RÉCENTS DÉVELOPPEMENTS DU SPATIALISATEUR

Thibaut Carpentier
UMR 9912 STMS IRCAM-CNRS-UPMC
1, place Igor Stravinsky, 75004 Paris
thibaut.carpentier@ircam.fr

RÉSUMÉ

Le *Spatialisateur* est une suite logicielle développée à l'Ircam et destinée à la spatialisation sonore et à la réverbération artificielle. Cet article dresse un bilan des développements réalisés au cours des dernières années.

1. HISTORIQUE

Le *Spatialisateur* (communément appelé Spat∼) est un outil dédié au traitement de spatialisation sonore et de réverbération en temps réel. Il est développé à l'Ircam depuis le début des années 1990, et s'intègre à l'environnement *Max/MSP* [1]. Depuis son origine, le *Spatialisateur* a été conçu de façon modulaire et « scalable » [11, 16]. Il se présente sous la forme d'une bibliothèque de processeurs centrés autour d'un moteur de réverbération algorithmique [13] et de modules de panoramique. Cette modularité permet de configurer l'outil pour différents cas d'usage (concerts, mixage, post-production, réalité virtuelle, installations interactives) en l'adaptant 1) au dispositif de restitution — casque ou ensemble de haut-parleurs — et 2) à la puissance de calcul disponible.

Le *Spatialisateur* peut être contrôlé par le biais d'une interface de haut-niveau qui inclut des attributs perceptifs issus d'études psychoacoustiques [17, 18]. Cette interface permet à l'utilisateur de spécifier (et moduler) la qualité acoustique de l'effet de salle synthétisé d'une manière intuitive et indépendante du mode de restitution.

Du point de vue logiciel, le *Spatialisateur* fut tout d'abord développé comme un ensemble de *patches* et *abstractions* Max. Au cours des années 1990 et 2000, les modules les plus critiques (filtrage et réverbération) furent portés en code C sous forme d'objets externes. Toutefois cette architecture (abstractions Max et *externals* en C) n'offrait que peu de flexibilité pour configurer les modules du *Spatialisateur* : il était typiquement limité à 3 sources en entrée et une douzaine de canaux de sortie. Dans les années 2008-2010, le besoin est donc apparu d'entreprendre une refonte complète du développement. Celle-ci devrait en outre permettre de tirer profit des interfaces audio « massivement » multicanales et de la puissance de calcul désormais disponibles.

Cet article dresse un bilan des développements réalisés au cours des dernières années, en mettant l'accent sur les nouvelles fonctionnalités.

2. ARCHITECTURE LOGICIELLE

En 2008-2010, le code du *Spatialisateur* a été entièrement réécrit en langage C++ (et marginalement en objective-C). En effet, la puissance de la programmation orientée objet est bien adaptée à la conception modulaire et scalable du *Spatialisateur*. Les briques de traitement de signal les plus critiques sont optimisées et vectorisées (par exemple à l'aide du *framework* Accelerate [2] sous MacOS).

Le code source est réparti dans deux bibliothèques : l'une dédiée aux tâches de traitement du signal, l'autre dédiée aux interfaces utilisateurs. Ces deux bibliothèques sont largement *cross-platform*, indépendantes du logiciel hôte — i.e. elles ne dépendent nullement de *Max/MSP* —, et autonomes (la principale dépendance est le *framework Juce* [3] qui est employé pour le développement des interfaces utilisateurs).

Les bénéfices de cette complète restructuration sont multiples :

- les performances des algorithmes de traitement des signaux sont accrues (par rapport à la version implémentée en patches),
- la configuration des algorithmes est désormais complètement flexible : les nombres d'entrées (ou de sources) et de sorties des objets sont régis par des attributs *@numinputs* et *@numoutputs* (voir Figure 4) dont les valeurs varient typiquement de 1 à 512 en fonction des installations),
- le *Spatialisateur* n'est plus tributaire de *Max/MSP*, et il peut désormais se déployer dans de nouveaux environnements : plusieurs *bindings* ont été développés afin de répondre à différents cas d'usages (voir Figure 1). Des *externals Max/MSP* sont utilisés pour les applications de concert ou d'installations interactives. Des *plugins* [4] insérables dans des stations de travail audionumériques ont été

1. http://cycling74.com/

2. http://developer.apple.com

3. http://www.juce.com

4. Diffusés par la société Flux : http://www.ircamtools.com

réalisés pour les applications de mixage et post-production. Différents outils ont été intégrés à l'environnement *OpenMusic* [3] pour les aspects d'écriture et synthèse spatialisée. Enfin des *mexfunctions* sont générées dans l'environnement *Matlab*, et utilisées (en interne) à des fins de recherche et de simulations numériques.

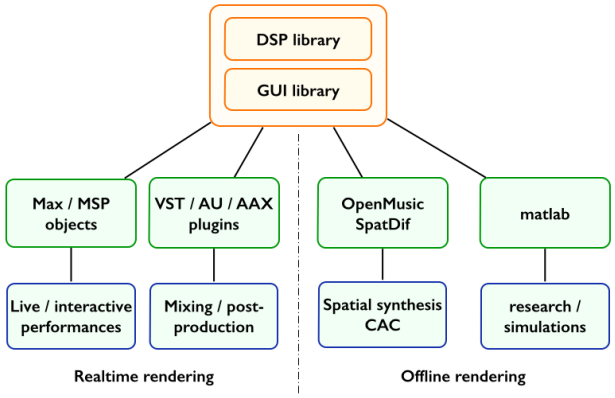

Figure 1. Déploiement du *Spatialisateur* dans divers environnements de travail.

Dans la suite de cet article, nous détaillerons essentiellement les nouveautés du *Spatialisateur* dans l'environnement *Max/MSP*. Cette incarnation demeure en effet la plus employée et elle intègre les innovations les plus récentes.

3. RÉVERBÉRATEUR

Le *Spatialisateur* s'articule essentiellement autour d'un moteur de réverbération algorithmique multicanal. L'effet de salle généré se compose de quatre sections temporelles (Figure 2) : le son direct, les réflexions précoces (*early*) générées par une ligne à retards avec plusieurs dates de sorties, un ensemble de réflexions tardives et plus denses (nommé *cluster*) créé par plusieurs lignes à retards alimentant une matrice de décorrélation, et enfin une queue de réverbération tardive synthétisée par un réseau de retards re-bouclés (*Feedback Delay Network* ou FDN) [13].

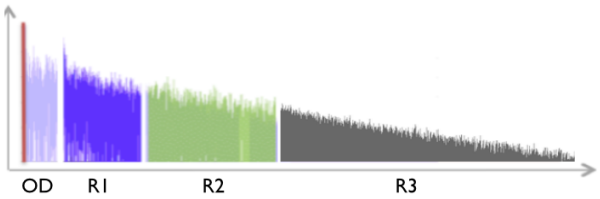

Figure 2. Archétype de réponse selon le modèle du *Spatialisateur*. OD : son direct. R1 : réflexions précoces. R2 : réflexions tardives. R3 : queue de réverbération.

Depuis la version 4.x du Spat∼, plusieurs évolutions ont été apportées à l'algorithme de FDN, par rapport à l'implémentation originale de Jot [13, 11] : le nombre de canaux de rebouclage — jusqu'alors limité à 8 — est étendu (typiquement on pourra utiliser 16 voire 32 canaux). Augmenter le nombre de canaux de rebouclage permet d'augmenter la densité modale et la densité d'échos, ce qui se révèle utile notamment lors de la synthèse de réverbérations longues (plusieurs secondes), ou pour traiter des sons percussifs avec des transitoires nets. Par ailleurs nous avons introduit dans le FDN un réseau de filtres simulant l'absorption du son dans l'air [2]. Ces filtres présentent un gabarit de type passe-bas qui rend la dépendance de la durée de réverbération plus conforme aux phénomènes observés dans une salle. Enfin, le contrôle du relief de décroissance (i.e. du temps de réverbération), originellement opéré sur deux ou trois bandes, peut désormais se faire sur un nombre quelconque de bandes de fréquence (voir notamment l'objet *spat.multiverb∼*).

Hormis ces évolutions, l'architecture globale du réverbérateur (et son articulation avec l'étage de *panning*) demeure la même que précédemment [11, 15], et nous donnons pour rappel un synoptique de la chaîne de traitement audio (Figure 3).

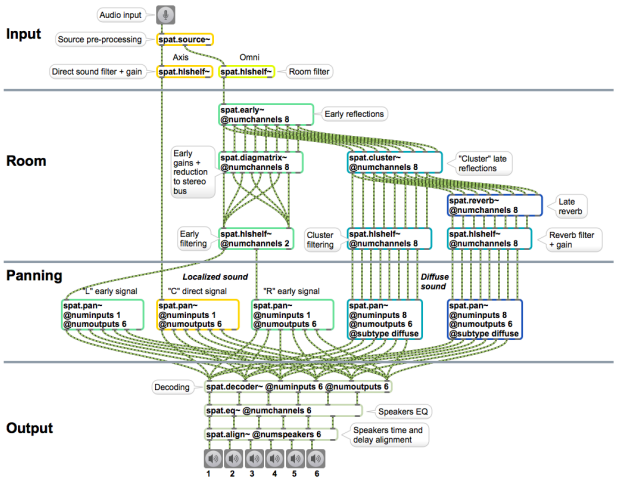

Figure 3. Architecture globale du *Spatialisateur*. Le module « Input » réalise les pré-traitements tels que filtrage de la source, effet Doppler, simulation de l'absorption par l'air. Le module « Room » synthétise l'effet de salle en quatre sections temporelles qui subissent chacune un filtrage indépendant. Ces quatre sections sont ensuite *pannées* selon un modèle temps-espace simple : les réflexions précoces R1 « entourent » le son direct (i.e. leurs directions d'arrivée sont réparties dans un cône dont l'axe est la direction du son direct, et dont l'angle d'ouverture est ajustable), tandis que les sections R2 et R3 sont spatialement diffuses. Enfin le module « Output » réalise le décodage des signaux en fonction du dispositif de restitution, l'égalisation des haut-parleurs et leur alignement en temps et en niveau.

4. PANNING

L'étage de *panning* du *Spatialisateur* est désormais implémenté dans un unique objet, spat.pan∼. L'objet est polymorphe, et, en fonction de ses attributs, se configure en termes de nombre de canaux d'entrées, de sorties, et type spatialisation à mettre en œuvre. Une vaste gamme d'algorithmes de spatialisation est supportée : techniques stéréophoniques conventionnelles (AB, XY, MS), synthèse binaurale, panning d'intensité — notamment : VBAP en 2D ou 3D [26], VBIP [14], DBAP [19], SPCAP [28] —, etc.

Le reste de cette section expose plus en détails certaines familles de panning qui ont fait l'objet des développements les plus significatifs au cours des dernières années.

4.1. Higher Order Ambisonics

Le *Spatialisateur* supporte l'encodage et le décodage au format *Ambisonic* de premier ordre (B-format) depuis de nombreuses années. Au début des années 2000, le paradigme *Ambisonic* a été étendu et formalisé [5] pour les ordres supérieurs (*Higher Order Ambisonics* ou HOA). Ces avancées ont été intégrées au *Spatialisateur* au cours des dernières années : l'objet *spat.pan∼* supporte désormais l'encodage HOA en 2D ou 3D, pour un ordre quelconque (tous les algorithmes sont implémentés de façon récursive). Les œuvres créées à l'Espace de Projection de l'Ircam [24] utilisent typiquement un encodage HOA 3D jusqu'à l'ordre 7, donc sur 64 harmoniques sphériques. Le « *Nearfield Compensation Coding* » (NFC-HOA, voir [6]) est également disponible, et des filtres passe-haut de compensation sont appliqués — en fonction de l'ordre — pour éviter les suramplifications en basses fréquences. L'objet *spat.hoatransform∼* permet quant à lui d'opérer des transformations dans le domaine de Fourier spatial ; il est par exemple possible de pondérer les différentes harmoniques sphériques de sorte à créer des effets de flou spatial (*blur*), ou d'appliquer des matrices de rotation 3D à la scène sonore. Enfin l'objet *spat.decoder∼* réalise les fonctions de décodage HOA, en 2D ou 3D. Plusieurs algorithmes de décodage sont proposés : outre les approches traditionnelles d'échantillonnage du champ harmonique à la position des haut-parleurs (*sampling decoder*) et d'appariement des modes d'excitation du champ sonore (*mode-matching decoder*), une nouvelle approche dite « à préservation d'énergie » [30] est disponible. Cette dernière préserve l'homogénéité de l'énergie et la largeur apparente de la source, même pour des dispositifs irréguliers de haut-parleurs (par exemple, hémisphère). spat.decoder∼ permet également d'opérer un décodage bi-bande, avec une fréquence de *crossover* ajustable, et d'appliquer des optimisations *in-phase* ou *max-re* dans chacune des bandes [5]. La Figure 4 présente l'architecture de travail standard pour création de scènes HOA de synthèse.

Enfin, divers outils utilitaires ont été développés pour simplifier l'usage et l'adoption de HOA au sein du *Spatialisateur*, et assurer sa compatibilité avec les nombreuses autres bibliothèques disponibles (e.g. [29]). Par exemple *spat.hoaconverter∼* permet de convertir les différents formats existants (N3D, SN3D, FuMa, etc.), *spat.hoasorting∼* ré-arrange l'ordre des canaux *Ambisonic* selon différents conventions, *spat.eigenencode∼* permet d'encoder en HOA les signaux issus d'un dispositif microphonique sphérique, etc.

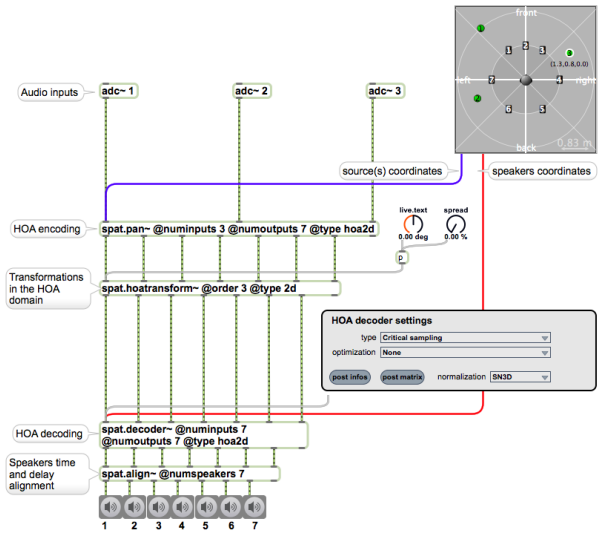

Figure 4. Architecture de traitement HOA. Dans cet exemple, 3 sources sont encodées à l'ordre 3 (en 2D) puis décodées sur 7 haut-parleurs.

4.2. Synthèse binaurale en champ proche

La synthèse binaurale/transaurale temps-réel est supportée dans le *Spatialisateur* depuis ses toutes premières versions [15]. Cette synthèse repose sur le filtrage des signaux sources par un jeu de HRTFs [5]. Les HRTFs sont traditionnellement mesurées pour un ensemble discret de positions situées à une distance fixe du sujet, dans le champ lointain. Toutefois on sait que les HRTFs sont significativement différentes pour des sources situées dans la région proximale du sujet, i.e. pour des distances inférieures à un mètre de la tête. Un cas d'usage typique est celui d'applications de réalité virtuelle dans lesquelles le sujet peut manipuler des entités sonores à portée de main. Des travaux ont donc été entrepris pour, d'une part établir une technique d'extrapolation radiale des HRTFs [25], et d'autre part étendre le moteur de synthèse binaurale (*spat.pan∼*) au cas de sources situées dans la région proximale. La solution retenue pour l'implémentation temps-réel utilise des filtres non-individualisés de correction d'ILD [6] [7] associées à une sélection croisée des filtres HRTFs et une correction géométrique des délais et gains monoraux [27].

5. Head-Related Transfer Function
6. Interaural Level Differences

4.3. TransPan

TransPan [1] est un nouvel outil, développé en collaboration avec le CNSMDP [7], et dédié (essentiellement) au mixage 5.1. L'enjeu de *TransPan* est de palier les faiblesses traditionnelles du dispositif 5.1, à savoir le manque de stabilité et de précision des sources latérales. Pour cela, *TransPan* combine plusieurs couches de spatialisation : 1) une couche de panning *surround* traditionnel (panoramique de temps ou d'intensité, et prise de son par des arbres microphoniques) et 2) une couche de traitement binaural/transaural [15] s'appuyant sur les paires de haut-parleurs frontale et arrière. La couche de traitement binaural/transaural permet d'augmenter la précision spatiale des sources latérales, voire de créer des sources en élévation.

L'objet *spat.transpan∼* est une station de mixage qui permet de combiner ces différentes couches de spatialisation. Le mix des couches peut se faire de façon manuelle ou automatique, sur la base de courbes de mélange empiriques qui visent à maximiser les effets spatiaux tout en minimisant les risques de coloration timbrale inhérente à l'usage de la technique transaurale. La Figure 5 illustre une vue de l'interface de contrôle de *spat.transpan∼*.

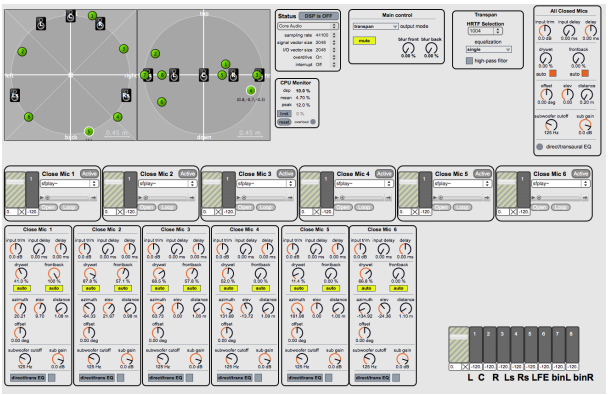

Figure 5. Une interface de contrôle de TransPan. Exemple avec six sources sonores.

5. INTERFACES UTILISATEUR

À partir de la version 4.x, la bibliothèque du *Spatialisateur* s'est enrichie de plusieurs interfaces de contrôle et de *monitoring* de la spatialisation.

spat.viewer (voir Figure 6) est un outil de visualisation et édition de scènes sonores. L'objet permet d'afficher et manipuler un nombre variable de sources et de haut-parleurs dans une représentation 2D (vue de dessus, de l'arrière, ou les deux côte à côte). L'apparence des éléments est largement configurable, par le biais d'attributs *Max/MSP*. Les positions des entités (sources ou haut-parleurs) sont contrôlables par un ensemble de messages *Max/MSP* ; la syntaxe est simple, lisible, et plusieurs systèmes de coordonnées sont supportés. En outre, de nombreux objets utilitaires sont fournis pour effectuer rapidement des opérations géométriques usuelles (translation, rotation, changement d'échelle, calcul de distance et de plus proches voisins, groupement de sources, conversion de coordonnées absolues vers relatives, etc.). Enfin, il est à noter que le *spat.viewer* n'est pas lié au moteur de rendu du *Spatialisateur*, et il peut être utilisé pour piloter d'autres processeurs de spatialisation.

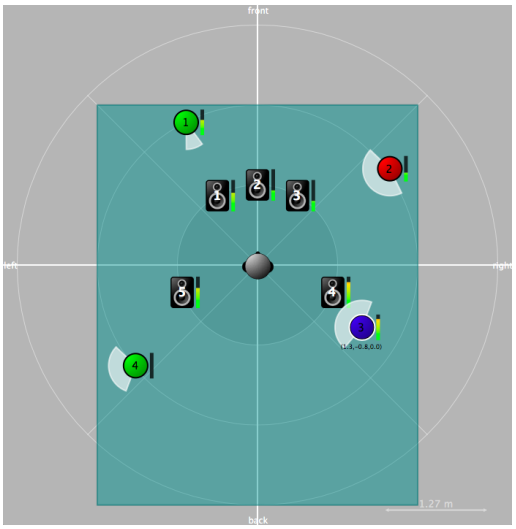

Figure 6. Interface du *spat.viewer* avec quatre sources sonores, cinq haut-parleurs et un auditeur, situés dans un environnement virtuel. Représentation en vue de dessus.

spat.oper (voir Figures 7 et 8) est l'interface de contrôle du *Spatialisateur* (i.e. de l'objet *spat.spat∼*). *spat.oper* est un « opérateur perceptif » qui implémente le contrôle de haut-niveau de la qualité acoustique de la salle [17, 18]. Il permet de naviguer le long de différents axes perceptifs, indépendant entre eux, et nommés « facteurs perceptifs ». Ces axes sont corrélés avec des critères d'acoustiques physiques, et sont au nombre de neuf ; trois d'entre eux caractérisent la source (présence, chaleur, brillance), trois caractérisent la salle (réverbérance, intimité, vivacité), et trois concernent l'interaction de la source avec la salle (présence de la salle, réverbérance courante et enveloppement). En modulant ces axes, il est possible d'interpoler continûment entre plusieurs situations acoustiques.

Outre les facteurs perceptifs, spat.oper permet de contrôler les filtrages (du son direct et du son envoyé vers la réverb), les caractéristiques de rayonnement des sources (orientation et ouverture du diagramme de rayonnement), la loi d'atténuation en distance, ainsi que diverses options. *spat.oper* se présente sous forme d'onglets : un onglet pour chaque source (Figure 8) et un onglet pour chaque salle virtuelle (Figure 7). Enfin *spat.oper* intègre un composant *spat.viewer* pour une

7. Conservatoire national supérieur de musique et de danse de Paris

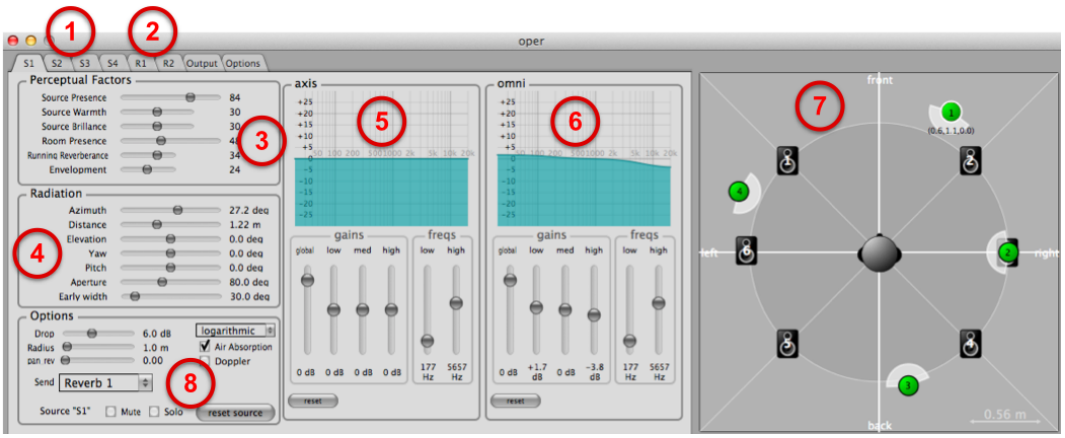

Figure 8. Vue du *spat.oper* : onglet source. ① sélection de l'onglet source. ② sélection de l'onglet réverb. ③ facteurs perceptifs de source. ④ localisation et directivité de source. ⑤ filtrage du son direct. ⑥ filtrage du son réverbéré. ⑦ visualisation de la scène sonore. ⑧ options de source.

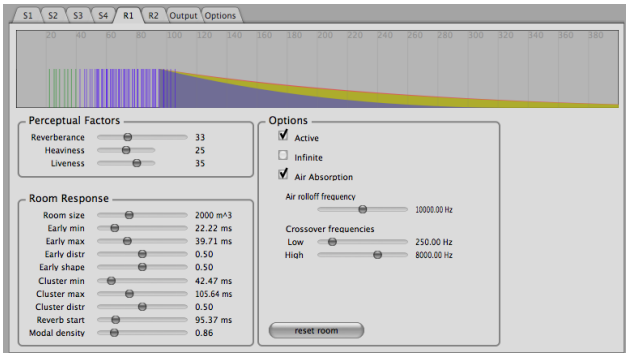

Figure 7. Vue du *spat.oper* : onglet reverb. Plusieurs réverbérations sont disponibles en parallèle, et chacune des sources est envoyée dans l'une ou l'autre de ces réverbs.

visualisation schématique de la scène sonore.

Plusieurs autres interfaces graphiques ont été développées (et sont constamment enrichies) pour le contrôle, l'*authoring* et le *monitoring* de la spatialisation. Les détailler dépasserait le cadre de cet article, aussi nous contentons-nous d'en donner un aperçu (Figure 9).

6. RÉPONSES IMPULSIONNELLES DE SALLE

Avec l'augmentation de la puissance de calcul des processeurs, l'utilisation de réverbérateurs à convolution est devenue plus courante dans les applications temps-réel (par exemple [10]). Aussi le *Spatialisateur* s'est-il doté d'une suite d'outils destinés à la manipulation de réponses impulsionnelles (IR) de salle. Cette section présente de façon sommaire ces outils.

6.1. Convolution (paramétrique)

spat.conv~ est un moteur (multicanal) de convolution basé sur une technique de FFT partitionnée par blocs [9], avec *overlap-save*. Le traitement des blocs à grande latence s'opère dans un *thread* d'arrière-plan afin de tirer profit des architectures multi-processeurs modernes. L'objet permet à l'utilisateur de faire un compromis entre la latence audio — possiblement nulle — et la charge computationnelle.

spat.converb~ est un autre processeur de convolution qui se base sur les mêmes algorithmes de filtrage rapide. En outre *spat.converb*~ segmente la réponse impulsionnelle en quatre sections temporelles (dont les durées peuvent être ajustées), à la manière du modèle du *Spatialisateur* (Figure 2), et un filtrage paramétrique est appliqué sur chacun des segments de l'IR. L'utilisateur peut régler les filtrages soit au bas-niveau (réglage des gains et fréquences de coupure), soit en appliquant l'approche perceptive du *spat.oper* [23].

6.2. Mesure acoustique de réponses de salles

spat.smk~ est un outil de mesure de réponses impulsionnelles de systèmes linéaires invariants, basé sur la méthode des sinus glissants [8]. Les caractéristiques du *sweep* (durée, plage fréquentielle, etc.) peuvent être paramétrées, et l'objet réalise la mesure — éventuellement multicanale — du système, la déconvolution des signaux, puis la sauvegarde des données. À l'issue d'une mesure, l'objet évalue différents critères (rapport signal à bruit, taux de distorsion harmonique, temps de propagation, temps de réverbération, etc.) afin de contrôler la qualité de la mesure.

spat.smk~ peut également être couplé à d'autres objets de calcul acoustique, tels que *spat.edc* qui calcule et affiche la courbe de décroissance énergétique, *spat.mixingtime* qui estime et affiche le temps de mélange d'une IR, ou encore

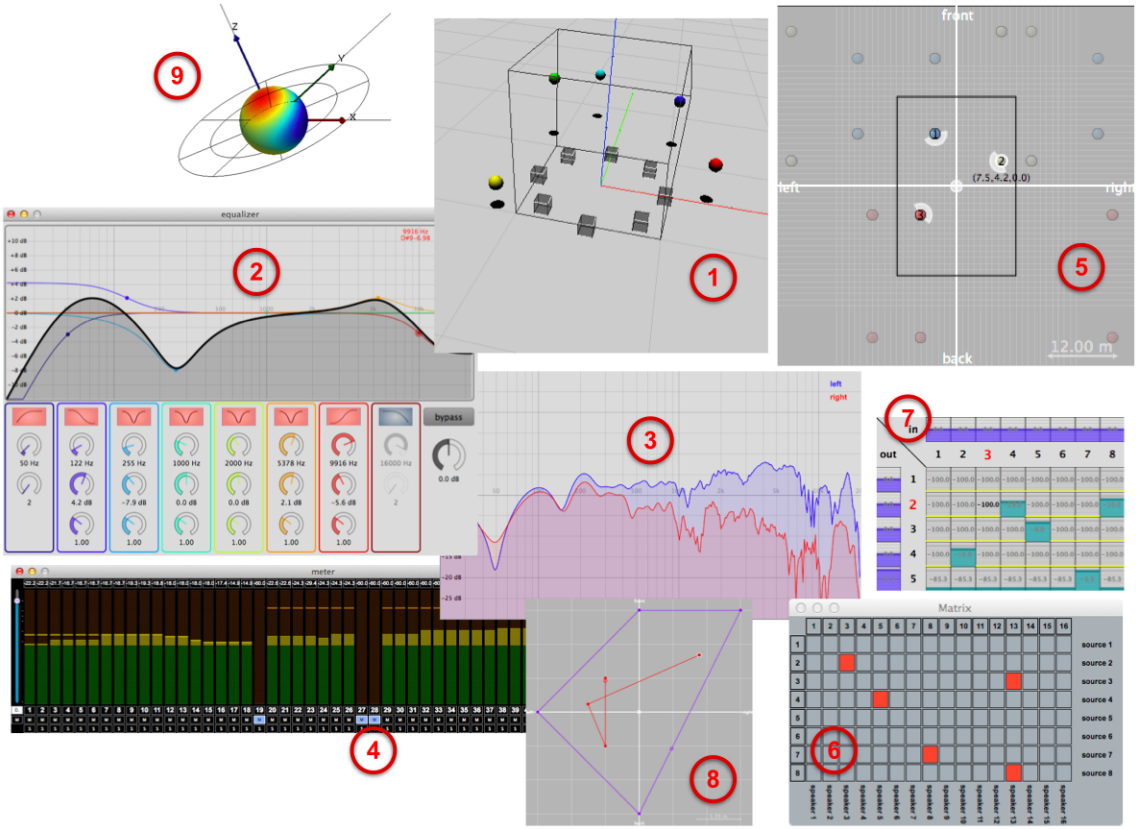

Figure 9. Quelques interfaces graphiques du *Spatialisateur*. ① interface *Jitter* pour le *spat.viewer*. ② égaliseur paramétrique. ③ visualisation de HRTFs (réponses en magnitude/phase). ④ vu-mètres multicanaux. ⑤ modèle géométrique de sources images. ⑥ matriçage binaire. ⑦ outil de matriçage. ⑧ édition de trajectoires spatiales. ⑨ visualisation 3D de champs sonores.

spat.ir.infos qui évalue de nombreux indices objectifs standards (clarté, temps central, *early decay time*, etc.).

6.3. Analyse de réponses de salles

Une analyse plus détaillée des IRs peut également être menée, à l'aide de l'objet *spat.rat* (Room Acoustics Toolbox). Cet objet implémente une méthode d'analyse temps-fréquence du relief de décroissance énergétique (densité d'énergie spectrale à chaque instant) [12]. À partir de cette analyse, l'objet déduit les caractéristiques de la décroissance : niveau de bruit, temps de réverbération et spectre de puissance initiale.

Cette analyse est utile à double escient : elle permet d'une part de débruiter les IRs mesurées (la partie bruitée est supprimée puis remplacée par prolongement du profil de décroissance) en vue de leur utilisation dans un moteur de convolution [12], et d'autre part d'estimer la valeur des facteurs perceptifs correspondant à cette IR. Autrement dit, l'objet permet à partir d'une IR mesurée d'obtenir un « preset » du spat.oper approchant les caractéristiques de cette IR.

6.4. Réverbérateur hybride

Les approches convolutives telles que *spat.conv*∼ (voir paragraphe 6.1) sont appréciées car elles préservent « l'authenticité » et la signature acoustique de salles réelles. Toutefois les moteurs de convolution restent difficiles à paramétrer (par exemple changer la durée de réverbération sans artefacts), et leur coût de calcul dépend de la durée de l'IR. À l'opposé, les processeurs purement paramétriques tels que les FDNs offrent un contrôle très flexible de la distribution temps-fréquence, ils sont performants et scalables ; cependant ils sont souvent critiqués en raison d'artefacts audibles (coloration, densités modale et d'échos insuffisantes) dans les réflexions précoces.

spat.hybrid∼ est un nouvel outil de réverbération qui adopte une approche hybride [4] : à partir d'une IR chargée, le processeur réalise la synthèse des réflexions précoces par convolution, tandis que la section tardive est remplacée par un réseau de retards rebouclés dont les caractéristiques sont automatiquement ajustées afin « d'imiter » le profil de décroissance de l'IR originale sans différence perceptible. Cette technique mixte bénéficie donc de l'authenticité de la convolution et de la flexibilité du FDN. L'objet peut de

plus être piloté par une approche de haut-niveau telle que *spat.oper* [4, 23].

7. AUTRES DÉVELOPPEMENTS

7.1. Outils de traitement multicanal

Le travail du son multicanal est un champ spécifique de l'informatique musicale dans lequel les utilisateurs sont souvent confrontés à des outils mal adaptés, même pour des opérations relativement basiques. C'est pourquoi la bibliothèque du *Spatialisateur* a été enrichie d'un certain nombre d'objets utilitaires simples, visant à accomplir efficacement des tâches usuelles. Sans souci d'exhaustivité, mentionnons ici quelques exemples :

spat.sfplay~ et *spat.sfrecord~* remplacent les conventionnels *sfplay~* et *sfrecord~* de *Max/MSP*. Toutefois ils permettent de gérer jusqu'à 250 canaux [8] et de lire/écrire des fichiers au format WAV RF64 qui permet de dépasser la limitation traditionnelle de taille de fichier à 4 Go.

spat.send~ et *spat.receive~* permettent de créer des bus d'envoi multicanaux.

spat.virtualspeakers~ permet de *downmixer* n'importe quel contenu multicanal en un flux stéréo binaural, en s'appuyant sur un paradigme de haut-parleurs virtualisés [22].

spat.align~ réalise un recalage (en délais et en gains) d'un ensemble quelconque de haut-parleurs, en fonction de leur disposition géométrique.

De même, *spat.delaycalibration~* et *spat.gaincalibration~* permettent de réaliser la calibration rapide et automatique d'un dispositif de haut-parleurs afin de les ajuster en gains et en délais. À la différence de *spat.align~*, ils utilisent une mesure acoustique réalisée *in situ*.

7.2. Effets divers

Il est traditionnel d'appliquer certains effets audio sur les canaux de sortie d'un dispositif de haut-parleurs. L'emploi des objets *Max/MSP* usuels se révèle généralement laborieux puisqu'ils fonctionnent (pour la plupart) en mono. La bibliothèque du *Spatialisateur* propose donc un grand nombre d'effets classiques, en version multicanale : *spat.compressor~* (compresseur/expandeur), *spat.eq~* (égaliseur paramétrique à base de filtres biquadratiques), *spat.graphiceq~* (égaliseur graphique avec nombre de bandes réglable), *spat.limiter~*, *spat.noisegate~*, *spat.softclipping~*, *spat.dcfilter~*, etc.

7.3. Support du standard SOFA

La distribution du *Spatialisateur* inclut un ensemble de données HRTFs pouvant être utilisés dans les modules de synthèse binaurale. À l'origine, ces données étaient stockées dans des fichiers « *coll* » de *Max/MSP*. Ce format, textuel, est peu flexible et ne permet pas l'inclusion de méta-données. Plus tard, un autre format de stockage — basé sur SDIF [9] — a été élaboré. Certes fonctionnel, ce format propriétaire était peu propice à l'échange avec d'autres institutions ou des logiciels tiers.

Dans les versions les plus récentes du *Spatialisateur*, ces formats (*coll* et SDIF) sont abandonnés au profit du *Spatially Oriented Format for Acoustics* (SOFA). SOFA [20] est un format pour le stockage et l'échange de données acoustiques, en particulier de HRTFs. SOFA est un format ouvert, extensible, auto-documenté, indépendant de l'architecture matérielle, supportant la compression, pouvant transiter sur réseau, et adopté en tant que standard par l'Audio Engineering Society [21]. Depuis 2013, le *Spatialisateur* intègre *libsofa* [10], une implémentation C++ du format SOFA qui permet donc de charger des HRTFs issues des nombreuses bases de données disponibles [11].

8. CONCLUSION

Dans cet article nous avons présenté des développements récents du *Spatialisateur*, depuis sa version 4.x, fruit d'une refonte logicielle complète. Les améliorations et nouvelles fonctionnalités portent sur tous les aspects de l'architecture : module de réverbération, module de panning, interfaces de contrôle. Au total, plus de 150 *externals* sont proposés dans la distribution actuelle [12].

Par ailleurs, les récentes recherches se sont axées sur l'analyse et le traitement de réponses impulsionnelles mesurées. Cela pose les bases d'une (possible) nouvelle architecture du *Spatialisateur* qui s'appuierait sur un moteur convolutif ou hybride et paramétrisé par les facteurs perceptifs.

9. REFERENCES

[1] Baskind, A., Carpentier, T., Noisternig, M., Warusfel, O., Lyzwa, J.-M., « Binaural and transaural spatialization techniques in multichannel 5.1 production », *27th Tonmeistertagung VDT International Convention*, Cologne, Allemagne, 2012.

[2] Bass, H., Bauer, H.-J., Evans, B., « Atmospheric absorption of sound : analytical expressions », *Journal of the Acoustical Society of America*, 52(3), 1972, p. 821 – 825.

[3] Bresson, J., Agon, C., Assayag, G., « OpenMusic. Visual Programming Environment for Music Composition, Analysis and Research », *ACM MultiMedia (OpenSource Software Competition)*, Scottsdale, USA, 2011.

[4] Carpentier, T., Noisternig, M., Warusfel, O., « Hybrid reverberation processor with perceptual control », *17th Int.*

8. Il s'agit de la limitation actuelle de *Max/MSP*.

9. Sound Description Interchange Format

10. `http://sourceforge.net/projects/sofacoustics/`

11. `http://hrtf.ircam.fr`

12. `http://forumnet.ircam.fr/product/spat/`

Conference on Digital Audio Effects (DAFx-14), Erlangen, Sept. 2014.

[5] Daniel, J., « Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia », thèse de doctorat de l'Université Paris VI, 2001.

[6] Daniel, J., Moreau, S., « Further Study of Sound Field Coding with Higher Order Ambisonics », *116th Convention of the Audio Engineering Society*, Berlin, Allemagne, 2004.

[7] Duda, R. O., Martens W. L., « Range dependence of the response of a spherical head model », *Journal of the Acoustical Society of America*, 104(5), 1998, p. 3048 – 3058.

[8] Farina, A., « Simultaneous measurement of impulse response and distortion with a swept-sine technique », *108th Convention of the Audio Engineering Society*, Paris, France, 2000.

[9] Gardner, W.G., « Efficient Convolution without Input-Output Delay », *97th Convention of the Audio Engineering Society*, San Francisco, États-Unis, 1994.

[10] Harker, A., Tremblay, P.-A., « The Hisstools impulse response toolbox : convolution for the masses », *International Computer Music Conference*, Ljubljana, Slovénie, 2012.

[11] Jot, J.-M., « Real-time Spatial Processing of Sounds for Music, Multimedia and Interactive Human-Computer Interfaces », *ACM Multimedia Systems Journal (Special issue on Audio and Multimedia)*, 7(1), 1997, p. 55 – 69.

[12] Jot, J.-M., Cerveau, L., Warusfel, O., « Analysis and Synthesis of Room Reverberation Based on a Statistical Time-Frequency Model », *103rd Convention of the Audio Engineering Society*, New-York, États-Unis, 1997.

[13] Jot, J.-M., Chaigne, A., « Digital delay networks for designing artificial reverberators », *90th Convention of the Audio Engineering Society*, Paris, France, 1991.

[14] Jot, J.-M., Larcher, V., Pernaux, J.-M., « A comparative study of 3-D audio encoding and rendering techniques », *16th Audio Engineering Society International Conference on Spatial Sound Reproduction*, Rovaniemi, Finlande, 1999.

[15] Jot, J.-M., Larcher, V., Warusfel, O., « Digital Signal Processing Issues in the Context of Binaural and Transaural Stereophony », *98th Convention of the Audio Engineering Society*, Paris, France, 1995.

[16] Jot, J.-M., Warusfel, O., « A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications », *International Computer Music Conference*, Banff, Canada, 1995.

[17] Jullien, J.-P., « Structured model for the representation and the control of room acoustic quality », *International Congress on Acoustics*, Trondheim, Norvège, 1995, p. 517 – 520.

[18] Kahle, E., Jullien, J.-P., « Subjective Listening Tests in Concert Halls : Methodology and Results », *International Congress on Acoustics*, Trondheim, Norvège, 1995, p. 521 – 524.

[19] Lossius, T., Balthazar, P., De La Hogue, T., « DBAP - Distance-Based Amplitude Panning », *International Computer Music Conference*, Montréal, Canada, 2009.

[20] Majdak, P., Iwaya, Y., Carpentier, T., Nicol, R., Parmentier, M., Roginska, A., Suzuki, Y., Watanabe, K., Wierstorf, H., Ziegelwanger, H., Noisternig, M., « Spatially Oriented Format for Acoustics : A Data Exchange Format Representing Head-Related Transfer Functions », *134th Convention of the Audio Engineering Society*, Rome, Italie, 2013.

[21] Audio Engineering Society, « AES standard for file exchange - Spatial acoustic data file format », New York, USA, 2015.

[22] Møller, H., « Fundamentals of Binaural Technology », *Applied Acoustics*, 36, 1992, p. 171 – 218.

[23] Noisternig, M., Carpentier T., Warusfel O., « Perceptual Control of Convolution Based Room Simulators », *Acoustics 2012*, Hong Kong, Chine, 2012.

[24] Noisternig, M., Carpentier, T., Warusfel, O., « Espro 2.0 – Implementation of a surrounding 350-loudspeaker array for sound field reproduction », *Spatial Audio in today's 3D World - AES 25th UK Conference*, York, Royaume-Uni, 2012.

[25] Pollow, M., Nguyen, K.-V., Warusfel, O., Carpentier, T., Müller-Trapet, M., Vorländer, M., Noisternig, M., « Calculation of Head-Related Transfer Functions for Arbitrary Field Points Using Spherical Harmonics Decomposition », *Acta Acustica united with Acustica*, 98, 2012, p. 72 — 82.

[26] Pulkki, V., « Virtual Sound Source Positioning Using Vector Base Amplitude Panning », *Journal of the Audio Engineering Society*, 45(6), 1997, p. 456 – 466.

[27] Romblom, D., Cook, B., « Near-field compensation for HRTF processing » ; *125th Convention of the Audio Engineering Society*, San Francisco, États-Unis, 2008.

[28] Sadek, R., Kyriakakis, C., « A Novel Multichannel Panning Method for Standard and Arbitrary Loudspeaker Configurations », *117th Convention of the Audio Engineering Society*, San Francisco, États-Unis, 2004.

[29] Sèdes, A., Guillot, P., Paris, E., « The HOA library, review and prospects », *International Computer Music Conference / Sound & Music Computing conference*, Athènes, Grèce, 2014.

[30] Zotter, F., Pomberger, H., Noisternig, M., « Energy-Preserving Ambisonic Decoding », *Acta Acustica united with Acustica*, 98, 2012, p. 37 – 47.