



Rapport de stage 2A

---

# Logiciel d'aide à l'improvisation musicale

---

**SCRIME de Bordeaux**

Maitres de stage :

Yacine AMAROUCHENE

Myriam DESAINTE-CATHERINE

**Jérémy LIXANDRE**

Stagiaire (ENSEIRB-MATMECA)

Dates : 01/06/2017 - 28/07/2017

# Résumé

J'ai effectué mon stage au laboratoire du SCRIME de Bordeaux sous l'encadrement de Myriam Desainte-Catherine (SCRIME) et Yacine Amarouchene, chercheur en physique au LOMA. Ceux ci cherchaient à développer un outil informatique d'aide à l'improvisation musicale, domaine encore peu étudié dans le domaine de l'informatique musicale.

L'objectif était de permettre à un ensemble de musiciens d'avoir un feedback visuel en temps réel de leur jeu, et ainsi savoir si ils jouent "juste" selon des critères choisis arbitrairement.

Afin de remplir cet objectif, il a été décidé de représenter le jeu des musiciens sous la forme d'une matrice colorée, ou chaque case  $(i, j)$  représenterait la "justesse" du musicien  $i$  par rapport au musicien  $j$  : par exemple vert si ils jouent bien ensemble, rouge sinon. Cette idée de "justesse" sera quantifiée par la notion de corrélation dans le logiciel.

Le but était donc de permettre l'affichage en temps réel d'une matrice de corrélation correspondant au jeu des musiciens. Qui plus est, le livrable devait être le plus modulaire possible afin de permettre très facilement de changer les méthodes de calcul de la matrice, et ainsi pouvoir étudier les interactions entre les musiciens selon des critères précis et choisis. Il était aussi nécessaire d'envisager à l'avance les futures améliorations du logiciel pour créer une architecture adaptée.

De plus, pour des raisons de portabilité nous avons choisi d'implémenter ce logiciel sur la plateforme intégrée Bela, spécialisée dans le traitement du signal, et pouvant gérer un nombre important de flux audio simultanément.

Le projet a été implémenté en C++ avec un serveur web en Javascript. Le programme se lance depuis l'ordinateur ou depuis l'appareil, et affiche sur une page web une matrice colorée correspondant aux corrélations entre les musiciens. De plus, il est très simple de modifier de nombreux paramètres du programme à l'exécution grâce à un fichier de configuration, et même d'implémenter soi-même ses propres plugins de calcul sans avoir à recompiler le projet.

Enfin j'ai rédigé une documentation précise du fonctionnement du logiciel et ai proposé des possibilités d'améliorations ultérieures avec la marche à suivre.

# Table des matières

<b>1</b>	<b>Remerciements</b>	<b>3</b>
<b>2</b>	<b>Introduction</b>	<b>3</b>
<b>3</b>	<b>Présentation du laboratoire</b>	<b>4</b>
<b>4</b>	<b>Présentation de la mission technique</b>	<b>5</b>
4.1	Objectifs généraux . . . . .	5
<b>5</b>	<b>Réalisation</b>	<b>8</b>
5.1	Méthode de travail . . . . .	8
5.2	Livrable . . . . .	8
5.3	Documentation . . . . .	12
<b>6</b>	<b>Conclusion</b>	<b>14</b>
<b>7</b>	<b>Bilan personnel</b>	<b>14</b>

# 1 Remerciements

Avant de développer, je souhaite remercier le SCRIME de m'avoir accueilli pendant ce stage.

Madame DESAINTE-CATHERINE et Monsieur AMAROUCHENE, qui m'ont proposé ce stage m'ayant permis d'en apprendre plus dans le domaine de l'informatique musicale, et qui m'ont encadré et conseillé pendant mon travail.

Les membres du SCRIME et du LaBRI, pour leurs bons conseils et leur sympathie, ainsi que toutes les personnes avec qui j'ai pu travaillé, m'ayant permis de m'enrichir de cette expérience.

# 2 Introduction

Dans le cadre de mon stage de deuxième année, j'ai choisi de passer 8 semaines au sein du SCRIME de Bordeaux, dans le cadre d'un projet de recherche sur l'improvisation musicale. Etant donné que j'avais décidé d'effectuer ma troisième année d'études à l'étranger, j'ai été contraint de raccourcir la durée de mon stage de manière à partir à temps.

J'ai choisi ce stage car le domaine d'informatique musicale m'intéresse tout particulièrement : j'ai d'ailleurs suivi en deuxième année à l'ENSEIRB deux cours sur l'informatique musicale et la création sonore, et j'ai également effectué un projet de 2ème année sur l'improvisation musicale. Les clients de ce projet, Madame Desainte-Catherine et Monsieur Amarouchene, m'ont proposé à l'issue de celui ci d'effectuer un stage avec eux sur un sujet similaire.

Enfin j'étais assez curieux de découvrir le milieu de la recherche, dans lequel je n'avais pas envisagé d'aller auparavant, mais qui pouvait m'intéresser dans ce domaine précis.

Ce rapport fait état du travail réalisé entre le 1er Juin 2017 et le 28 Juillet 2017. Après avoir présenté le laboratoire, j'expliquerai la mission technique qui m'a été assignée. Puis j'exposerai les détails de mon travail, avant un bilan personnel.

### 3 Présentation du laboratoire



J'ai effectué mon stage sous la direction du SCRIME (Studio de Création et de Recherche en Informatique et Musiques Expérimentales) de Bordeaux, spécialisé comme son nom l'indique en informatique musicale, laboratoire rattaché au LaBRI que je ne crois pas avoir besoin de présenter.

Ce laboratoire développe un grand nombre de projet liés à l'informatique musicale, notamment des projets sur la spatialisation du son, l'extraction de données sonores, et l'improvisation musicale.

Beaucoup de chercheurs de ce laboratoire travaillent avec des musiciens improvisateurs, très souvent dans le milieu de la musique électro-acoustique et des musiques expérimentales.

J'ai cependant effectué mon stage dans les bureaux du LaBRI en étant encadré par M.Yacine AMAROUCHENE, chercheur en physique au LOMA (Laboratoire Ondes et Matière d'Aquitaine), ainsi que Mme Myriam Desainte-Catherine du SCRIME.



## 4 Présentation de la mission technique

Cette partie présente le contexte du stage ainsi que les objectifs du projet.

### 4.1 Objectifs généraux

Le projet consistait en la création d'un outil informatique pour l'improvisation musicale, permettant à des musiciens jouant en même temps d'avoir une retrospective visuelle et en temps réel de leur improvisation.

De manière très simple, on voulait par exemple que 4 musiciens, jouant ensemble et branchés sur l'outil, puissent visualiser sur un écran si leur improvisation est "juste" ou "fausse".

A partir de ce problème très général, nous avons par la suite décidé d'utiliser une visualisation permettant de comparer les signaux audio deux à deux. Il s'agira de voir par exemple si, dans un groupe de cinq musiciens, les musiciens 1 et 2 jouent bien ensemble, ainsi que les musiciens 2 et 3, 1 et 3... On préfère ceci à une visualisation globale qui mettrait tous les musiciens dans le même paquet, ainsi on pourra constater le décalage d'un musicien par rapport à un autre avec plus de précision.

A partir de cette contrainte, il a été décidé de d'utiliser une visualisation sous forme matricielle. Ainsi l'objet ligne  $i$  colonne  $j$  permettra de comparer les musiciens  $i$  et  $j$ .

Enfin, pour comparer deux pistes musicales, on introduit la notion de **corrélation**. Dans la suite de ce rapport, la corrélation désigne simplement une fonction  $f$  prenant deux tableaux en argument (représentant des signaux sonores), et retournant un flottant entre 0 et 1.

Cette fonction  $f$  possède de plus ces propriétés :

- $f$  est symétrique, donc  $f(i, j) = f(j, i)$
- si  $t$  est un tableau rempli de 0,  $f(t, i) = 0$  pour tout  $i$
- si  $t$  est non nul,  $f(t, t) = 1$

A partir de ces fonctions de corrélation, on pourra obtenir une matrice de coefficients que l'on pourra décider d'afficher selon ses préférences.

Ainsi le problème se résume à la chose suivante :

```
Entrée : N pistes musicales (musiciens et/ou pistes enregistrées)
Sortie : une matrice M de taille NxN,
        où M(i,j)=M(j,i) est la corrélation des pistes i et j.
```

Cette matrice devait se mettre à jour régulièrement et en temps réel. De plus, la notion de "corrélation" étant large et redéfinissable, on souhaitait pouvoir changer de fonction de corrélation facilement selon les besoins, voire de pouvoir ajouter nos propres fonctions de corrélation au logiciel.

Il a été de plus décidé d'utiliser comme support la plateforme Bela, plateforme embarquée similaire à des objets comme l'Arduino, mais spécialisée dans le traitement du signal et particulièrement du signal sonore. Elle contient un grand nombre d'entrées/sorties audio et analogiques,

permet d'atteindre des temps de latence sonore très faibles et est donc idéale pour ce genre de projets.

Elle se connecte par USB à un ordinateur, et on y accède via une adresse IP.

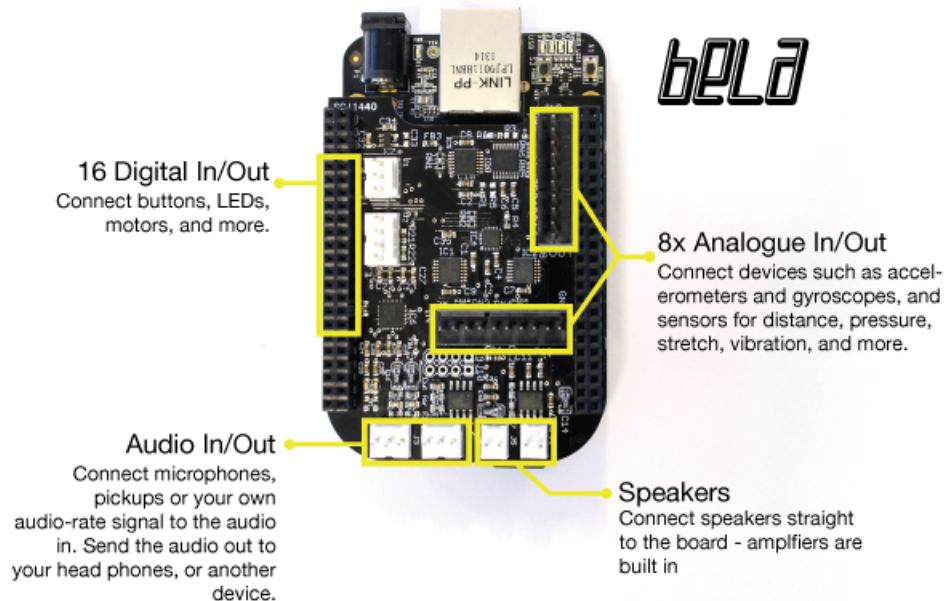


FIGURE 1 – Plateforme Bela

Précisons aussi que le projet démarrait de zéro, il n'y avait donc pas de logiciel préalable utilisant déjà les procédés décrits plus tôt. Ainsi l'objectif était d'arriver en fin de stage à un livrable utilisable, mais améliorable facilement, de manière à obtenir à long terme un outil simple d'utilisation avec de nombreuses fonctionnalités.

Finalement les objectifs visés étaient les suivants :

- répondre au problème exposé ci dessus, à la fois pour des signaux réels et pour des fichiers audio
- proposer une visualisation simple de la matrice (sur une page web)
- rendre simple le changement d'algorithmes de calcul de la matrice (notamment le changement de fonctions de corrélation)
- permettre la possibilité d'ajouter des effets audio sur chaque piste avant calcul.
- de manière générale, faire un logiciel modulaire qui pourra être facilement améliorable ou modifiable par la suite.

L'intérêt de ce logiciel pour la recherche musicale réside dans le fait qu'il n'y a à ce jour que très peu d'outils dédiés spécifiquement à l'analyse d'improvisation, alors que ce type d'outil pourrait être très utile pour de nombreux musiciens voulant avoir une retrospective en temps réel de leur jeu, et ainsi s'adapter.

Voici quelques exemples d'utilisations possibles d'un tel logiciel :

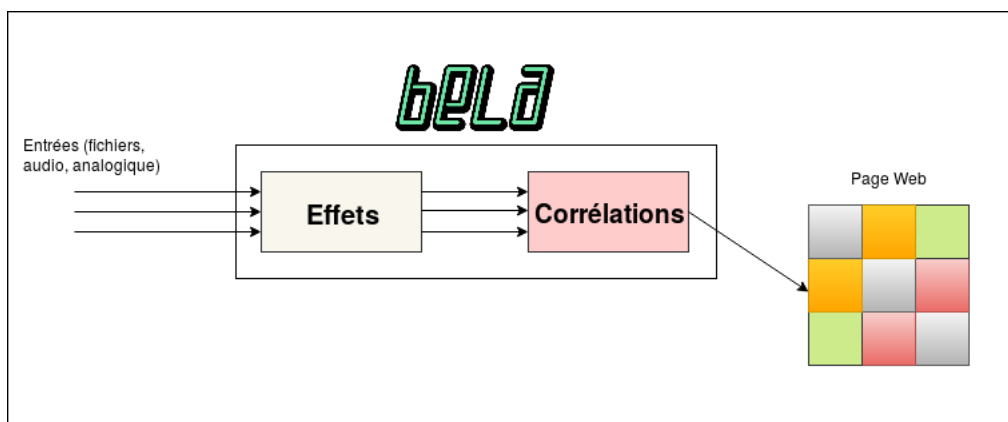


FIGURE 2 – Résumé des objectifs du projet

- un musicien seul, apprenant un instrument, pourrait jouer accompagné d’une piste audio enregistrée, pour savoir si il joue en rythme ou non. On utiliserait alors des corrélations basées sur la recherches de données relatives au rythme
- un groupe de musiciens enregistrant un morceau pourraient utiliser ce logiciel pour détecter les petits décalages rythmiques entre les instruments
- un chanteur pourrait s’entraîner à chanter juste, en utilisant des corrélations liées au pitch ou aux notes
- un groupe de musiciens pourrait chercher à détecter si tout le monde joue dans la même gamme de notes

Nous avons de plus abordé d’autres possibilités d’utilisation ultérieure du logiciel. Il serait par exemple possible de changer le système de visualisation. Au lieu de percevoir les corrélations comme une image, il est tout à fait possible de les représenter par un son. Par exemple, un coefficient de corrélation (qui est un flottant entre 0 et 1) pourrait être représenté sous la forme d’un son modulé en amplitude : le résultat final ne serait donc plus une matrice mais un ensemble de sons synthétisés sur une sortie audio de la plateforme Bela.

Cette idée peut donc avoir un grand intérêt pour la recherche, par exemple pour les possibilités d’apprentissage de la musique que cela implique.



## 5 Réalisation

Cette section donne un aperçu du travail réalisé au cours de ce stage. La première partie donne une idée des méthodes de travail employées, tandis que les sous-chapitres suivants détaillent le livrable rendu aux clients.

### 5.1 Méthode de travail

Etant seul sur ce projet, j'ai travaillé par méthode agile. Mes deux premières semaines au laboratoire m'ont permis de faire état des solutions disponibles pour le traitement sonore en temps réel.

J'ai également appris en début de stage les bases de l'utilisation de Node JS, une plateforme logicielle utilisée pour faire des applications client / serveur en Javascript, dont j'avais besoin pour réaliser la visualisation matricielle.



J'ai ensuite pu travailler sur la plateforme intégrée Bela, après avoir appris à l'utiliser.

En milieu de stage, j'ai pu présenter mes avancées aux chercheurs et stagiaires du laboratoire lors de la conférence du Cycle SCRIME. J'ai également pu découvrir la multitude de projets en cours dans cet institut.

Enfin, j'ai réalisé en fin de stage une documentation détaillée de l'outil.

De manière générale, j'ai travaillé de manière très autonome, en cherchant moi même des solutions aux problèmes rencontrés, et en apprenant par moi-même à prendre en main les outils que je ne connaissais pas (Node js, Bela, dlopen, socket.io).

### 5.2 Livrable

Le livrable a été codé en C++ avec un serveur web en Javascript. Il est intégré à la plateforme Bela et se compile en ssh dessus, ou via l'IDE intégré.

Il est possible de brancher jusqu'à 10 entrées physiques différentes sur l'appareil. De plus, il est possible d'ajouter des pistes audio préenregistrées au mix final. On a estimé qu'il était possible d'ajouter jusqu'à 15 flux audio (entrée physique et fichier) avec un résultat satisfaisant, c'est à dire une matrice de corrélation se mettant à jour en temps réel sans retard. Ce résultat dépend aussi des fonctions de calcul choisies.

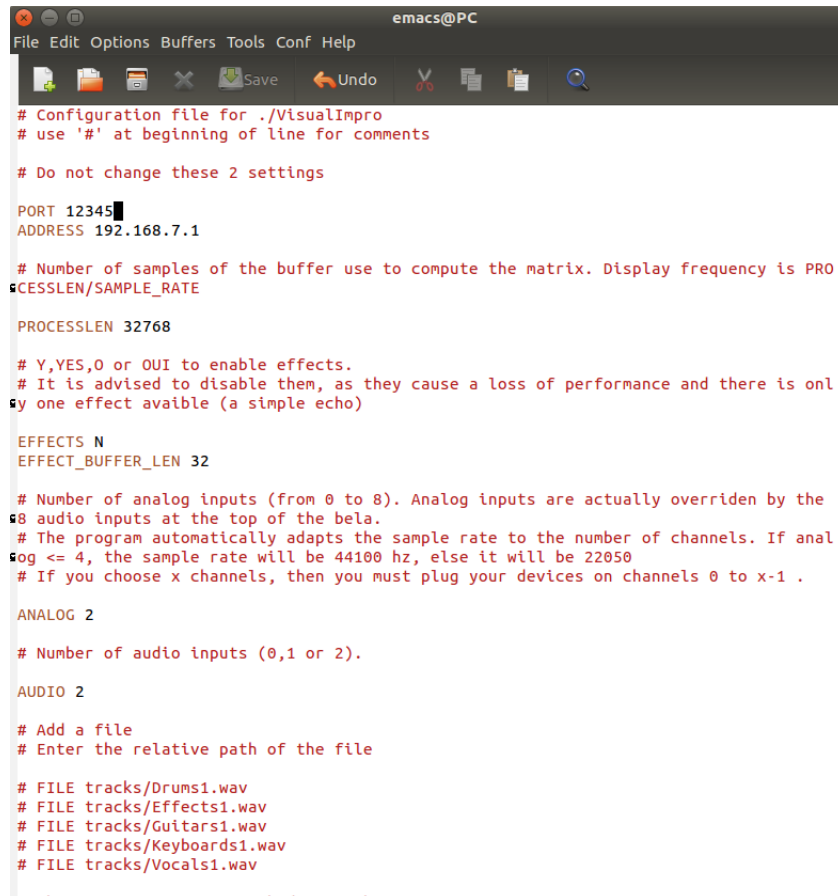
Il existe deux manières de lancer le programme. La première nécessite uniquement un ordinateur capable de se connecter en ssh sur la plateforme, soit tout ordinateur sur Linux, Windown, Mac, cependant elle est un peu fastidieuse. La seconde nécessite un PC sous Linux avec Nodejs installé, mais est extrêmement simple et s'exécute depuis l'ordinateur et pas en ssh.

#### Lancement depuis l'ordinateur

Pour exécuter le programme depuis l'ordinateur, il est nécessaire de récupérer un fichier Visua-IImproExe.zip. Cette archive contient un fichier de configuration à modifier, ainsi qu'un exécutable.

```
jeremy@PC:~/Enseirb/stage2A/bela/VisualImproExe$ ls
config.cfg  index.html  Makefile  node_modules  server.js  src  tracks  VisualImpro
```

FIGURE 3 – Dossier de l'exécutable sur ordinateur



```
emac@PC
File Edit Options Buffers Tools Conf Help

# Configuration file for ./VisualImpro
# use '#' at beginning of line for comments

# Do not change these 2 settings

PORT 12345
ADDRESS 192.168.7.1

# Number of samples of the buffer use to compute the matrix. Display frequency is PRO
CESSLEN/SAMPLE_RATE

PROCESSLEN 32768

# Y,YES,0 or OUI to enable effects.
# It is advised to disable them, as they cause a loss of performance and there is onl
y one effect available (a simple echo)

EFFECTS N
EFFECT_BUFFER_LEN 32

# Number of analog inputs (from 0 to 8). Analog inputs are actually overridden by the
8 audio inputs at the top of the bela.
# The program automatically adapts the sample rate to the number of channels. If anal
og <= 4, the sample rate will be 44100 hz, else it will be 22050
# If you choose x channels, then you must plug your devices on channels 0 to x-1 .

ANALOG 2

# Number of audio inputs (0,1 or 2).

AUDIO 2

# Add a file
# Enter the relative path of the file

# FILE tracks/Drums1.wav
# FILE tracks/Effects1.wav
# FILE tracks/Guitars1.wav
# FILE tracks/Keyboards1.wav
# FILE tracks/Vocals1.wav
```

FIGURE 4 – Fichier de configuration du programme

Il faut tout d'abord modifier le fichier de configuration selon les paramètres désirés.

ANALOG et AUDIO désignent le nombre d'entrées audio et analogiques voulues. Bela comporte 2 entrées audio et 8 entrées analogiques. Le programme choisit automatiquement la meilleure configuration selon le nombre d'entrées choisies.

FILE désigne le nom d'un fichier à ajouter au mix. On peut ajouter des fichiers au dossier `tracks` pour plus de simplicité.

PROCESSLEN et EFFECT\_BUFFER\_LEN sont des paramètres liés au nombre d'échantillons audio à traiter en temps réel.

Il est aussi possible de changer les fonctions de calcul de corrélations.

Lorsque le fichier est configuré, il suffit alors de lancer l'exécutable `./VisualImpro`. On peut alors suivre ceci depuis le terminal :

```

jeremy@PC:~/VisualImpro/VisualImproExe$ ./VisualImpro
scp ./configtmp.cfg root@192.168.7.2:/root/Bela/projects/VisualImpro/config/
configtmp.cfg                               100% 1176    1.2KB/s   00:00
VISUALIMPRO SERVER:
Server listening on 192.168.7.1:12345
ssh root@192.168.7.2 'cd /root/Bela/projects/VisualImpro && ./VisualImpro config/./conf
igtmp.cfg' &
***** VisualImpro *****
CONNECTED: 192.168.7.2:56376

----- Settings -----
* Files : 0
* Standard Audio Tracks : 2
* Additionnal Audio Tracks : 2
* Sample Rate : 44100
* Buffer Processing Length : 32768
* Processing Speed (bufflen/samplerate) : 0.743039 s
* Effects disabled

```

FIGURE 5 – Vue du programme depuis un terminal

De plus, une page web sous Firefox s'ouvre automatiquement, affichant en temps réel la matrice de corrélations.

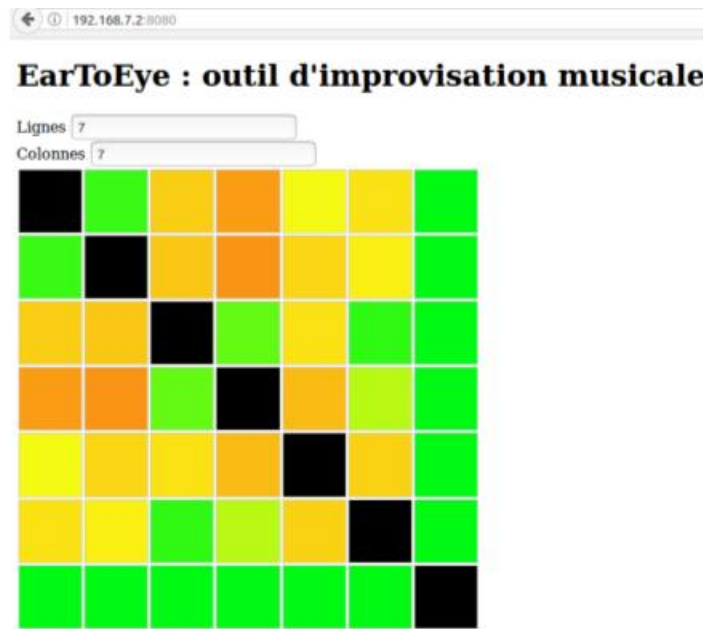


FIGURE 6 – Matrice de corrélation sur une page web à un instant  $t$

Ainsi, la case ligne  $i$  colonne  $j$  symbolise la corrélation entre les pistes  $i$  et  $j$ . Les pistes sont dans l'ordre : fichiers - analogiques - audio.

Pour arrêter le programme, il suffit d'appuyer sur CTRL + C. Tous les fichiers temporaires sont effacés du dossier, de même que sur Bela. De plus l'utilisateur a accès à un fichier `log` permettant de retracer l'évolution de la matrice. Ces fichiers peuvent être utiles pour une étude de corrélation

sur des temps longs, supérieurs à la seconde.

Cette méthode d'utilisation est donc simple et a été réalisée dans le but de permettre à de futurs développeurs d'implémenter facilement une interface graphique pour exécuter le programme au lieu de le lancer depuis un terminal.

## Utilisation en ssh

Il est aussi possible de lancer le programme "à la main", c'est à dire en ssh sur Bela. Pour cela, il est nécessaire de lancer séparément un serveur web (dans un dossier de la plateforme), le programme et la page web. Il est aussi nécessaire de modifier le fichier de configuration dans le même répertoire que l'exécutable.

Cette méthode est plus compliquée, de plus elle est plus couteuse car le serveur web et le programme tourneront sur le même processeur (contrairement à la méthode précédente où le serveur tourne sur l'ordinateur). Cependant il est utile de la connaître pour un développeur.

## Ajout de fonctions de corrélation

De plus, il est possible de modifier facilement les méthodes de calcul de la matrice. En effet, pour chaque couple de pistes, ce calcul est fait en trois temps :

1. une fonction de "preprocessing" traite les deux signaux d'entrée séparément pour sortir deux signaux de sortie (ex : filtrage du signal, transformée de Fourier). Cette fonction peut également servir à diminuer le temps de calcul global. En effet les corrélations sont calculées un nombre quadratique de fois sur des tableaux de taille de l'ordre de 10000 (car les extraits audio corrélés durent généralement entre 0,1 et 1 seconde). Il est donc très utile de pouvoir réduire la taille de ces tableaux en amont, en conservant par exemple un échantillon sur 2, 3 ou 4.
2. une fonction de corrélation prend en argument les deux tableaux (signaux traités précédemment) et retourne une corrélation (un flottant entre 0 et 1) telle que définie dans la section précédente.
3. une fonction d'échelle de couleur prend en argument ce flottant et renvoie un triplet RGB représentant une couleur. Cette fonction permet de décider du dégradé de couleurs de la matrice (ex : la matrice va du blanc au noir, ou du vert au rouge).

Voici en image un résumé du traitement :

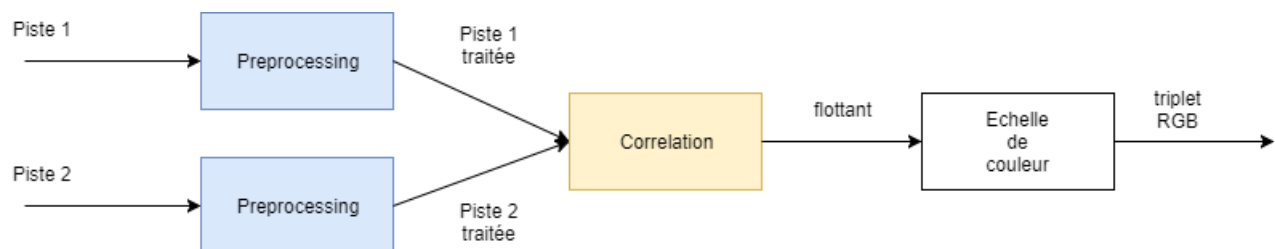


FIGURE 7 – Algorithme de calcul d'une case de la matrice

Il est possible avec l'architecture du logiciel de modifier chacune de ces trois fonctions. Le choix est fait dans le fichier de configuration. De plus, il est possible d'ajouter une fonction de ce type sans avoir à recompiler le programme. La seule contrainte est que les noms de fichiers et prototypes de fonctions soient conformes à un certain standard.

Un exemple de prototype imposé pour une fonction de calcul de corrélation :

```
float CoeffX (vector<float>, vector<float>);
```

Lorsque ces contraintes sont respectées, il suffit de placer la fonction dans un dossier du programme et compiler avec `make`. Toutes les fonctions du dossier sont alors compilées dans une bibliothèque dynamique, laquelle est chargée dynamiquement au début du programme grâce à `dlopen`.

Quelques exemples de fonctions ont été implémentées :

- des échelles de couleur allant du vert au rouge, du noir au blanc
- des fonctions de préprocessing sélectionnant un échantillon sur deux, ou calculant une enveloppe d'énergie moyenne
- une fonction de corrélation calculant un produit scalaire entre deux tableaux

De même que précédemment, l'architecture a été pensée de manière à ce que l'implémentation d'une interface graphique pour le choix de corrélation, ou l'ajout d'une fonction, soient simples à réaliser.

## Autres pistes

D'autres pistes d'améliorations ont été étudiées. Il est par exemple possible d'ajouter des effets audio à chaque piste en amont de la corrélation, par exemple ajouter à la piste 1 une reverb, à la piste 2 un changement de hauteur, etc. La boucle principale du programme permet déjà de gérer des effets en amont, et il est également possible de choisir dans le fichier de configuration si l'on veut activer les effets.

Cependant il n'est pas encore possible de choisir ses effets à l'exécution, mais seulement de les rajouter à la main dans le code. De plus l'ajout d'effets peut être très coûteux en temps et impose le plus souvent une latence supplémentaire entre l'instant d'émission du signal d'origine et l'instant de calcul de la matrice.

## 5.3 Documentation

Le livrable étant censé poser les bases du logiciel, il sera amené à évoluer rapidement, en ajoutant des nouvelles corrélations, de nouveaux effets, ou en changeant la visualisation. Il était donc nécessaire de réaliser une documentation précise du fonctionnement du logiciel, aussi bien pour l'utilisateur que le programmeur.

En fin de stage, j'ai donc écrit une documentation d'une quinzaine de pages reprenant très précisément chaque point du fonctionnement du logiciel. Celle-ci est proposée en annexe. Elle détaille le fonctionnement du logiciel pour un simple utilisateur, mais aussi l'implémentation pour un développeur. Les bases de la programmation sur Bela sont expliquées, ainsi que la structure de

base du programme.

Elle contient aussi des propositions d'amélioration du logiciel avec des méthodes pour les implémenter.

## 6 Conclusion

Finalement le livrable rendu répond aux besoins suivants :

- permettre de traiter un grand nombre de flux audio simultanément
- utilisation simple, avec peu de contraintes d'installation
- afficher une matrice de corrélation entre les différents musiciens
- permettre de modifier les paramètres de calcul sans recompiler le programme, et d'ajouter soi même des fonctions de calcul
- permettre à d'autres développeurs de faire évoluer le logiciel simplement, en y ajoutant par exemple des interfaces graphiques

Il pose ainsi les fondations d'un futur logiciel multi-usage, avec notamment les possibilités d'amélioration suivantes :

- fonctions de corrélation basées sur la détection du rythme et de notes
- utilisation d'une sortie audio comme "visualisation" des corrélations
- ajout d'effets audio en amont
- interface graphique d'ajout de fonctions de corrélation, basées sur d'autres langages comme PureData, Faust
- interface graphique / web de gestion des paramètres du son

## 7 Bilan personnel

A titre personnel, je garde une très bonne expérience de ce stage de deuxième année. J'ai notamment apprécié le fait d'être assez autonome dans la gestion de mon travail, bien que je devais évidemment rendre des comptes régulièrement. J'ai aussi apprécié le fait de ne pas être limité au niveau des idées de réalisation.

J'ai également découvert un domaine que je n'avais jamais eu la curiosité de découvrir auparavant, celui de la recherche. J'ai pu observer au cours du stage la diversité des projets de recherche du laboratoire, en assistant notamment à une conférence du SCRIME en Juin. L'ambiance était également très bonne.

De plus, je tire de ce travail de nouvelles compétences : en Javascript avec Nodejs, en C et C++, ainsi que sur la programmation sur plateforme intégrée et la gestion du temps réel. J'ai finalement pu découvrir de nouvelles choses sur l'informatique musicale, qui est ce pourquoi j'avais voulu faire ce stage.