'une note. Prenons l'exemple de la croche, un des symboles de base pour la représentation musicale.



FIGURE 14 – Croche 1



FIGURE 15 – Croche 2

Il y a donc plusieurs syntaxes pour les différentes notes ; de plus, la syntaxe varie entre les différents éditeurs, il faut donc en tenir compte dans notre méthode de classification des rythmes.

Pour cela, on nous a proposé plusieurs méthodes, notamment :

1. L'utilisation de machine learning (utilisation de Réseaux de Neurones (en particulier les réseaux convolutifs)).
2. L'utilisation de Template Matching.

Nous avons pendant longtemps essayé de déployer un réseau de neurones . En effet ceux-ci possèdent beaucoup d'avantages et ils donnent de bons résultats même dans des domaines très complexes, ils sont plus performants que les statistiques ou les arbres de décisions pour découvrir les relations entre les variables. Ce système n'oblige pas à s'interroger sur la forme de la fonction à estimer ce qui est très pratique. Le réseau convolutif a une structure qui se rapproche de la structure de l'oeil et est très utile et rapide pour résoudre notre problème. Le problème étant bien sûr de l'entrainer sur une base de données.

Le principal problème étant que n'ayant pas assez de compétences en Python (les tutoriels sont en Python principalement pour les réseaux de neurones) et de connaissances sur les réseaux de neurones nous n'avons pas réussi à comprendre comment mettre en oeuvre les réseau du type caffe, pylearn2, theano ou tensorflow.

Nous nous sommes alors rabattus sur des critères géométriques plus simple pour définir les rythmes, que l'on reconnaîtra à l'aide de Template Matching.

4.1.2 Reconnaissance de notes

Cette fonction est centrale dans notre code. Elle s'occupe de discriminer les notes afin de savoir quelle est la hauteur de la note, s'il s'agit bien d'une note ou d'autre chose (barre de mesure, dièse, bémol, clef de sol etc..).

Explication de la démarche à l'aide du code

On propose d'expliquer la méthode en commentant le code de la fonction.
On commence dans le code par inclure les librairies nécessaires à son exécution :

```
#include<reconnaissancenotes.h>
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/gpu/gpu.hpp"
#include "opencv2/highgui/highgui.hpp"
```

On déclare ensuite le type de la fonction :

```
trioIntString reconnaissancenotes(Eigen::MatrixXi img ,
int premiere_ligne , int interligne , int xb , int yb ,
int xh , int yd)
```

Nous prenons en compte le cas où le template du disque que l'on va définir est plus grand (en largeur ou en hauteur) que l'imagette (autrement dit ce l'imagette n'est pas une note) :

```
if(img.row>interligne && img.col>interligne)
```

Ensuite nous définissons le template qui sera un disque de rayon interligne/2 :

```
int milieu;
Eigen::MatrixXi templ;
if (interligne&1)
{
    templ.create(interligne, interligne, CV_32IC1);
    milieu = (interligne - 1)/2;
}
else
{
    templ.create(interligne+1, interligne+1, CV_32IC1);
    milieu = (interligne)/2+1;
}
```

Nous remplissons le template :

```
int i;
int j;
for (i=0; i<templ.rows; i++)
{
    for (j=0; j<templ.cols; j++)
    {
        if ((i-milieu)*(i-milieu)+(j-milieu)*(j-milieu)<interligne*interligne)
        {
            templ.at<int>(i, j)= (1);
        }
        else
        {
            templ.at<int>(i, j)= (0);
        }
    }
}
```

On définit ensuite la matrice de résultat (celle qui contiendra l'ensemble des résultats du template matching)

```
int resultcols = abs(img.cols() - templ.cols()) + 1;
int resultrows = abs(img.rows() - templ.rows()) + 1;
Eigen::MatrixXi toto(resultrows, resultcols);
cv::Mat result(toto.rows(), toto.cols(), CV_32FC1,
toto.data());
```

On définit ensuite la méthode que l'on compte utiliser : les résultats obtenus dans la matrice de résultat seront ceux de l'intercorrélation des deux images. Nous expliquerons plus tard les raisons de ce choix.

```
int match_method = CV_TM_CCORR;
```

Nous faisons ensuite le template matching et nous localisons le maximum d'intercorrélation afin de repérer où se situe la partie ronde de la note.

```
cv::matchTemplate( img, templ, result, match_method );
double minVal;
double maxVal;
cv::Point minLoc;
cv::Point maxLoc;
cv::Point matchLoc;

cv::minMaxLoc( result, &minVal, &maxVal, &minLoc,
&maxLoc, cv::Mat() );
{
matchLoc = maxLoc;
}
```

Nous plaçons ensuite la note par rapport aux lignes de portée en calculant sa position par rapport à la troisième ligne de la portée. Nous pavons complètement la portée, à chaque hauteur de note est attribuée une hauteur centrale, et des hauteurs de "seuil", que l'on prend égales à *interligne*/4 en bas et en haut de cette hauteur centrale. Cela donne effectivement un pavage de la portée.
Note : le type trioIntString est une structure contenant deux int (.a,.b) et un string (.c).

```

int milieu_de_la_note=matchLoc.x+templ.cols/2+xb;
float seuil = interligne/8;
float pas= interligne/2;
int epaisseur = floor(intervalle/5);
if(maxVal > epaisseur*3*(intervalle-epaisseur))
{
    trioIntString retour;
    if ( premiere_ligne+pas+seuil>milieu_de_la_note
&&milieu_de_la_note>premiere_ligne+pas-seuil)
    {
        std::cout << "Re" << std::endl;
        retour.c="Re";
    }
    else if ( premiere_ligne+seuil>milieu_de_la_note
&&milieu_de_la_note>premiere_ligne-seuil)
    {
        std::cout << "mi" << std::endl;
        retour.c="mi";
    }
    else if ( premiere_ligne-pas+seuil>milieu_de_la_note
&&milieu_de_la_note>premiere_ligne-pas-seuil)
    {
        std::cout << "fa" << std::endl;
        retour.c="fa";
    }
    else if ( premiere_ligne-2*pas+seuil>milieu_de_la_note
&&milieu_de_la_note>premiere_ligne-2*pas-seuil)
    {
        std::cout << "sol" << std::endl;
        retour.c="sol";
    }
    else if ( premiere_ligne-3*pas+seuil>milieu_de_la_note
&&milieu_de_la_note>premiere_ligne-3*pas-seuil)
    {
        std::cout << "la" << std::endl;
        retour.c="la";
    }
    else if ( premiere_ligne-4*pas+seuil>milieu_de_la_note
&&milieu_de_la_note>premiere_ligne-4*pas-seuil)
    {
        std::cout << "si" << std::endl;
    }
}

```

```

    retour .c=" si ";
}
else if ( premiere_ligne -5*pas+seuil > milieu_de_la_note
&& milieu_de_la_note > premiere_ligne -5*pas-seuil )
{
std :: cout << "do" << std :: endl ;
retour .c="do";
}
else if ( premiere_ligne -6*pas+seuil > milieu_de_la_note
&& milieu_de_la_note > premiere_ligne -6*pas-seuil )
{
std :: cout << "Re_haut" << std :: endl ;
retour .c="Re_haut";
}
else if ( premiere_ligne -7*pas+seuil > milieu_de_la_note
&& milieu_de_la_note > premiere_ligne -7*pas-seuil )
{
std :: cout << "mi_haut" << std :: endl ;
retour .c="mi_haut";
}
else if ( premiere_ligne -8*pas+seuil > milieu_de_la_note
&& milieu_de_la_note > premiere_ligne -8*pas-seuil )
{
std :: cout << "fa_haut" << std :: endl ;
retour .c="fa_haut";
}
else if ( premiere_ligne -9*pas+seuil > milieu_de_la_note
&& milieu_de_la_note > premiere_ligne -9*pas-seuil )
{
std :: cout << "sol_haut" << std :: endl ;
retour .c="sol_haut";
}
else
{
std :: cout << " Il_ya_une_erreur " << std :: endl ;
}

```

Il faut ensuite regarder, dans le cas où on a trouvé une note noire, si celle-ci est une noire, une croche, une double-croche, ou plus. On fait ceci en appelant une fonction d'identification nommée reconnaissanceCroche. Elle fonctionne de la manière suivante :

1. On efface le rond de la note.

2. On effectue une recherche de rectangle de largeur celle de la note sur deux, de hauteur l'interligne divisé par trois.
3. On efface ce rectangle et ses prolongements éventuels.
4. On recommence tant que la valeur du maximum de l'autocorrélation est supérieure à $W/2 * interligne/4$
5. Si aucune barre de croche n'a été trouvée, on cherche par croissance de région le point le plus à l'extrême horizontalement de notre note. Si cette distance est de l'ordre de l'interligne/2, c'est une croche.
6. Sinon, c'est une noire.

```

int W = img . cols () ;
int H = img . rows () ;
int Hp= interligne /4;
cv :: Mat templ , imag , result ;
templ . create (Hp , W/2 , CV_32FC1) ;
imag . create (H , W , CV_32FC1) ;
for (int i = 0 ; i < Hp ; i++){
    for (int j = 0 ; j < W/2 ; j++){
        templ . at<float>(i , j) = 1;
    }
} for (int i = 0 ; i < H ; i++){
    for (int j = 0 ; j < W ; j++){
        imag . at<float>(i , j) = (float) img(i , j);
    }
}
int resultcols = img . cols () - templ . cols + 1;
int resultrows = img . rows () - templ . rows + 1;
double minVal , maxVal ;
cv :: Point minLoc ;
cv :: Point maxLoc ;

result . create (resultrows , resultcols , CV_32FC1) ;

cv :: matchTemplate(imag , templ , result , CV_TMCCORR) ;
cv :: minMaxLoc(result , &minVal , &maxVal , &minLoc , &
    maxLoc , cv :: Mat()) ;
// La croche est-elle simple , double , ... ?
int nombreCroche = 0;
while (maxVal > (interligne *W) /4){
    nombreCroche++;
}

```

```

for(int i = maxLoc.x ; i < maxLoc.x+Hp && i < H ;
    i++) {
    for(int j = maxLoc.y ; j < maxLoc.y+W/2 && j <
        W ; j++) {
        imag .at<float>(i , j) = 0;
    }
}
cv :: matchTemplate(imag , templ , result , CV_TM_CCORR
);
cv :: minMaxLoc(result , &minVal , &maxVal , &minLoc , &
    maxLoc , cv :: Mat());
}
interm . symbole = "croche" + nombreCroche;
return interm;
}

```

On retourne ensuite les différentes valeurs.

```

std :: cout << "Fin_de_la_reconnaissance_de_la_note_" << std :: endl ;
int abscissedurond = matchLoc.y + yb ;
std :: cout << "c'est_une_note" << std :: endl ;

std :: cout << "matchloc.x" << abscissedurond << std :: endl ;
std :: cout << "matchloc.x" << milieu_de_la_note << std :: endl ;

retour . a=abscissedurond ;
retour . b=milieu_de_la_note ;

return retour ;

```

Si le résultat du template matching est inférieur à $epaisseur * 3 * (interligne - epaisseur)$ (seuil que nous expliquerons par la suite) nous choisissons de dire qu'il ne s'agit pas d'une note mais bien d'un dièse, d'un bémol, d'une barre de mesure ou bien d'une clef de sol.

On identifie les barres de mesure grâce à un profil horizontal : si le maximum du profil est inférieur à deux fois l'épaisseur moyenne des lignes de portée, on dit que c'est une barre de mesure.

```

else
{
    std :: cout << " ceci_n'est_pas_une_note" << std :: endl;
    Eigen :: MatrixXi somme = Eigen :: MatrixXi :: Zero ( img . rows () , 1 );
    for ( int i=0; i<img . rows () ; i++)
    {
        for ( int j=0; j<img . cols () ; j++)
        {
            somme ( i , 0 ) = img ( i , j ) + somme ( i , 0 );
        }
    }
    std :: cout << img << std :: endl ;
    std :: cout << " _somme_____ " << somme << std :: endl ;
    trioIntString retour;
    if ( somme . maxCoeff () < 2 * epaisseur )
    {
        std :: cout << " ceci_est_une_barre_de_mesure" << std :: endl ;
        retour . c = " barre_de_mesure" ;
    }
    else
    {
        std :: cout << " ceci_est_autre_chose" << std :: endl ;
    }
    retour . a = -1;
    retour . b = -1;
    return retour ;
}
}

```

Justification de la méthode de template matching utilisée

Opencv propose les méthodes de template matching suivantes (I la matrice représentant l'image et T la matrice du template) :

— $method = CV_TMSQDIFF$

$$R(x, y) = \sum_{x', y'} (T(x', y') - I(x + x', y + y'))^2$$

— $method = CV_TMSQDIFF_NORMED$

$$R(x, y) = \frac{\sum_{x', y'} (T(x', y') - I(x + x', y + y'))^2}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}}$$

- $method = CV_T M_C CORR$
 $R(x, y) = \sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))$
- $method = CV_T M_C CORR_NORMED$
 $0 R(x, y) = \frac{\sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}}$
- $method = CV_T M_C COEFF$
 $R(x, y) = \sum_{x', y'} (T'(x', y') \cdot I(x + x', y + y'))$
avec $T'(x', y') = T(x', y') - 1/(w \cdot h) \cdot \sum_{x'', y''} T(x'', y'')$
 $I'(x + x', y + y') = I(x + x', y + y') - 1/(w \cdot h) \cdot \sum_{x'', y''} I(x + x'', y + y'')$

- $method = CV_T M_C COEFF_NORMED$

$$R(x, y) = \frac{\sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x', y'} T'(x', y')^2 \cdot \sum_{x', y'} I'(x + x', y + y')^2}}$$

La fonction que nous avons utilisée est celle qui donne les meilleurs résultats dans la reconnaissance d'ellipses.

Justification des seuils de détection

Une intercorrélation consiste en fait à calculer l'aire de deux figures superposées, en les décalant progressivement l'une par rapport à l'autre. Dès lors, l'intercorrélation maximale entre notre template rond et une note noire est, en supposant que notre template est plus petit que le rond de la noire, l'aire du cercle :

$$A = \pi * (interligne)^2$$

De plus, si on a une note blanche, en supposant que le cercle qui forme la blanche est à peu près de l'épaisseur des lignes de portée qu'on assimile à $interligne/10$, l'aire sera toujours supérieure à

$$A = 3 * ((interligne)^2 - (interligne - epaisseur)^2) \text{ C'est à dire } A = 3 * epaisseur(epaisseur - interligne)$$

On peut donc très facilement trouver des seuils pour la détection des notes.

5 Améliorations et Optimisations

Tout code mérite d'être optimisé ; et le nôtre ne fait pas exception. La partie la plus lente de notre algorithme est, et de loin, la partie de redressement de l'image, qui consiste à un calcul d'intercorrélation entre deux parties de l'image. Une première idée fut d'implémenter du Multithreading afin de dispatcher les différentes parties du calcul entre différents centres de calcul disponibles à la machine utilisée.

5.1 Multithreading

On utilise pour implémenter le multithreading la librairie pthread. Chaque Thread possède un numéro propre, qui permet de dispatcher d'une manière efficace les calculs, ainsi qu'un pointeur qui permet de transporter avec lui dans la fonction les données nécessaires au calcul. Il faut bien entendu utiliser des calculs modulaires afin d'éviter un débordement et des calculs redondants.

```
void calculThread(void fun){
    threadData thr = (threadData*) fun;
    int nthreads = thr.h;
    int n = thr.xm;
    Eigen::MatrixXi image = thr->image;
    int H = image.rows();
    int W = image.cols(nthreads);
    int indHmax = 0;
    int valHmax = H;
    int a = 0;
    int b = 0;
    for(int h = n ; h < H+W ; h = h+nthreads*nthreads){
        a = 0;
        b = 0;
        for(int x = 0 ; (x+h) < H ; x++){
            for(int y = 0 ; y <= W/2-1 ; y++){
                a = a + image(x,y)*image(x+h, y+W/2);
            }
        }
        for(int x = h ; x < H ; x++){
            for(int y = 0 ; y <= W/2-1 ; y++){
                b = b + image(x,y)*image(x-h, y+W/2);
            }
        }
    }
    if(a > valHmax){
        indHMax = a;
```

```

    } if (b > valHmax) {
        valHmax = b;
    }
}
thr->h = indHmax;
thr->xm = valHmax;
pthread_exit(NULL);
}

```

5.2 Programmation GPU

Afin d'accélérer encore les calculs de l'autocorrélation, nous nous sommes proposés de transmettre les calculs aux cellules de calcul de Shader des cartes graphiques. Celles-ci en possédant de très nombreuses, les calculs se retrouvent très grandement accélérés. Malheureusement, par manque de logiciel de programmation graphique adapté à toutes les cartes (NVidia ou ATI, des cartes graphiques qui ne sont pas usuellement utilisées dans les smartphones actuels) et à cause à la fois de la difficulté de ce genre de programmation et de leurs débugage extrêmement difficile, nous n'avons pas pu terminer cette implémentation et nous n'avons pas aboutis à un algorithme fonctionnel.

5.3 Amélioration pour la Robustesse

Après quelques tests, nous nous rendîmes compte que les partitions trop "parfaites" posaient des problèmes à nos algorithmes de calcul : effectivement, chaque calcul utilisant le Profil Vertical Py et cherchant des suites de maximum rendaient des problèmes. Dans la Recherche des Xis (maximums locaux correspondant aux lignes de portées), nous avions mis des inégalités strictes afin de sélectionner une des lignes de portée à partir des maximums de Py, mais ces maximums étaient égaux pour chaque portion de la portée ! Il ne faut donc en sélectionner qu'un seul ; nous prenons le premier.

De même, pour la détection des interlignes, l'autocorrélation donnait un résultat maximal égal à 1 parce que les portées étaient bien trop parfaites : on a donc dû modifier l'indice de départ afin de dépasser la première ligne : on dépasse la première ligne, puis cherche le maximum d'autocorrélation.

5.4 Développement de l'interface pour la présentation des résultats

5.4.1 L'interface console

Nous avons mis en place dans un soucis d'ergonomie une interface avec la console afin de nous permettre d'accéder plus librement aux fichiers et de mieux présenter nos résultats.

Cette interface d'un design simple, épurée et ergonomique a été codée grâce à la librairie dirent version 1.21 disponible sur le net, et qui permet à l'utilisateur de lire et connaître les fichiers présents dans un répertoire.

Notre interface possède les propriétées suivantes :

- Cette interface permet d'ouvrir directement un fichier à partir de son adresse et de lancer le programme d'interprétation de partitions.
- Cette interface permet de lire et d'afficher de manière dynamique tous les fichiers dans le dossier test et de lancer le programme sur une image.
- Cette interface permet de quitter le programme.
- Cette interface permet de quitter le programme.

5.4.2 L'interface graphique

Par manque de temps nous n'avons pas pu faire aboutir cette partie du projet. Nous avons néanmoins eu le temps de créer une ébauche de l'interface graphique :

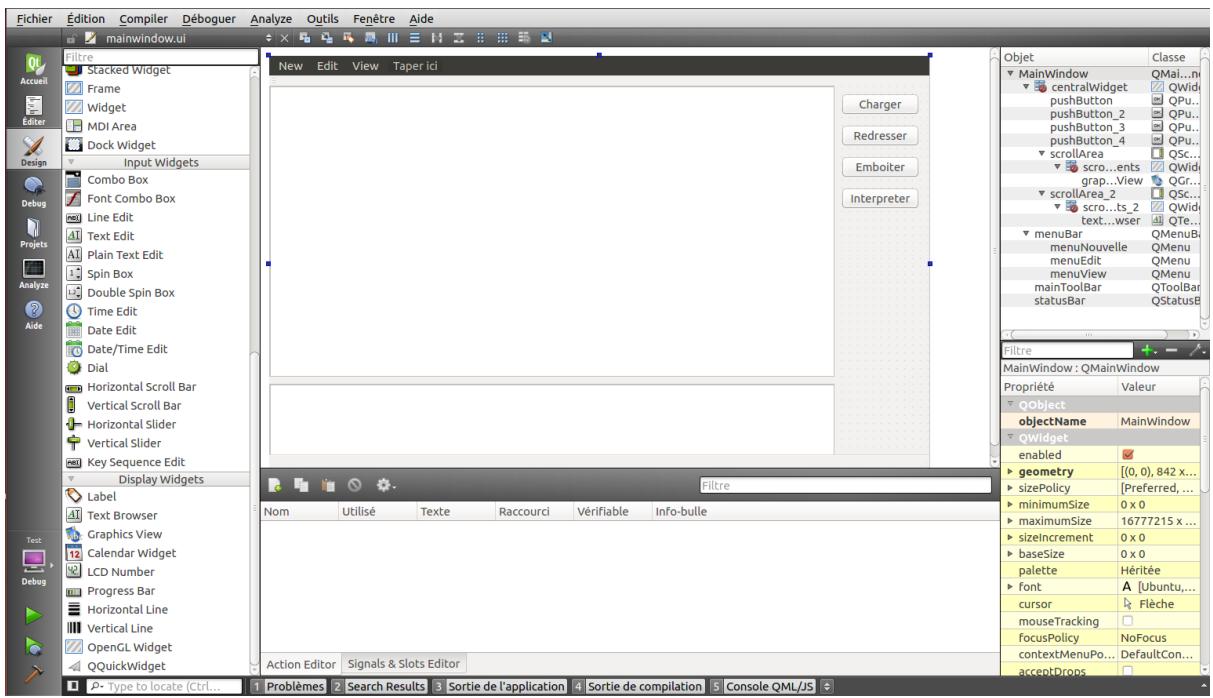


FIGURE 16 – Ébauche de l’Interface Graphique - QtDesigner

5.5 Conclusions

Nous avons donc présenté une méthode de segmentation qui permet de détecter les symboles caractérisés par un segment vertical, de les délimiter par une boîte englobante, et de créer une image qui contient tous les autres symboles, en particulier les silences, les points et les rondes.

La détection des segments verticaux a été réalisée avec un soin tout particulier, afin de garantir robustesse et précision de l'extraction.

6 Choix du C++

Une fois toutes ces fonctions modélisées en Matlab, et, ayant vérifié qu'elles sont effectivement fonctionnelles, nous réalisâmes que nous ne pourrions pas utiliser ce code dans notre application : il est pour l'instant impossible d'utiliser du code Matlab sur un smartphone standard. Il fallait changer de langage, en gardant à l'esprit que nous nous devions d'optimiser autant que possible notre programme, à la fois en vitesse et en utilisation mémoire. Une première idée fut d'utiliser Java ; idée que nous mêmes de côté pour plusieurs raisons :

1. Une gestion mémoire non-controlable.
2. Une vitesse d'exécution relativement lente.
3. Un manque de librairies par rapport à d'autres langages (C++, Python)
4. Une connaissance déjà approfondie de ce langage : nous n'aurions rien appris.

Nous nous portâmes alors sur le C++ pour les raisons suivantes :

1. Une vitesse d'exécution en moyenne très rapide devant beaucoup d'autres langages (Python et Java)
2. Une myriade de librairies C++ libres à notre disposition.
3. Des capacités et une finesse de contrôle mémoire extrêmement importantes.
4. Un manque de connaissances de ce langage pourtant grandement utilisé.
5. Une possibilité d'utilisation des codes C++ sur Android.
6. La possibilité d'utiliser les Cartes Graphiques des smartphones pour accélérer les processus de calcul.

7 Test

Dans cette partie nous proposons de tester nos algorithmes décris précédemment avec différentes partitions simples et plus complexes.

7.1 Le problème de la clef de sol et de l'armure

On se propose de mettre en évidence le problème de la clef de sol et de l'armure ici. Nous utilisons la partition suivante :

Mumbo jumbo

Nicolas Fleurier

Copyright © 2005 Nicolas Fleurier.

FIGURE 17 – Test : Mise en évidence du problème de la clef

Nous recadrons la partition pour enlever le titre qui pollue notre détection et nous isolons la clef de sol et l'armure.

FIGURE 18 – Sous image du Test : Mise en évidence du problème de la clef

Nous obtenons le fichier test suivant :

Note : Re - Rythme : crochenumero de portee 1

Note : La - Rythme : crochenumero de portee 1

Note : Do diese - Rythme : numero de portee 1

Note : Do diese - Rythme : barre de mesurenumero de portee 1

Note : Do diese - Rythme : barre de mesurenumero de portee 1

 Note : Do - Rythme : crochenumero de portee 1

 Note : - Rythme : crochenumero de portee 1

 Note : La - Rythme : crochenumero de portee 1

 Note : Fa - Rythme : crochenumero de portee 1

Note : Do diese - Rythme : numero de portee 1

 Note : Si - Rythme : crochenumero de portee 1

 Note : La - Rythme : crochenumero de portee 1

 Note : Fa - Rythme : crochenumero de portee 1

Note : Do diese - Rythme : numero de portee 1

 Note : Si - Rythme : crochenumero de portee 1

Les limites Nous mettons en évidence à l'aide de ces test plusieurs défauts de notre programme :

- le problème de la clef de sol
- le problème de l'armure
- le problèmes des altérations

En effet la clef de sol n'est pas traitée séparément par notre programme. Donc les empans verticaux de la clef sont détectés comme des notes potentielles.

De même pour l'armure et les altérations elles ne sont pas traitées de façon indépendantes par notre programme elles sont donc mal détectées.

7.2 La partition : Le caillou

Nous choisissons de faire un premier test avec la partition suivante :

Le Caillou

Jo Akepsimas / Maurice Carême

Nom du copiste

The musical score consists of two staves of music in common time (indicated by a 'C') with a key signature of one sharp (F#). The top staff is labeled 'Chant' and has dynamics 'f' (fortissimo) and 'Fine'. The lyrics for this staff are: 'Un petit caillou rond Rit sur le sable blond.' The bottom staff has dynamics 'mf' (mezzo-forte) and 'Fine'. The lyrics for this staff are: 'Il rit d'être aussi rond Qu'un banc de sable blond.'

FIGURE 19 – Partition de test : Le Caillou

7.3 Test sur une sous-portée

Dans un premier temps nous regardons ce que nous donne le fichier test sous une sous-image découpée manuellement.

A manually cut-out rectangular image from the original musical score, showing a portion of the first staff. It includes the lyrics 'Un petit caillou rond Rit sur le sable blond.' and ends with a 'Fine' dynamic. The image is set against a white background.

FIGURE 20 – Sous image de la partition de test : Le Caillou

Nous obtenons donc en sortie le fichier test suivant :

```
Note : Si – Rythme : noire
Note : Sol – Rythme : croche
Note : Si – Rythme : croche
Note : La – Rythme : croche
Note : Sol – Rythme : croche
Note : Do dièse – Rythme : barre de mesure
Note : La – Rythme : croche
```

Note : Do diese – Rythme : barre de mesure
Note : La – Rythme : croche
Note : Fa – Rythme : croche
Note : La – Rythme : croche
Note : Sol – Rythme : croche2
Note : Fa – Rythme : croche
Note : Do diese – Rythme :
Note : Sol – Rythme : croche
Note : Do diese – Rythme : barre de mesure
Note : Do diese – Rythme : barre de mesure

Analysons les résultats : la hauteur des notes est bien juste. Ensuite nous détectons bien les barres de mesure avec pour note do dièse.

Analysons ensuite le rythme. Les noirs sont bien détectées et les croches aussi. Le rythme est bon sauf pour les blanches pointées qui sont non pas comptées comme des blanches mais comme des croches.

7.4 Test sur toute la partition : Le Caillou

7.4.1 Les résultats

Nous testons la partition "le Caillou" entièrement. Nous obtenons le fichier texte suivant :

Note : Do diese - Rythme : barre de mesurenuméro de portée 1

Note : Re - Rythme : crochenuméro de portée 1

Note : Do diese - Rythme : barre de mesurenuméro de portée 1

Note : Do diese - Rythme : numéro de portée 1

Note : La - Rythme : crochenuméro de portée 1

Note : Mi - Rythme : crochenuméro de portée 1

Note : Si - Rythme : noirenuméro de portée 1

Note : Sol - Rythme : crochenuméro de portée 1

Note : Si - Rythme : crochenuméro de portée 1

Note : La - Rythme : crochenuméro de portée 1

Note : - Rythme : crochenuméro de portée 1

Note : Sol - Rythme : crochenuméro de portée 1

Note : - Rythme : croche2numéro de portée 1

Note : - Rythme : crochenuméro de portée 1

Note : Do diese - Rythme : barre de mesurenuméro de portée 1

Note : - Rythme : croche3numéro de portée 1

Note : La - Rythme : crochenuméro de portée 1

Note : - Rythme : croche3numéro de portée 1
Note : Do diese - Rythme : numéro de portée 1
Note : Do diese - Rythme : numéro de portée 1
Note : Do diese - Rythme : numéro de portée 1
Note : Do diese - Rythme : numéro de portée 1
Note : La - Rythme : crochenuméro de portée 1
Note : - Rythme : croche3numéro de portée 1
Note : - Rythme : croche2numéro de portée 1
Note : - Rythme : crochenuméro de portée 1
Note : - Rythme : crochenuméro de portée 1
Note : Fa - Rythme : crochenuméro de portée 1
Note : La - Rythme : crochenuméro de portée 1
Note : Sol - Rythme : croche2numéro de portée 1
Note : Fa - Rythme : crochenuméro de portée 1
Note : Do diese - Rythme : barre de mesurenuméro de portée 1
Note : Sol - Rythme : crochenuméro de portée 1
Note : Do diese - Rythme : barre de mesurenuméro de portée 1
Note : Do diese - Rythme : barre de mesurenuméro de portée 1
Note : Do diese - Rythme : barre de mesurenuméro de portée 2
 Note : Re - Rythme : crochenuméro de portée 2
Note : Do diese - Rythme : barre de mesurenuméro de portée 2
 Note : Do diese - Rythme : numéro de portée 2
 Note : - Rythme : crochenuméro de portée 2
 Note : - Rythme : crochenuméro de portée 2
Note : Re - Rythme : noirenuméro de portée 2
Note : Si - Rythme : crochenuméro de portée 2
Note : Re - Rythme : crochenuméro de portée 2
Note : Do - Rythme : croche2numéro de portée 2
Note : Si - Rythme : crochenuméro de portée 2
Note : Do diese - Rythme : barre de mesurenuméro de portée 2
Note : Do - Rythme : noirenuméro de portée 2
Note : Do diese - Rythme : barre de mesurenuméro de portée 2
 Note : Do - Rythme : noirenuméro de portée 2
 Note : La - Rythme : crochenuméro de portée 2
 Note : Do - Rythme : crochenuméro de portée 2
 Note : Si - Rythme : croche2numéro de portée 2
 Note : La - Rythme : crochenuméro de portée 2
Note : Do diese - Rythme : barre de mesurenuméro de portée 2
Note : Si - Rythme : noirenuméro de portée 2
Note : Do diese - Rythme : barre de mesurenuméro de portée 2
 Note : Do diese - Rythme : numéro de portée 2

7.4.2 Analyse des résultats

Les résultats sont conformes aux attentes. Les notes sont bien analysées (la hauteur des notes est bonne). Le rythme a les mêmes problèmes que précédemment avec les blanches pointées.

Ici nous voyons bien les limites de notre programme. Toutes les notes où il n'y a pas de notes détectées ; il s'agit de la détection du titre : Le Caillou (croche 2 veut dire double croche et croche 3 triple croche).

L'autre problème consiste en la détection des clefs de sol non traitées ici. Il y a plusieurs fausses détections sur le début des portées.

7.5 Analyse de l'image test

Nous retournons à l'image "Je regrette". Nous proposons de faire des test sur l'image suivante :

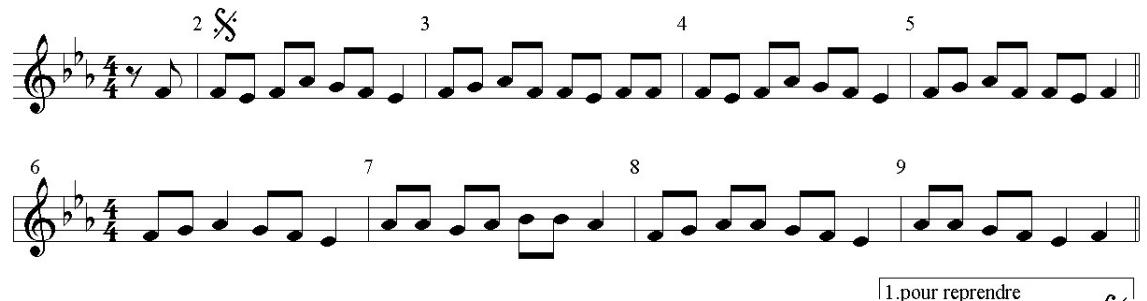


FIGURE 21 – Partition de test : "Je regrette"

Nous obtenons en sortie le fichier txt suivant :

```
Note : - Rythme : crochenumero de portee 1
Note : Do diese - Rythme : numero de portee 1
Note : Do diese - Rythme : numero de portee 1
Note : Do diese - Rythme : numero de portee 1
Note : Re - Rythme : crochenumero de portee 1
Note : Sol - Rythme : crochenumero de portee 1
Note : Fa - Rythme : crochenumero de portee 1
Note : Fa - Rythme : crochenumero de portee 1
Note : Mi - Rythme : crochenumero de portee 1
Note : Fa - Rythme : crochenumero de portee 1
Note : La - Rythme : crochenumero de portee 1
Note : Sol - Rythme : crochenumero de portee 1
Note : Fa - Rythme : crochenumero de portee 1
Note : Mi - Rythme : noirenumero de portee 1
```

Note : Fa – Rythme : crochenumero de portee 1
Note : Sol – Rythme : crochenumero de portee 1
Note : La – Rythme : crochenumero de portee 1
Note : Fa – Rythme : crochenumero de portee 1
Note : Fa – Rythme : crochenumero de portee 1
Note : Mi – Rythme : crochenumero de portee 1
Note : Fa – Rythme : crochenumero de portee 1
Note : Fa – Rythme : crochenumero de portee 1
Note : Mi – Rythme : crochenumero de portee 1
Note : Fa – Rythme : crochenumero de portee 1
Note : La – Rythme : crochenumero de portee 1
Note : Sol – Rythme : crochenumero de portee 1
Note : Fa – Rythme : crochenumero de portee 1
Note : Mi – Rythme : noirenumero de portee 1
Note : Fa – Rythme : crochenumero de portee 1
Note : Sol – Rythme : crochenumero de portee 1
Note : La – Rythme : crochenumero de portee 1
Note : Fa – Rythme : crochenumero de portee 1
Note : Fa – Rythme : crochenumero de portee 1
Note : Mi – Rythme : crochenumero de portee 1
Note : Fa – Rythme : noirenumero de portee 1
Note : – Rythme : crochenumero de portee 2
Note : Do diese – Rythme : numero de portee 2
Note : Do diese – Rythme : numero de portee 2
Note : Do diese – Rythme : numero de portee 2
Note : Re – Rythme : crochenumero de portee 2
Note : Sol – Rythme : crochenumero de portee 2
Note : Fa – Rythme : crochenumero de portee 2
Note : Sol – Rythme : crochenumero de portee 2
Note : La – Rythme : noirenumero de portee 2
Note : Sol – Rythme : crochenumero de portee 2
Note : Fa – Rythme : crochenumero de portee 2
Note : Mi – Rythme : noirenumero de portee 2
Note : La – Rythme : crochenumero de portee 2
Note : La – Rythme : crochenumero de portee 2
Note : Sol – Rythme : crochenumero de portee 2
Note : La – Rythme : crochenumero de portee 2
Note : Si – Rythme : crochenumero de portee 2
Note : Si – Rythme : crochenumero de portee 2
Note : La – Rythme : noirenumero de portee 2
Note : Fa – Rythme : crochenumero de portee 2
Note : Sol – Rythme : crochenumero de portee 2

```

Note : La – Rythme : crochenumero de portee 2
Note : La – Rythme : crochenumero de portee 2
Note : Sol – Rythme : crochenumero de portee 2
Note : Fa – Rythme : crochenumero de portee 2
Note : Mi – Rythme : noirenumero de portee 2
Note : – Rythme : numero de portee 2
Note : La – Rythme : crochenumero de portee 2
Note : La – Rythme : crochenumero de portee 2
Note : Sol – Rythme : crochenumero de portee 2
Note : Fa – Rythme : crochenumero de portee 2
Note : Mi – Rythme : noirenumero de portee 2
Note : Fa – Rythme : noirenumero de portee 2
Note : – Rythme : numero de portee 2

```

Analyse Nous obtenons bien les bonnes notes avec les bons rythmes modulo le problème du début de portée avec l'armure et la clef qui reste mal détectée. Les tests sont concluant : malgré de nombreuses limitations, l'algorithme permet d'analyser des partitions musicales assez simples.

7.6 Analyse à l'aide de Matrices de Confusion

Nous avons collecté les données d'utilisation de notre algorithme - notes bien déterminées, formes bien déterminées, et avons mis des données sous forme de matrices de confusion afin de présenter les résultats de notre algorithme d'une manière ordonnée. Chaque construction a été réalisée sur une partition différente. Voici les différentes matrices obtenues.

7.6.1 Matrice de confusion de "Je Regrette"



FIGURE 22 – Partition de test : Le Caillou

Nous obtenons en sortie le fichier txt suivant :

```

Note : Do diese – Rythme : barre de mesure numero de portee
0
Note : Re – Rythme : croche numero de portee 0

```

Note : Do diese – Rythme : barre de mesure numero de portee 0
Note : Do diese – Rythme : numero de portee 0
Note : – Rythme : croche numero de portee 0
Note : – Rythme : croche numero de portee 0
Note : Re – Rythme : noire numero de portee 0
Note : Si – Rythme : croche numero de portee 0
Note : Re – Rythme : croche numero de portee 0
Note : Do – Rythme : croche2 numero de portee 0
Note : Si – Rythme : croche numero de portee 0
Note : Do diese – Rythme : barre de mesure numero de portee 0
Note : Do – Rythme : noire numero de portee 0
Note : Do diese – Rythme : barre de mesure numero de portee 0
Note : Do – Rythme : noire numero de portee 0
Note : La – Rythme : croche numero de portee 0
Note : Do – Rythme : croche numero de portee 0
Note : Si – Rythme : croche2 numero de portee 0
Note : La – Rythme : croche numero de portee 0
Note : Do diese – Rythme : barre de mesure numero de portee 0
Note : Si – Rythme : noire numero de portee 0
Note : Do diese – Rythme : barre de mesure numero de portee 0
Note : Do diese – Rythme : numero de portee 0

Ce fichier nous permet d'écrire la matrice de confusion suivante :

	Noire	Croche	Blanche	Barre Mesure	Clef de Sol
Noire	2	0	1	0	0
Croche	0	8	1	0	0
Blanche	0	0	0	0	0
Barre Mesure	0	0	0	6	1
Clef de Sol	0	0	0	0	0
Efficacite	100%	100%	0%	100%	0%

TABLE 1 – Matrice de Confusion générée à partir de la partition "Je Regrette"

7.6.2 Matrice de confusion "Le Caillou"



FIGURE 23 – Partition de test : Le Caillou

Nous obtenons en sortie le fichier texte suivant :

```
Note : — Rythme : crochenumero de portee 0
Note : Do diese — Rythme : numero de portee 0
Note : Do diese — Rythme : numero de portee 0
Note : Do diese — Rythme : numero de portee 0
Note : Re — Rythme : crochenumero de portee 0
Note : Sol — Rythme : crochenumero de portee 0
Note : Fa — Rythme : crochenumero de portee 0
Note : Sol — Rythme : crochenumero de portee 0
Note : La — Rythme : noirenumero de portee 0
Note : Sol — Rythme : crochenumero de portee 0
Note : Fa — Rythme : crochenumero de portee 0
Note : Mi — Rythme : noirenumero de portee 0
Note : La — Rythme : crochenumero de portee 0
Note : La — Rythme : crochenumero de portee 0
Note : Sol — Rythme : crochenumero de portee 0
Note : La — Rythme : crochenumero de portee 0
Note : Si — Rythme : crochenumero de portee 0
Note : Si — Rythme : crochenumero de portee 0
Note : La — Rythme : noirenumero de portee 0
Note : Fa — Rythme : crochenumero de portee 0
Note : Sol — Rythme : crochenumero de portee 0
Note : La — Rythme : crochenumero de portee 0
Note : La — Rythme : crochenumero de portee 0
Note : Sol — Rythme : crochenumero de portee 0
Note : Fa — Rythme : crochenumero de portee 0
Note : Mi — Rythme : noirenumero de portee 0
Note : La — Rythme : crochenumero de portee 0
Note : La — Rythme : crochenumero de portee 0
Note : Sol — Rythme : crochenumero de portee 0
Note : Fa — Rythme : crochenumero de portee 0
```

Note : Mi	Rythme : noirenumero de portee 0
Note : Fa	Rythme : noirenumero de portee 0
Note : La	Rythme : crochenumero de portee 2
Note : La	Rythme : crochenumero de portee 2
Note : Sol	Rythme : crochenumero de portee 2
Note : Fa	Rythme : crochenumero de portee 2
Note : Mi	Rythme : noirenumero de portee 2
Note : Fa	Rythme : noirenumero de portee 2
Note : -	Rythme : numero de portee 2

Ce fichier nous permet d'écrire la matrice de confusion suivante :

	Noire	Croche	Blanche	Barre	Mesure	Clef de Sol
Noire	6	0	1	0		1
Croche	0	20	1	0		0
Blanche	0	0	0	0		0
Barre Mesure	0	0	0	6		1
Clef de Sol	0	0	0	0		0
Efficacite	100%	100%	0%	100%		0%

TABLE 2 – Matrice de Confusion générée à partir de la partition "Le Caillou"

7.6.3 Matrice de confusion de "Frère Jacque"

Nous étudions la partition de "Frère Jacque" suivante :

Frère Jacques

Flûte à bec soprano
<http://www.apprendrelaflute.com>

FIGURE 24 – Partition de test : "Frère Jacque"

Nous obtenons en sortie le fichier txt suivant :

```
Note : Re – Rythme : crochenumero de portee 0
Note : – Rythme : crochenumero de portee 0
Note : Do diese – Rythme : numero de portee 0
Note : Do diese – Rythme : numero de portee 0
Note : Do diese – Rythme : numero de portee 0
Note : Fa – Rythme : noirenumero de portee 0
Note : – Rythme : noirenumero de portee 0
Note : La – Rythme : noirenumero de portee 0
Note : Fa – Rythme : noirenumero de portee 0
Note : Fa – Rythme : noirenumero de portee 0
Note : – Rythme : noirenumero de portee 0
Note : La – Rythme : noirenumero de portee 0
Note : Fa – Rythme : noirenumero de portee 0
Note : La – Rythme : noirenumero de portee 0
Note : Do diese – Rythme : numero de portee 0
Note : Si – Rythme : noirenumero de portee 0
Note : Do diese – Rythme : numero de portee 0
Note : La – Rythme : noirenumero de portee 0
Note : Do diese – Rythme : numero de portee 0
Note : Si – Rythme : noirenumero de portee 0
Note : Do diese – Rythme : numero de portee 0
Note : Re – Rythme : crochenumero de portee 1
Note : – Rythme : crochenumero de portee 1
Note : Do diese – Rythme : numero de portee 1
Note : Do diese – Rythme : numero de portee 1
Note : Do – Rythme : crochenumero de portee 1
Note : Re – Rythme : crochenumero de portee 1
Note : Do – Rythme : crochenumero de portee 1
Note : Do diese – Rythme : numero de portee 1
Note : Si – Rythme : crochenumero de portee 1
Note : La – Rythme : noirenumero de portee 1
Note : Fa – Rythme : noirenumero de portee 1
Note : Do – Rythme : crochenumero de portee 1
Note : Re – Rythme : crochenumero de portee 1
Note : Do – Rythme : crochenumero de portee 1
Note : Do diese – Rythme : numero de portee 1
Note : Si – Rythme : crochenumero de portee 1
Note : La – Rythme : noirenumero de portee 1
Note : Fa – Rythme : noirenumero de portee 1
Note : Fa – Rythme : noirenumero de portee 1
Note : – Rythme : crochenumero de portee 1
```

Note : Do diese – Rythme : barre de mesure numero de portee 1
Note : Fa – Rythme : noire numero de portee 1
Note : – Rythme : croche numero de portee 1
Note : Do diese – Rythme : barre de mesure numero de portee 1
Note : Do diese – Rythme : numero de portee 1

Ce fichier nous permet d'écrire la matrice de confusion suivante :

	Noire	Croche
Noire	20	0
Croche	0	8
Efficacite	100%	100%

TABLE 3 – Matrice de Confusion générée à partir de la partition "Frère Jacque"

7.6.4 Conclusion

Dans les classes que nous avons privilégié (noirs, croches) les résultats sont tout à fait remarquables, et il ne reste nos éventuels successeurs qu'à résoudre le problème de la détection des clefs de sol, d'armure etc.... Nous leurs suggérons d'utiliser du template matching avec les symboles souhaités pour résoudre ce problème.

8 Conclusions

Ce projet nous a permis de découvrir de nombreux domaines des Sciences et de l'Ingénierie et de compléter notre formation d'ingénieur sur plusieurs points :

- Nous avons approfondis nos connaissances en traitement de l'image. En effet nous avons réutilisé les connaissances de Signaux et Systèmes 1 et 2 (autocorrélation etc...) mais nous avons aussi découvert d'autres techniques qui sont classiques (histogrammes, étude des profils verticaux et horizontaux, template matching, réseaux de neurones) dans ce domaine.
- Nous avons renforcé nos connaissances en Matlab en utilisant les fonctions relatives au traitement de l'image (imshow etc....). Nous manipulons maintenant Matlab (les matrices etc...) avec une fluidité bien supérieure à celle de l'année précédente.
- Nous avons appris le C++ ce qui nous a pris beaucoup de temps (recherche des librairies etc.....) mais c'est un langage très utilisé dans l'industrie et qui permet beaucoup de finesse.

Nous avons créé un algorithme robuste, évolutif et adaptatif qui permet de détecter de manière fiable un certain nombre de symboles de base de la musique classique.

Par un manque de temps et de connaissance préalables en C++, en traitement de l'image et en machine learning nous n'avons pas pu mener à bout les objectifs fixés : c'est à dire avoir une application android fonctionnelle. Nous vous remercions de nous avoir permis d'y travailler pendant toute cette année.

Ce projet nous a beaucoup apporté et c'est un projet que nous souhaiterions éventuellement continuer dans les années futures.

9 Annexe 1 : Répertoire des fonctions codées

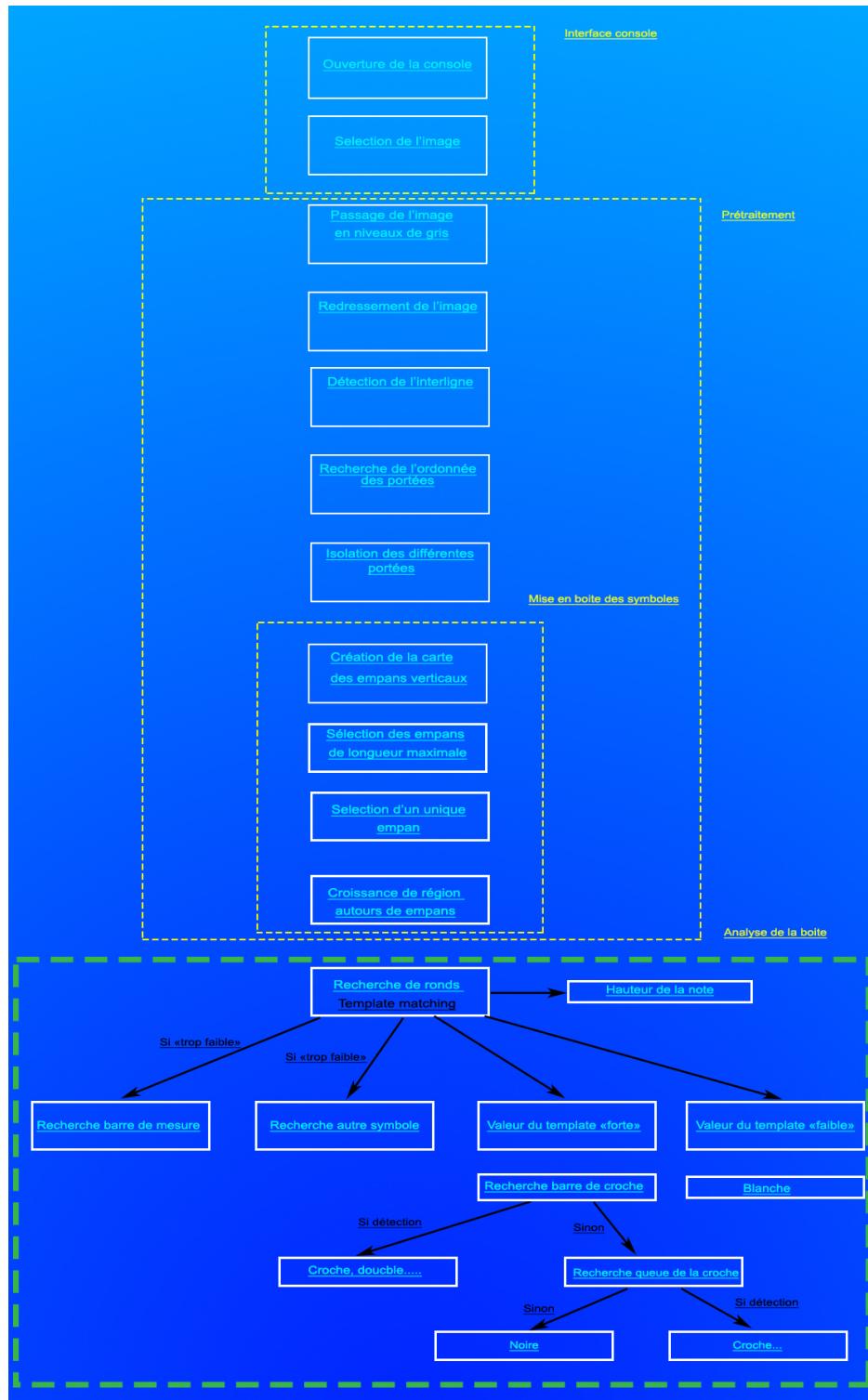


FIGURE 25 – Diagramme résumant la démarche du code

Nous présentons dans cette annexe toutes les fonctions utilisées dans notre projet auxquelles nous joignons une brève description. Il s'agit ici des fonctions C++ qui ont à peu de chose près leurs équivalents Matlab au niveau des noms et des fonctionnalités. Pour mieux suivre les différents appels de fonction dans le code du main nous proposons un petit diagramme explicatif qui résume la démarche adoptée (voir ci-dessus).

9.1 Le main

Le main décrit le déroulement de notre algorithme.

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
#include <Eigen/Dense>
#include <imageProc.h>
#include <struct.h>
#include <correction_biais.h>
#include <detectionInterligne.h>
#include <rechercheDesXis.h>
#include <sousImagePortee.h>
#include <effacementPortee.h>
#include <histogrammeLargeurLigne.h>
#include <emboitage.h>
#include <fusionVerticale.h>
#include <carteEmpans.h>
#include <carteFinaleDesEmpans.h>
#include <selectionUniqueEmpan.h>
#include <selectionPlusieursEmpans.h>
#include <emboitageImage.h>
#include <reconnaissanceNotes.h>
#include <affichageVectText.h>
using namespace cv;
using namespace std;

int main()
{
    cv::Mat imageRead = cv::imread("test/beau_sapin.jpg",
        CV_LOAD_IMAGE_GRAYSCALE);
```

```

Eigen :: MatrixXi imageUtilisee = niveauGris(imageRead ,
    255/2);
imageRead . release ();
int H = imageUtilisee . rows ();
int W = imageUtilisee . cols ();
Eigen :: MatrixXi imageCorrigee = correction_biais(
    imageUtilisee); // ON doit changer les matrices :
                    agrandissement .
imashow (imageCorrigee , 1);

// Eigen :: MatrixXi imageCorrigee = imageUtilisee ;
intVectXi result = detectionInterligne (imageCorrigee) ;
int interligneMoyen = result . i;
Eigen :: VectorXi Py = result . vec;
Eigen :: VectorXi localisation = rechercheDesXis (Py , W,
    interligneMoyen);

histogrammeLargeurLigne (imageCorrigee) ;

sousImagePorteeRes sousImagesData = sousImagePortee (
    localisation , imageCorrigee , interligneMoyen);
std :: vector<Eigen :: MatrixXi> portees = sousImagesData .
    portees;

int nportees = portees . size ();

for (int i = 0 ; i < nportees ; i++){
    imashow (portees [ i ] , i );
}

Eigen :: VectorXi abscisseTroisiemeLigne =
    sousImagesData . abscisseTroisiemeLigne;
Eigen :: VectorXi offsetX = sousImagesData . offsetX ;
Eigen :: VectorXi Hprime = sousImagesData . Hprime ;
effacementPortee (imageCorrigee , Py , interligneMoyen ,
    localisation , offsetX , Hprime , 0 , W , 1);

Eigen :: MatrixXi cartedesempans=carteEmpans (
    imageCorrigee);

```

```

Eigen :: MatrixXf imagesecondephase=
    selectionPlusieursEmpans (imageCorrigee) ;
fusionVerticale (imageCorrigee , cartedesempans) ;

Eigen :: MatrixXi varcarteFinalDesEmpans=
    carteFinaleDesEmpans (imagesecondephase ,
    interligneMoyen , cartedesempans) ;

int nombre_empans =0 ; // nombre d'empans dans la carte
Eigen :: MatrixXi carteterminale=selectionUniqueEmpan (
    interligneMoyen , varcarteFinalDesEmpans ,
    nombre_empans) ;
//imashowcarte (carteterminale , 2);
std :: vector<quadInt> resultatEmboitage =
    emboitageImage (imageCorrigee , carteterminale ,
    interligneMoyen , nombre_empans) ;
quadInt note;

// voir les notes
for (int i = 0 ; i < resultatEmboitage . size () ; i++)
{
    // imashownotes (resultatEmboitage [ i ] ,
    // imageCorrigee , i );
    // waitKey (0);
    // */

    note=resultatEmboitage [ i ];
    int xh , xb , yg , yd;
    xh = note.a;
    xb = note.b;
    yg = note.c;
    yd = note.d;
    Eigen :: MatrixXi imageNote = imageCorrigee . block (xh
        , yg , xb-xh , yd-yg) ;
    imashow (imageNote , 100+i ) ;

    int premiere_ligne = 522;

    trioIntString coordonneesnote=reconnaissanceNotes (
        imageNote , premiere_ligne , interligneMoyen , xb ,

```

```

    yg ,xh ,yd) ;

    affichageavec text (imageNote , coordonnesnote . c ) ;
}

waitKey (0) ;
// Wait
for a keystroke in the window
return 0;
}

```

9.2 Correction du biais

9.2.1 Correction du biais multithreadé sans fft

Le but de la correction du biais est de corriger le biais (l'inclinaison) éventuel de l'image utilisée. Pour accélérer le processus, nous avons multithreadé le programme afin de rendre son exécution plus rapide.

Nous avons utilisé une fonction d'autocorrélation que nous avons programmée nous même en calculant naïvement la somme.

9.2.2 Correction du biais avec fft

Le but de cette correction était d'améliorer encore la rapidité de la correction du biais, le multithreading rendant le processus toujours trop lent. Le principe de la correction est basé sur une autocorrélation en 2D très couteuse en temps si exécutée naïvement.

Le but de cette amélioration est d'utiliser une librairie utilisant un algorithme en papillon pour le calcul de la transformée de Fourier de l'autocorrélation (qui n'est qu'un produit dans l'espace de Fourier) ce qui permettrait un gain énorme en temps de calcul.

Malheureusement, à cause d'erreurs d'adressage multiples dont les causes nous sont encore inconnues, nous n'avons pas été à même de mettre en place un algorithme fonctionnel qui aurait pourtant permis de faire gagner un temps substantiellement précieux.

9.3 Détection de l'interligne

Cette fonction est aussi multithreadée. Le but de cette fonction est de détecter et de calculer l'interligne entre les différentes lignes d'une portée en utilisant un profil vertical calculé par ledit multithreading.

9.4 Recherche des Xis

Cette fonction recherche les différents pics de pixels horizontaux afin de déterminer les positions des lignes de portée de la partition.

9.5 Sous Image Portee

Cette fonction coupe l'image en différentes sous images, afin d'isoler les différentes portées pour les traiter de manière indépendante.

9.6 Effacement Portee

Nous effaçons les différentes lignes de portée de manière "intelligente".

9.7 Histogramme Largeur Ligne

Cette fonction permet de calculer l'épaisseur moyenne des lignes en parcourant verticalement à plusieurs endroits la partition et comptant la longueur verticale maximale des objets qu'on y rencontre. La valeur maximale non-nulle de cet histogramme est supposée être l'épaisseur des lignes de portée.

9.8 Fusion verticale

Permet de séparer deux empans différents, et de ne garder qu'un de deux mêmes empans.

9.9 Carte Empans

Cette fonction est une étape de la mise en boite : le but est de créer une première carte d'empans.

9.10 Carte Finale des Empans

Le but de cette phase est de sélectionner l'empan de longueur maximale en utilisant la première carte construite précédemment.

9.11 Selection Unique Empan

Cette fonction est une étape de la mise en boite : le but est de créer une carte d'empans avec un seul empan vertical.

9.12 Selection Plusieurs Empans

Le but de cette phase est d'extraire les empans.

9.13 Emboitage Image

On fait une croissance de région limitée spatialement à partir des empans verticaux trouvés précédemment pour faire des "boîtes" encadrant les symboles voulus.

9.14 Reconnaissance Notes

Cette fonction a été développée précédemment.

9.15 Affichage avec texte

Permet d'écrire les résultats de la reconnaissance.

9.16 Interface console

La nouvelle fonction main est celle qui permet de lancer l'interface console qui nous permet de ne pas recompiler à chaque fois le programme lorsque l'on désire changer l'image sur laquelle on travaille.

10 Annexe 2 : Délivrables

Nous vous fournirons avec notre projet le code Matlab que nous avons utilisé afin de tester les algorithmes de prétraitement. Ils sont sous forme de scripts et de fonctions ; le script principal s'appelle testm, et nos fonctions suivent la syntaxe suivante :

// Commentaires descriptifs donnant l'utilisateur la façon d'utiliser la fonction, et ce qu'elle rend.

// En commentaire : données de tests.

// La fonction à proprement parler.

11 Remerciements

Enfin, nous souhaiterions remercier les différentes personnes qui nous ont apporté une aide précieuse tout au long de ce projet. La liste est, longue mais nous souhaiterions remercier spécifiquement les personnes suivantes :

- Florence ROSSANT pour sa thèse : "Reconnaissance de partitions musicales par modélisation floue des informations extraites et des règles de notation", thèse sans laquelle nous n'aurions jamais pu autant avancer dans un si court laps de temps.
- Messieurs Rossignol et Ianotto pour leur encadrement tout au long de ces huits mois.
- Messieurs Geist et Fix pour leur aide dans le choix des méthodes de reconnaissance des symboles.
- Monsieur Colette pour ses déblocages de nos problèmes de code.
- Monsieur Frezza Buet pour nous avoir dissuadés définitivement d'utiliser des réseaux de neurones
- Les fondateurs de Matlab pour nous avoir permis de tester nos algorithmes.
- Les fondateurs des différentes librairies C++ que nous avons utilisées (Eigen, OpenCV, Dирent, fftw).