*C. Colnat, K. Giraud-Esclasse, E. Iliesco, T. Mazaleyrat, A. Musabini, E. Poirion et E. Thomas **

L'improvisation musicale intégrée à une nouvelle génération de robot humanoïde: NAO, un futur grand musicien

***Abstract*— De nos jours, les robots interagissent avec leur environnement visuel de manière complexe pour répondre aux besoins de l'avancée technologique. Leur interaction auditive avec le monde extérieur n'a pas été autant développée notamment avec l'environnement musical. L'utilisation d'un robot NAO et d'un logiciel d'improvisation musicale a permis de tester et de développer ce secteur. PyOracle a été utilisé pour créer de l'improvisation musicale et la beat-tracker Aubio est utilisé pour synchroniser les mouvements du robot sur la pulsation de la musique. Cette intégration a pour but de permettre au robot d'enregistrer, d' improviser et de danser en rythme.**

I. Introduction

La robotique est un des domaines technologiques où le développement des composants et des outils croît très rapidement ces dernières années. La recherche actuelle est orientée vers l'humanisation de cette technologie par le biais de robot dit « humanoïde » c'est-à-dire ressemblant à l'homme. Nao est un robot humanoïde de 58 cm programmable développé par la société Aldebaran-Robotics[1] [1]. Des outils sont disponibles pour contrôler le robot, notamment pour le faire parler, marcher ou reconnaître des structures.

La partie auditive de ce robot est actuellement la moins développée. Des modules existent pour le détection sonore ou la reconnaissance de parole mais aucuns ne permettent une interaction musicale du robot. Le projet mené avec ce robot a pour but de compléter cette partie par l'intégration d'un logiciel d'improvisation musicale nommé PyOracle [2]. Ce logiciel, implémenté en python, permet l'analyse de la structure d'un échantillon musical et la génération d'une piste audio improvisée. En complément, un beat-tracker nommé Aubio [3]–[5] est ajouté afin d'apporter la connaissance du rythme d'une piste audio au NAO. Il peut ainsi improviser sur une piste audio acquise et battre le rythme correspondant.

II. Choix matériel et logiciel

A. NAO – un robot humanoïde

Le robot humanoïde NAO utilisé est la dernière génération disponible, appelée NAO Next Gen. Il possède entre autres une distribution Linux “Gentoo”, un processeur Intel Atom fonctionnant à 1.66GHz, 25 degrés de liberté, 4 microphones et 2 haut-parleurs.

La gestion de l'audio avec NAO est effectuée avec les modules ALAudioRecorder et ALAudioPlayer de NAOqi (Fig. 1) [6], la bibliothèque de fonctions intégrée dans le robot. Les fonctions d'enregistrement du module ALAudioRecorder permettent de récupérer les sons perçus par les 4 microphones placés sur la tête du NAO (Fig. 2). Ces sons enregistrés respectent alors l'un des 2 formats suivant, suivant le choix de l'utilisateur:

- 4 voies échantillonnées à 48kHz (extension .wav ou .ogg).
- 1 voie échantillonnée à 16kHz (extension .wav ou .ogg).

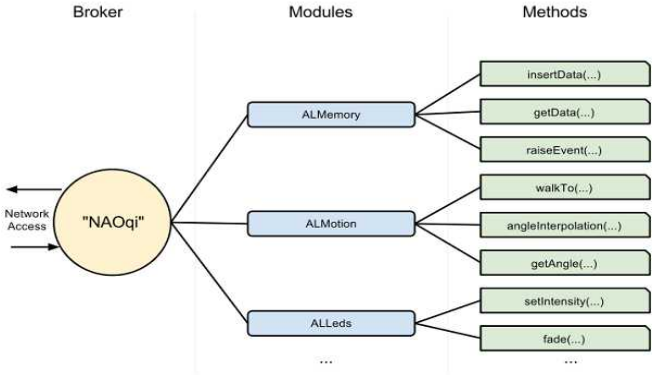

Figure 1: Bibliothèque NAOqi et les modules de contrôles disponibles

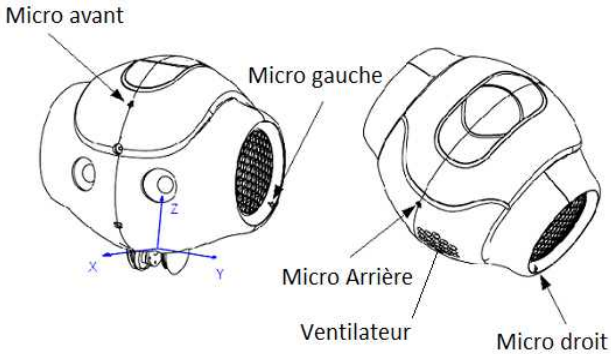

Figure 2: Positionnement des microphones et haut-parleurs sur le NAO

Les fonctions du module ALAudioPlayer permettent d'envoyer des données aux 2 haut-parleurs. De plus, la balance entre ces derniers peut être réglée. Les performances en termes de temps de chargement des fichiers audio à envoyer aux haut-parleurs dépendent de l'extension des fichiers utilisés. Les fichiers .wav non compressés sont ceux pour lesquels le temps de chargement est le plus court.

Des choix ont donc été faits pour que le temps de traitement audio soit acceptable. Les sons sont enregistrés en utilisant une seule voie (microphone gauche) échantillonnée à 16kHz d'extension .wav, et les fichiers envoyés aux haut-parleurs sont d'extension .wav.

Par ailleurs, la qualité des microphones du NAO est peu satisfaisante. Cela se répercute sur la qualité des fichiers audio enregistrés. En plus de cela, le ventilateur placé à l'arrière de la tête du robot génère un bruit stationnaire qui détériore la qualité de l’acquisition.

* Université Pierre et Marie Curie – Paris-Sorbonne Université

[1] Aldebaran-Robotics est une entreprise française fondée en 2005 par Bruno Maisonnier.

B. *Suppression du bruit généré par le robot*

Le bruit stationnaire généré par le ventilateur du robot est caractérisé avec un enregistrement de son sur le microphone gauche (car le moins bruité) dans le silence d'une chambre anéchoique. Cet enregistrement est utilisé pour générer une “clef” du bruit du ventilateur. Pour ce faire, dans un code python, la transformée de Fourier à court terme est effectuée sur le fichier sonore enregistré, et une plage du spectrogramme obtenu est sélectionnée comme “clef spectrale” du bruit. A ce spectrogramme est ensuite soustrait la “clef” du bruit généré par le ventilateur (Fig. 3) [7]. A la suite de cette soustraction, les valeurs négatives sont fixées à zéro, afin d'améliorer le filtrage. Enfin, la transformée inverse est appliquée et le fichier audio est enregistré au format .wav. Puisque le bruit du ventilateur est stationnaire, la “clef” est calculée et enregistrée une fois, et peut être réutilisée pour tous les traitements futurs.

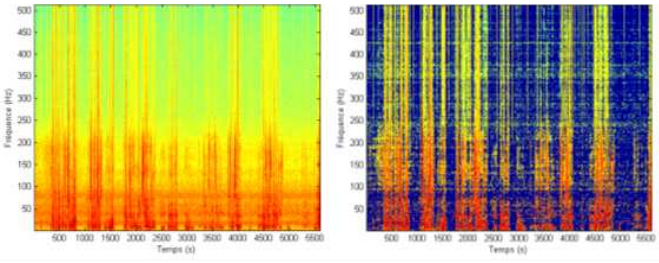

Figure 3: Spectre du signal sonore avant (gauche) et après (droite) traitement.

C. *PyOracle – improvisation musicale*

La partie improvisation a été effectué avec le logiciel PyOracle, développé par Greg Surges. Il est intégralement codé en python et peut-être utilisé seul. Cela le rend avantageux par rapport à d'autres logiciels comme Omax qui possède de nombreuses dépendances avec des logiciels sous licence. Son implémentation en Python le rend facilement utilisable avec le NAO. Néanmoins, il ne permet pas d'improvisation en temps réel. Le fonctionnement global du logiciel se décompose en 3 étapes :

- Traitement du fichier son en entrée.
- Génération de l'oracle (Fig. 4).
- Génération de l'improvisation.

Pour une optimisation du processus de traitement, le fichier d'entrée doit avoir certaines caractéristiques notamment concernant le format : .wav , mono et échantillonné à 16000Hz.

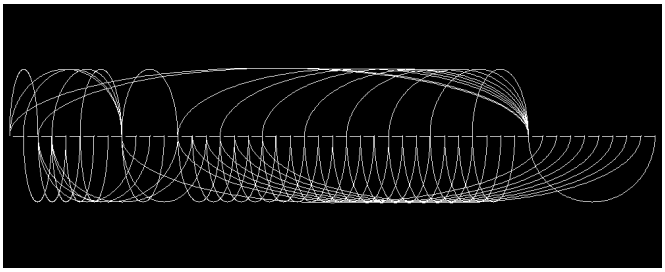

Figure 4: Oracle générée à partir d'un morceau FunkBass suivant le mode "chroma"

D. *Aubio – beat tracker*

Le beat-tracker consiste à déterminer la position des temps musicaux dans un signal audio de musique. L'algorithme de beat-tracking repose sur un algorithme classique de détection d'attaque dans un signal audio, suivant l'hypothèse qu'un temps musical est nécessairement marqué par une pulsation d'énergie. Pour ce faire : le signal audio est tout d'abord découpé en trames de 6 secondes, l'enveloppe temporelle d'énergie est extraite à l'intérieur de la fenêtre d'analyse, puis le tempo moyen et la position exacte des temps musicaux sont déduits par auto-corrélation sur l'enveloppe temporelle d'énergie. Le tempo moyen sur la trame d'analyse correspond au maximum de la fonction d'auto-corrélation, et la position des temps musicaux est déduite à partir du tempo estimé. L'utilisation de trames d'analyses de 6 secondes permet en outre les variations de tempo à l'intérieur d'un morceau de musique.

III. IMPLÉMENTATION

L'implémentation se décompose en trois grandes parties qui peuvent être représentées en blocs fonctionnels visibles sur la suite (Fig. 3).

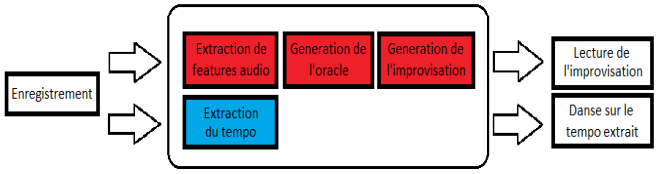

Figure 5: Processus de traitement de l'acquisition sonore à la restitution via les hauts parleurs du NAO.
Rouge → PyOracle ; Bleu → Aubio ; Blanc → NAO.

A. *Plateforme – NAO : Implémentation des réactions NAO*

Pour pouvoir intégrer le logiciel sur le robot, il faut avant tout pouvoir s'y connecter. Afin que plusieurs utilisateurs puissent s'y connecter en même temps, une connexion sans fil est préférable. Une première connexion filaire est nécessaire pour configurer le WiFi sur NAO. Une fois activé, on peut s'y connecter à distance à l'aide de son adresse IP en ssh.

Pour que le robot écoute une musique et danse lorsqu'il improvise, il faut exécuter des codes Python contenant les appels de fonctions adéquates. Il est ainsi possible de contrôler l'audio permettant l'enregistrement de son par ses 4 microphones et le mouvement répertoriant différentes danses possibles. La bibliothèque NaoQi déjà intégrée sur le Nao offre un large panel de fonctions qui nous permet de remplir le cahier des charges.

B. *PyOracle : Installation et utilisation du logiciel*

Il est possible, dans un premier temps, de compiler le logiciel sur une machine virtuelle de NAO, afin de simuler son bon fonctionnement et s'assurer de la faisabilité de l'installation. Malgré la nécessité d'installer quelques bibliothèques supplémentaires, la complétude du système virtuel par rapport au système réel a permis de faire

fonctionner PyOracle correctement. Par la suite deux options d'intégration sont possible :

- Solution embarquée sur le robot
- Solution de contrôle du robot à distance (externalisation des processus de calcul).

La solution embarquée requiert l'installation de bibliothèques non présentes dans le robot. L'espace mémoire disponible sur le NAO étant limité, la distribution Linux du Nao ne disposant pas des commandes habituelles telles que «apt» et les librairies standard du C étant corrompues, la solution embarquée n'a pas pu aboutir à ce jour.

Par conséquent la solution de contrôle à distance a été implémentée. Le robot enregistre un son et le transmet à un ordinateur externe. Ce dernier effectue les calculs de l'oracle lié à l'improvisation et envoie la piste audio résultante au robot. Il peut ainsi jouer l'improvisation et danser en rythme grâce aux informations récoltées par le beat-tracker.

C. *Beat-tracker : Installation et utilisation du logiciel*

L'utilisation du beat-tracker Aubio se fait depuis un terminal via la commande “aubiotrack”. Il faut donner en entrée le fichier source et indiquer le nom du fichier de sortie souhaité.

```
aubiotrack -i input.wav -o output.wav -f
```

L’argument -f force l’écriture de *output.wav* s’il existe déjà. Cette commande retourne un fichier audio. Afin d'obtenir une valeur de beat moyen, il existe une autre commande “aubioquiet”. Elle permet d’afficher les temps en secondes lorsqu'un son commence et s’arrête.

```
aubioquiet -i output.wav > output.txt
```

Nous donnons en entrée le fichier *output.wav* de la première commande de manière à récupérer les temps marquant les beats. Nous avons une sortie qui ressemble à ceci :

```
NOISY: 1.834376
QUIET: 1.869206
NOISY: 2.461315
QUIET: 2.496145
…
```

Il est alors possible de récupérer les intervalles entre deux beats correspondant aux délais entre « QUIET » (fin d'un beat) et « NOISY » (début d'un beat) permettant d'obtenir la valeur moyenne entre deux beats. Cette moyenne équivaut au tempo moyen.

IV. SCENARIO D'UTILISATION

L’utilisation du robot musicien s’articule autour de deux étapes :

- la phase d’écoute.
- la phase d’improvisation

La méthode de "contrôle à distance" du robot ayant été choisie comme méthode d’implémentation pour l'improvisation, la tâche principale est d'assurer l'acquisition audio vers un ordinateur pour la phase d’écoute.

Le code Python utilisé met NAO en attente d’un signal sonore avant de démarrer la procédure d’enregistrement de son. Lorsque le signal sonore est déclenché, le robot se met en position d’écoute (la main sur son oreille gauche) et dit "I am listening". Dans cet état, il enregistre pendant une durée déterminée par l'utilisateur et reste immobile afin que le son ne soit pas bruité par les actionneurs. En effet, les actionneurs des différentes articulations produisent un bruit non stationnaire. Il est plus compliqué à traiter que le bruit du ventilateur. A la fin de l'enregistrement l'ordinateur reçoit le fichier audio à l’aide de la commande scp appelée depuis le programme Python. Le robot reste toujours dans la même position.

Une série de traitements commence alors sur l'ordinateur. Tout d'abord, le fichier audio est filtré afin de supprimer le bruit du ventilateur. Puis l'estimation du beat est calculée. Cette estimation est le temps moyen entre deux beats. Finalement, l'improvisation musicale est réalisée avec le son filtré et un nouveau fichier au format .wav est créé sur l'ordinateur. Toujours avec la commande scp, le robot NAO reçoit ce fichier. Ceci marque la fin de la chaîne de traitement.

La phase d'improvisation n'est donc pas en temps réel. A la fin de la réception du fichier sonore d'improvisation, le robot quitte la position d'écoute et commence à jouer l'improvisation. En parallèle, le robot danse en utilisant des mouvements qui ont été au préalable enregistrés dans un programme Python. De plus, l'estimation du beat est passée en argument à l'appel de la fonction de danse pour donner la vitesse des mouvements. Ceci assure la synchronisation du robot sur le beat de la musique.

V. EXPÉRIMENTATIONS

Le robot ayant à suivre des rythmes variés lorsqu’il danse, il est nécessaire de savoir à quelle vitesse les mouvements doivent être limités. En considérant les spécifications des deux types de moteurs utilisés et de leurs réducteurs respectifs, on obtient des vitesses maximales de rotation à vide allant de 4rad/s à 9rad/s. Or, cela ne correspond pas à la réalité puisque chaque moteur actionne ici un membre du robot de masse non nulle.

La fonction utilisée pour effectuer les mouvements de danse interpole une trajectoire entre 2 postures qui ont été créées au préalable. L’interpolation de la trajectoire prend comme argument un temps. Puisque ce temps correspond à l’information renvoyée par le beat-tracker, il est alors plus simple d’imposer au robot une limite inférieure pour ce temps. Ainsi, une limite de 0,5 seconde pour passer d’une posture à une autre a été choisie. Lorsque le beat renvoyé par le beat-tracker est inférieur à 0,5 seconde, on double ce temps afin de garder la synchronisation avec le rythme de la musique. Cela permet d’éviter la surchauffe des articulations.

Au niveau logiciel, il est possible de jouer sur l’improvisation que génère l’oracle grâce un jeu de

paramètres. Des tests ont été effectués afin de détermier le jeu le plus polyvalent. Parmis ces paramètres trois sont plus significatifs que les autres :

- fft - fast fourier transform : il influe sur la taille du fichier de sortie et sur la qualité d'improvisation.
- continuité : caractérise la probabilité de saut d'une séquence à l'autre. Il est fixé à 0.75. La valeur maximale égale à 1 correspond à la séquence originale et la valeur minimale égale à 0 à une improvisation totale.
- lrs - low ratio share : il influe sur le lissage plus ou moins présent entre deux séquences.

Des tests ont également été effectué sur le beat tracker afin de déterminer les patterns qui sont le plus facilement détectés.

VI. Les expérimentations formelles à venir

A. *Mesurer la qualité d'écoute du robot*

L'improvisation musicale du NAO nécessite un fichier son d'entrée de bonne qualité. Pour compléter l'étude réalisée au cours de ce projet, il serait donc intéressant de mesurer la qualité d'écoute du robot. Pour cela, le calcul de la HRT (fonction de transfert relative à la tête - *head-related transfer function)* de la tête vide du robot (donc sans ventilateur) devra être réalisée dans une chambre anéchoique. Ceci rendra possible l'accès direct (sans passer par le CAN ni par des filtres de la chaîne d'acquisition) aux sorties analogiques des microphones.

De plus, en vue d'un filtrage du bruit des actionneurs du robot, un programme Python a été élaboré. Ce code permet de faire bouger toutes les articulations du robot, une par une, à trois vitesses différentes: minimale, moyenne et maximale.

B. *La qualité de l'improvisation du robot*

La seconde expérience à mener est de caractériser la dégradation de la qualité d'improvisation du logiciel en fonction de la qualité du fichier source. Pour cela il faut générer l'improvisation depuis un fichier original, non bruité. Il suffit de la comparer avec l'improvisation sur le même son mais enregistrer depuis NAO avec les mêmes paramètres de l'oracle. Deux types de comparaisons pourront être effectuées. Une comparaison purement empirique à l'oreille. Il s'agit ici de ne s'occuper que de l'improvisation, donc des sauts que l'oracle aura fait, et non de la qualité audio. Puis une comparaison beaucoup plus rigoureuse en comparant les graphes générés par l'oracle qui nous procure un visuel direct des sauts effectués par l'oracle. Ces comparaisons nous donneront une idée de la robustesse au bruit de l'oracle.

VII. Conclusion

Le travail réalisé a permis de développer un aspect de la robotique jusque-là mis de côté : l'improvisation audio. Pour pallier à cela, un logiciel d'improvisation musicale ainsi qu'un beat-tracker ont été implantés sur un robot humanoïde NAO. Ceci permet donc une interaction musicale entre le robot et un humain.

De plus, les bibliothèques fournis par NAOqi présentes sur le robot ont été utilisées pour la génération de mouvements. Ainsi, les improvisations du robot ont été agrémentées par des danses, synchronisées sur le rythme de la musique générée par NAO.

A long terme, ce travail pourra servir de base à une improvisation non plus musicale mais de dialogue avec des êtres humains. De plus, les danses exécutées par le robot pourront être elles aussi improvisées en fonction de caractéristiques extraites sur le son improvisé.

VIII. Remerciements

L'équipe de projet souhaite remercier Nicolas Obin pour son soutien, sa disponibilité et pour tous les moyens qu'il a mis à notre disposition tout au long du projet.

Nous souhaitons remercier S. Argentieri pour son aide et pour les locaux qu'il nous a fournis, M. Chetouani qui nous a fournis un robot de remplacement quand le notre est tombé en panne, G. Assayag pour le temps qu'il nous a accordé afin de nous expliquer les méthodes d'improvisation et le logiciel Omax.

Nous remercions la plateforme électronique et les informaticiens de l'université pour leur aide avec la connexion au robot, V. Meserette pour son assistance lors de la panne du robot, les intervenants de l'IRCAM qui nous ont aidé à plusieurs reprises, JL. Zarader pour nous avoir permis de déplacer le NAO et B. Chabouis pour son aide lors du tournage du film.

IX. References

[1] « Aldebaran Robotics », *Aldebaran Robotics*. [En ligne]. Disponible sur: https://www.aldebaran.com/en.

[2] G. Surges et S. Dubnov, « Feature Selection and Composition Using PyOracle », in *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.

[3] P. Brossier, J. P. Bello, et M. D. Plumbley, « Fast labelling of notes in music signals. », in *ISMIR*, 2004.

[4] P. M. Brossier, « Automatic annotation of musical audio for interactive applications », Queen Mary, University of London, 2006.

[5] M. E. Davies, P. M. Brossier, et M. D. Plumbley, « Beat tracking towards automatic musical accompaniment », in *Audio Engineering Society Convention 118*, 2005.

[6] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, et B. Maisonnier, « Mechatronic design of NAO humanoid », in *IEEE International Conference on Robotics and Automation, 2009. ICRA '09*, 2009, p. 769-774.

[7] Z. Chen, « Simulation of Spectral Subtraction Based Noise Reduction Method », *Int. J. Adv. Comput. Sci. Appl. IJACSA*, vol. 2, nº 8, 2011.