Sensitive Home Informatique

Colombel François, Hauchecorne Eric, Vandon Raphaël

29 juin 2009

Table des matières

1 Introduction

Alors que l'électronique embarquée est de plus en plus présente dans notre environnement, que ce soit dans les voitures ou dans l'électroménager, qui intègrent de plus en plus de fonctions avancées, ces nouvelles technologies permettent de plus en plus d'améliorer le confort en automatisant les tâches de notre quotidien. Nous avons pensé que cette amélioration du confort de l'utilisateur par l'electronisation de son environnement se devait de passer par l'amélioration du confort à son domicile. C'est pourquoi nous avons décidé de développer un projet de domotique, en axant nos effort en priorité sur l'éclairage et la chauffage, car ce sont les composantes les plus importantes du confort au domicile. De plus si l'amélioration du confort a été la priorité dans ce projet, nous avons aussi considéré le fait que la majorité de la population aimerait avoir un comportement plus écologique dans sa gestion de l'énergie, mais ne l'a pas par manque de motivation. Nous avons donc voulu, tout en maintenant, voir même en améliorant le confort de l'utilisateur, l'aider à optimiser la consommation de sa maison en électricité. Et nous avons voulu le faire en adaptant l'éclairage à ses besoin suivant l'éclairage extérieur, et en régulant le chauffage pour tenir compte de son absence et de ses horaires.

2 Présentation générale du projet

Le projet Sensitive Home est divisé en deux parties : informatique et électronique. Ces deux parties ont été développées conjointement par deux groupes distincts.

L'objectif du projet est de développer une solution de domotique qui soit simple à mettre en place pour l'utilisateur, et sans nécessiter de travaux de maçonnerie. Cette facilité d'installation est permise par l'utilisation de multiprises communiquantes sur lesquelles viennent se brancher les appareils de la maison, et qui relaient les informations venant de capteurs sans fil situés dans la maison (capteur de luminosité, capteur de température, ...). La conception de cette multiprise constitue la partie électronique du projet.

Le comportement de la maison est ensuite régi par des profils d'utilisation : on distingue des profils globaux, qui s'appliquent sur la maison dans son ensemble, et définissent des constantes telles que la température moyenne à maintenir, ou la luminosité globale dans la maison ; et des profils locaux, qui ne s'appliquent qu'à une seule pièce, prioritaires sur le profil global, et correspondant à une activité spécifique. Des profils globaux type seront "jour", "nuit", ou "absence prolongée", tandis qu'un profil local pourra être "lecture dans le salon" ou "travail dans le bureau". Le rôle de la partie informatique du projet est de permettre la création et la gestion de ces profils, ce qui est réalisé à travers une interface Web, adaptée à n'importe quel ordinateur, mais aussi à un iPhone ou un iPod Touch. Les changements de profil sont possibles également à travers une interface tactile qui est mise en place facilement sur n'importe quelle surface : murs, portes ou tables.

3 Architecture de la gestion de la maison

3.1 Architecture de classes

Le plus gros du travail d'architecture de classes a été fourni pour la mise en place des profils et la configuration de la maison. Nous avons utilisé une classe principale nomée *Maison*, qui nous permet d'accéder à toutes les informations du projet. La maison est modélisée en utilisant le design pattern Singleton, montré dans l'extrait de code 1.

Extrait de code 1 Singleton

```
private static Singleton singleton ;
//cet accesseur est le seul moyen d'obtenir l'unique instance de la classe
public static getSingleton()
{
  if(singleton == null)
    singleton = new Singleton() ;
  return singleton ;
}
//constructeur privé qui empêche l'instanciation depuis une autre classe
private Singleton()
{
}
```

La *maison* contient une liste de *Salles*, ainsi qu'une *liste de profils globaux* disponibles et le profil en cours d'utilisation. Chaque *Salle* contient une *liste de profils locaux*, le profil en cours d'utilisation, ainsi que la liste des *Multiprises* et des *modules de capteurs* de cette pièce. Une *Multiprise* contient un tableau de *Prises,* ainsi que des données relatives à la communication. Une *Prise* contient des informations sur son état et l'appareil qu'elle alimente.

Les classes Prise et Capteur se ressemble sur de nombreux points, nous avons donc écrit une classe abstraite dont ces deux classes héritent, et de même pour les classes Multiprise et Module de capteurs.

Les états d'une prise, ainsi que l'appareil branché dessus sont des membres d'une énumération.

Profils

À la base, un profil global devait contenir une liste de sous profils, permettant de paramétrer plus finement la maison, et chaque sous profil devait contenir une liste de Prises, ainsi que leur état dans ce profil, permettant de n'utiliser les lampes que d'un coin de la pièce par exemple. Malheureusement, pris par le temps, nous n'avons pas eu le temps de terminer l'implémentation des sous profils, et nous nous sommes limités à l'utilisation de profils globaux.

3.2 Asservissement

Sachant que nous ne recevons des informations des capteurs qu'à faible fréquence (toutes les 2 secondes), nous utilisons des méthodes d'asservissement rudimentaires. Pour les radiateurs, on ne dispose de toute façon que de deux états à contrôler : allumé ou éteint, qui sont directement contrôlés par les capteurs.

Pour la luminosité, les choses sont un peu plus compliquées : dans un premier temps, nous avions pris en compte le fait que chacune des lampes éclaire différemment la pièce, ce qui nous conduit à chercher comment obtenir la luminosité la plus uniforme possible dans la pièce (i.e. sur chacun des capteurs de luminosité). Si on suppose que l'on connaît l'effet que produit une légère incrémentation de tension de chacune des lampes sur l'ensemble des capteurs, on peut connaître la commande à envoyer en résolvant un système linéaire de la forme $Ax = b$, avec A une matrice contenant l'effet de chaque lampe sur chaque capteur, x la commande à déterminer, et b la luminosité souhaitée. Mais ce problème peut ne pas avoir de solution, si par exemple un capteur est dans une zone d'ombre. On va donc chercher à résoudre le problème au sens des moindres carrées, c'est à dire résoudre $A^T Ax = A^T b$. Cette méthode suppose que l'on connaisse l'effet des lampes sur les capteurs. Nous avons imaginé pour cela imposer une opération de calibration, pendant la quelle on met la pièce dans le noir, et les lampes sont allumées unes à unes. À chaque allumage, on retient les valeurs des capteurs en les associant à la lampe concernée.

Cependant plusieurs problèmes se posent : tout d'abord, l'effet des lampes sur les capteurs n'est pas linéaire, ce qui introduit une erreur dans le système. Ensuite, l'opération de calibration était difficile à intégrer de façon cohérente au projet, et enfin, ce système ne se corrige pas lui même par rapport aux nouvelles valeurs reçues.

Nous avons donc abandonné cette façon de faire pour en adopter une beaucoup plus simple : on considère les capteurs comme un tout, dont on ne cherche à asservir que la moyenne, et l'ensemble des lampes de la pièce également comme un tout, que l'on commande de façon uniforme. On envoie des commandes aux lampes en utilisant un pas variable, que l'on divise par deux chaque fois que l'on dépasse la valeur souhaitée : ainsi, si l'on monte trop fort, on va redescendre de la moitié. On arrête les corrections quand la luminosité moyenne est dans une zone de tolérance définie par une constante, à $\pm 10\%$ par exemple.

3.3 Problèmes rencontrés

Notre principal problème dans cette partie a été que nous avons chacun élaboré une architecture sensiblement différente dans notre coin avant d'en parler ensemble, et il en a résulté des incompatibilités tout au long du projet, nous obligeant parfois à revenir en arrière et nous faisant perdre du temps. Cela est du à un manque de communication entre nous au moment de définir cette architecture, et nous en avons conclu que nous aurions du dès le début du projet nous mettre d'accord dessus, et utiliser un diagramme UML pour représenter clairement notre pensée ; ce que nous essayerons de faire dès notre prochain projet.

4 Interface Web

Afin de configurer et contrôler le reste du projet nous avons mis en place une interface Web. Nous avons fait ce choix plutôt qu'une interface logicielle standard par soucis d'accessibilité des contrôles : une interface logicielle aurait permis de contrôler la maison uniquement depuis le serveur, tandis qu'avec une interface Web il devient possible de la commander non seulement depuis le serveur, mais aussi depuis un autre ordinateur du réseau local, ou même par internet depuis le travail où un lieu de vacances. De plus avec l'avènement nouveaux téléphones portables intégrant un navigateur internet comme notamment l'iPhone, une interface Web offre la possibilité de transformer son téléphone en télécommande pour la maison.

4.1 Technologies utilisées

Côté client

FIG. 1 – Interface côté client avec clip Flash

Pour que l'interface Web soit la plus accessible possible et consultable sur les différents appareils et navigateurs, côté client nous avons utilisé du simple HTML généré et mis en page par une feuille de style en CSS. Pour avoir un rendu plus dynamique de l'état de la maison qui puisse se mettre à jour en temps réel sans que l'utilisateur n'ai a envoyer de nouvelles requêtes manuellement, nous avons utilisé un clip en Adobe Flash (voir figure 1) qui communique avec le serveur par l'intermédiaire d'un fichier XML.

Côté serveur

Pour réaliser une interface logicielle en HTML il nous fallait utiliser un langage capable de le générer dynamiquement. Il existe trois plates-formes permettant de le faire : PHP, le framework .NET via l'ASP où le Java via le JSP ou les Servlets. Le tout le reste du projet étant en Java, que ce soit la communication avec la multiprise ou la gestion des profil, il apparaissait logique de conserver le même langage et de réaliser l'interface Web en Java aussi. Il fallait donc employer Java Edition Entreprises (anciennement J2EE) car la version standard ne gère pas les technologies Web. Enfin en utilisant du Java il nous restait encore deux options : le JSP, qui ressemble à une page HTML dans laquelle est intégré du code Java dans des balises, ou les Servlets, qui sont des classes Java permettant de gérer des requêtes HTTP et d'écrire du HTML via la sortie standard. De ces deux solutions nous avons choisit d'utiliser les Servlets car ils sont eux mêmes des classe Java et pouvaient à ce titre s'intégrer plus facilement dans le reste du projet et communiquer directement avec les autres classes du projet et d'échanger des données avec elles.

Serveur Web

Pour pouvoir déployer cette interface Web en Java nous avons installé un serveur Apache Tomcat qui permet de diffuser des projets Java Web sur un serveur Apache.

4.2 Architecture de l'interface

Une classe principale nommée Interface est la base de l'interface Web c'est elle qui reçoit toutes les requêtes de l'utilisateur et les redistribue aux autres classes de l'interface Web. Il s'agit aussi de la classe de déploiement du projet Web, c'est à dire qu'elle est chargée de créer et stocker toutes les données

communes au projet, que ce soit la configuration de la maison, ou les profils de température et d'éclairage pour la maison.

La classe CommunHtml contient différentes fonctions permettant de générer des bouts de code HTML que l'on retrouve fréquemment dans les page de l'interface, comme par exemple l'en tête de la page Web. La classe CommunHtml reçoit aussi les requêtes HTTP de la classe Interface et, en fonction de d'une variable "page" passée dans la requête, effectue un `switch` qui permet de générer la page que l'utilisateur à demandée.

Enfin une classe Formulaires génère les différents formulaires HTML qui permettront à l'utilisateur de configurer et commander la maison. L'appel à ces formulaires, qui dépendent de la page choisie, est fait dans le `switch` décrit précédemment dans la classe CommunHtml.

4.3 Fonctions de l'interface

Un menu commun à toutes les pages permet à l'utilisateur de sélectionner la page de l'interface qu'il souhaite parmi les différentes sections :

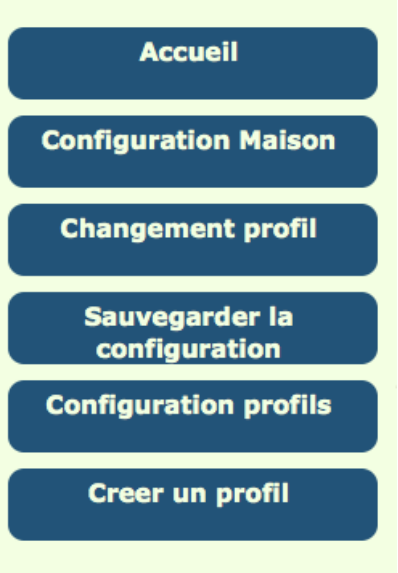

FIG. 2 – Menu

Accueil

Cette page (voir figure 1 page précédente) permet de connaître en temps réel l'état de la maison. Un clip Flash affiche une représentation de la maison et informe l'utilisateur de la température et de la luminosité dans chaque salle. Il va lire ces informations dans un fichier XML nommé etat.xml qui est mis à jour par le moteur principal de l'interface à chaque fois qu'une nouvelle données donnée de capteur est reçue. Si le navigateur ne gère pas le Flash, les informations sont affichées en format texte avec le nom des salles et les informations correspondantes.

Configuration de la maison

Cette page propose un formulaire qui permet de configurer la maison : il permet d'ajouter des salles et de les nommer, de répartir les différents modules de capteurs et multiprises dans les salles, de définir le type des capteurs sur les modules de capteurs et les appareils branchés à chaque multiprise .

Changement de profil

Permet de changer rapidement de profil à l'aide d'un simple menu déroulant.

Sauvegarde de la configuration

Sérialise toutes les données du projet dans des fichiers XML, qui seront ensuite désérialisés lors du prochain lancement du projet. Cela permet de relancer le serveur exactement dans l'état dans lequel il a été arrêté.

Configuration d'un profil

Permet de sélectionner un profil et de changer la température et la luminosité voulue dans ce profil.

Création d'un profil

Permet de créer un profil en donnant la température et la luminosité souhaité dans ce profil. Il est de plus possible de cocher une option «bouton» qui, si elle est sélectionnée, permet de d'assigner un bouton de l'interface tactile à ce profil. L'utilisateur n'a alors qu'a taper un nombre défini de fois à l'endroit où il veut créer ce nouveau bouton pour l'étalonner, et il devient lié au nouveau profil.

4.4 Interface iPhone

Pourquoi une interface mobile ?

Nous avons pensé que l'interface Web n'était pas suffisante pour le projet, car même si elle rendait le contrôle accessible sur n'importe quel ordinateur du réseau, s'il fallait aller sur un ordinateur à un chaque fois que l'utilisateur veut modifier la luminosité de la salle, le projet n'apporterait pas vraiment de confort face à un simple interrupteur. C'est pourquoi nous avons pensé que l'utilisateur pourrait avoir en permanence sur lui une télécommande pour contrôler sa maison, cependant cela lui imposait de se surcharger d'un appareil de plus en permanence et l'obligeait à investir dans une télécommande spécifique. C'est pourquoi nous nous sommes demandés : "Pourquoi ne pas contrôler la maison avec son téléphone ?". Les utilisateurs l'ont en général toujours sur eux et les téléphones récents offrent un navigateur Web qui serait capable d'exploiter notre interface. Mais pour que cette utilisation reste ergonomique il fallait développer une interface dédiée à un appareil mobile, qui n'a ni le même écran, ni les même contrôles qu'un ordinateur classique.

Pourquoi l'iPhone

Pour le projet, nous avons décidé de concentrer le développement sur l'iPhone et son petit frère l'iPod touch, qui utilise le même OS et le même navigateur, pour plusieurs raisons : premièrement le navigateur Safari intégré permet d'utiliser des fonctionnalités Web avancées ; ensuite l'interface tactile de l'iPhone est ergonomique pour la navigation Web sur ce type d'appareil, et enfin c'est l'un des téléphones de nouvelle génération le plus répandu sur le marché. De plus, plusieurs membres de l'équipe ont un iPhone ou un iPod Touch, ce qui facilite grandement les tests et les démonstrations.

Réalisation de l'interface dédiée

L'objectif était de minimiser la duplication de code au niveau du serveur pour faciliter la maintenance, c'est pourquoi les deux interfaces, iPhone et classique, utilisent le même code HTML généré par le serveur, ce qui permet de n'avoir aucune fonction spécifique à la plateforme client dans le serveur Java. Tout se fait lors du chargement de la page : le type d'appareil se connectant est détecté, s'il s'agit d'un iPhone ou d'un iPod touch, on utilise une feuille de style CSS différente. Ceci permet de changer complètement la façon dont s'affiche le HTML généré par le serveur, et de l'adapter à l'iPhone.

Différences de l'interface iPhone

Pour rendre la navigation plus facile sur un appareil avec un petit écran, le menu occupe toute la largeur de la la page, nous avons retiré le logo du projet, qui prenait trop de place, et les différents formulaires se retrouvent sous le menu. La taille des polices est optimisée pour faciliter la lecture sur un écran d'iPhone. Enfin, l'iPhone ne gérant pas le Flash, nous avons remplacé le clip sur l'écran d'accueil par un simple texte décrivant l'état de la maison.

4.5 Pistes d'amélioration

AJAX

L'interface Web étant une interface logicielle reportée sur une interface HTML, il apparaît intéressant d'exploiter les possibilités de l'AJAX et du Web 2.0, qui permettent justement de créer des applications

Web asynchrones avec les requêtes de l'utilisateur. Cela rendrait l'interface plus fluide, donnerait encore plus l'illusion d'utiliser un logiciel dans son navigateur, et permettrait en prime de limiter les transferts de données entre le client et le serveur au strict nécessaire. L'utilisation d'AJAX apparaît donc naturelle si l'on devait poursuivre le développement du projet, mais cela est légèrement plus lourd à mettre en place, et pas indispensable au fonctionnement ; c'est pourquoi cela n'a pas été notre priorité dans un premier temps.

Compatibilité des appareils mobiles

Pour s'adresser vraiment à tout le marché, il faudrait élargir l'interface pour iPhone à l'ensemble des appareils mobiles intégrant un navigateur internet. Cependant il faudrait pour cela prendre le temps de créer toutes les feuilles de style CSS spécifiques à chaque navigateur, et idéalement, disposer d'un appareil de chaque type pour tester le résultat ; c'est pourquoi nous ne l'avons pas fait dans l'état actuel du projet.

5 Interface tactile

FIG. 3 – Société sensitive Object

Nous pensions au début du projet utiliser un produit de la société Sensitive Object, qui vend des boîtiers prêts à l'emploi utilisant cette technologie, mais la société était en phase de développement d'un nouveau produit, et a refusé de nous vendre l'ancien. Nous avons donc provisoirement abandonné cette idée, mais plus tard nous avons rencontré un scientifique, Stephan Catheline, qui avait travaillé au développement de cette technologie, et nous a assuré qu'il était tout à fait possible de la mettre en place nous-mêmes en se basant sur l'article scientifique joint à ce rapport. Nous nous sommes donc mis au travail, et l'idée a été réintégrée au projet.

5.1 Principe physique

La localisation du point de contact avec la surface se fait à partir du son que produit ce contact. Après avoir enregistré le son produit par un contact sur chacun des points à reconnaître, on compare un nouveau son avec chacun des sons enregistrés, et celui auquel il correspond le plus correspond au point touché. Cette technologie est basée sur le retournement des ondes sonores.

Capture des données

Les ondes sonores se comportent comme des paquets de particules, et si les bords de la surface observée sont suffisamment chaotiques, toutes les "particules" de l'onde seront passées dans une zone donnée au bout d'un temps suffisamment long (appelé temps de Heisenberg), éventuellement après un grand nombre de réflexions sur les bords. Si on place un capteur dans cette zone, on va recevoir des informations venant de tous côtés du point source, et on a virtuellement entouré ce point de capteurs. Le décalage entre le premier pic, correspondant à l'onde reçue par le trajet direct, et les pics suivants, correspondant aux ondes réfléchies, ainsi que le nombre de paquets reçus par réflexion sont une signature unique du point qui a été tapé sur la surface, et permettent de le distinguer.

Si le milieu dans lequel on étudie les sons est trop dissipatif, il se peut que des paquets d'onde soient complètement absorbés avant d'avoir atteint le capteur, et on perd alors des capteurs virtuels sur un coté de la surface, ce qui peut affecter le contraste de façon plus ou moins gênante. Pour corriger cela, on

peut ajouter des capteurs (réels) à d'autres emplacements, pour essayer d'intercepter les paquets dans une autre zone avant qu'ils ne disparaissent. C'est pourquoi nous utilisons deux micros plutôt qu'un dans le projet.

Calcul de la corrélation

La corrélation d'un point avec un autre est un nombre indiquant combien les deux points sont similaires. Ainsi, la corrélation sera maximale entre deux points identiques, et nous l'espérons, plus faible ailleurs.

Considérons une boite noire dont la réponse impulsionnelle est $h(t)$, sa sortie pour une entrée $e(t)$ est $h(t) \otimes e(t)$ (convolution temporelle). Si on admet qu'un coup sur la surface tactile est assimilable à un dirac, alors on peut définir la réponse impulsionnelle $h_A(t)$ de la surface entre le point A et le capteur, qui sera le signal reçu par le capteur. Or un résultat de la théorie du signal nous dit que pour des entrées normalisées, le maximum en sortie d'un filtre de réponse impulsionnelle $h(t)$ est obtenu pour une entrée $h(-t)$. Ce maximum correspond au moment où les deux signaux sont en phase et se recouvrent parfaitement.

Donc lorsque l'on reçoit un nouveau son ($s(t)$), si on le retourne temporellement ($s(-t)$) et qu'on le convolue à tous les signaux connus ($s(-t) \otimes h_i(t)$), le signal obtenu présentera un pic pour chaque bouton, vu que les signaux sont sensiblement les mêmes, mais l'amplitude de ce pic sera maximale pour le point correspondant à ce son. L'amplitude du pic principal de la convolution calculée est la corrélation de l'origine du son avec le point i. Avec cette méthode, on est donc capable de déterminer quel point enregistré est le plus proche du point tapé. Pour ne pas réagir à des bruits parasites, on peut imposer une corrélation minimum pour prendre en compte le résultat.

Explication physique : Convoluer le signal retourné temporellement par $h_i(t)$, qui est la réponse impulsionnelle de la table, équivaut à renvoyer virtuellement à travers la table l'onde vers sa source : on fait défiler le temps à l'envers, et on observe le signal que l'on a émis avec le coup au point i. Le pic principal sera d'autant plus grand que l'on est virtuellement près de la source, et on aura évidemment un maximum sur la source elle-même.

La précision maximale de cette technique est imposée par la limite de la diffraction à $\frac{\lambda}{2}$.

5.2 Mise en application

Capture du son

FIG. 4 – Capteur piézoélectrique

Pour enregistrer le son, on utilise des capteurs piézoélectriques (MuRata PKS1, voir figure 4), ce qui permet d'écouter uniquement les vibrations de la table sur laquelle ils sont plaqués, et pas les bruits extérieurs (un capteur piézoélectrique émet un courant proportionnel à la déformation qui lui est appliquée). En réalité, on reçoit quand même les bruits extérieurs, mais avec une amplitude faible par rapport aux bruits propagés dans la table. Pour vraiment isoler les micros de l'environnement, nous utilisons de la pâte à modeler, ce qui permet en même temps de les fixer sur la table.

Pour écouter l'entrée son, nous nous sommes basés sur le code d'exemple fourni par Sun pour les application audio en Java[1]. Le code Java utilisé pour la capture audio est présenté dans l'extrait de code

[1]Code source disponible à l'adresse : http ://Java.sun.com/products/Java-media/sound/samples/JavaSoundDemo/

2, sans tenir compte des exceptions et des tests de compatibilité. La ligne 1 définit les paramètres de capture. Pour le projet nous avons utilisé du son échantillonné à 44,1 kHz et représenté sur 16 bits, ce qui est la meilleure qualité que l'on peut obtenir sur les cartes son standards. Nous avons besoin de cette qualité pour bien différencier les signaux venant des points de la surface, car les écarts entre les sons réverbérés sont minces (d'autant plus que le son se propage plus rapidement dans les solides). À la ligne 2, on récupère l'entrée micro de l'ordinateur, les lignes 3 et 4 ouvrent la ligne en écoute, et à la ligne 5, on récupère dans un tableau d'octets (bytes) les données brutes venant de la carte son.

Extrait de code 2 Capture du son

```
AudioFormat format = new AudioFormat(AudioFormat.Encoding.PCM_SIGNED, sampleRate, sampleSizeInBits,
channels, (sampleSizeInBits / 8) * channels, sampleRate, true);
line = (TargetDataLine) AudioSystem.getLine(new DataLine.Info(TargetDataLine.class, format));
line.open(format, line.getBufferSize());
line.start();
line.read(data, 0, bufferLengthInBytes);
```

Après avoir reçu les données de la carte son avec `line.read`, il reste quelques traitements à appliquer sur l'échantillon : Sachant que les données sont récupérées octet par octet alors que l'amplitude est échantillonnée sur 16 bits, il faut concaténer les octets reçus deux à deux (et obtenir ainsi des données sur 16 bits i.e. 2 octets) ce qui est illustré par la figure 5. Par ailleurs, sachant que nous récupérons les données de deux capteurs en "stéréo" sur la même entrée, les données alternent entre l'un et l'autre, et il faut, après avoir appliqué le premier traitement, les dissocier pour les placer dans deux tableaux différents. Dans le cas de données "mono" (un seul canal), aucun traitement n'est nécessaire.

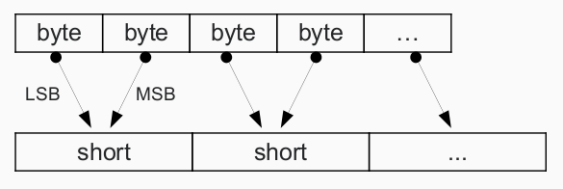

FIG. 5 – Concaténation des données

Pour détecter le signal qui nous intéresse (un choc), nous aurions pu utiliser un algorithme pour repérer une forte variation du signal, mais il est plus simple et plus rapide de simplement se fixer un seuil d'amplitude en deçà duquel les sons ne sont pas pris en compte. Pour déterminer ce seuil, nous avons capturé le son pendant 1 seconde avec du bruit ambiant pour se fixer une limite inférieure (le bruit ambiant ne doit pas être pris en compte), puis nous avons capturé le bruit d'un choc, ce qui nous donne une limite supérieure pour le seuil. Pour résumer, on fixe le seuil de détection tel que :

$$\text{bruit ambiant} < \text{seuil} < \text{amplitude maximale d'un coup}$$

Calcul de corrélation par FFT

Après avoir capturé le son d'un choc sur la surface, on va chercher à calculer sa corrélation avec les sons connus (qui est une opération de convolution temporelle). Un calcul direct de convolution temporelle est effectué par un ordinateur en $O(n^2)$, mais nous pouvons améliorer cette complexité en passant par la représentation fréquentielle des signaux, obtenue par transformée de Fourier : en effet, une convolution temporelle correspond à une multiplication fréquentielle, et le calcul d'une transformée de Fourier rapide (FFT) est opéré en $O(n \lg(n))$ quand n est une puissance de 2 (ce qui permet de procéder par dichotomie en séparant les échantillons pairs et impairs), et la multiplication est effectuée composante par composante, donc en $O(n)$, puis la transformée de Fourier inverse est identique à la FFT, et la recherche du maximum est en $O(n)$. La complexité de l'opération est alors dominée par la FFT, en $O(n \lg(n))$.

On vient de voir que la FFT est plus rapide si n est une puissance de 2, mais notre signal capturé n'a pas forcément une longueur qui est une puissance de 2. Pour corriger cela, on va effectuer un zero padding sur le signal, c'est à dire rajouter une série de zéros à la fin du signal jusqu'à atteindre la longueur voulue. Par ailleurs, on ne cherche à comparer que l'aspect du signal, sans se préoccuper de sa puissance, on

normalise donc la transformée de Fourier du signal, ce qui nous permet de plus d'avoir une corrélation en pourcentage (1 correspondant à 100%).

l'ordre des opérations est donc le suivant :

zero padding $\longrightarrow$ FFT $\longrightarrow$ normalisation $\longrightarrow$ multiplication $\longrightarrow$ FFT^{-1} $\longrightarrow$ recherche du maximum (pic)

et les 3 dernières opérations sont répétées pour chaque son connu. La recherche du maximum de corrélation peut être effectuée à la volée.

L'algorithme de calcul de la FFT est très courant, et en tant qu'ingénieurs, nous sommes sensés nous appuyer sur l'existant pour l'améliorer et l'utiliser dans un contexte nouveau, et pas réinventer la roue à chaque projet. Nous n'avons donc pas implémenté le calcul de FFT nous mêmes, mais l'avons trouvé sur internet[2], parmis les multiples sources disponnibles.

Remarque : Avec certains micros et/ou cartes son, le signal subit un offset (le silence ne correspond pas à la valeur zero), et il a donc une composante continue, qui va se traduire par un pic autour de 0 Hz sur sa transformée de Fourier. Or cette composante ne nous intéresse pas, ce sont les variations du signal que nous comparons. Pour ne pas prendre en compte cette composante continue, on peut soit commencer par soustraire à chaque échantillon la moyenne temporelle du signal, soit ne prendre en compte les échantillons qu'à partir d'une certaine fréquence (100 Hz par exemple).

Problèmes rencontrés

J'ai commencé à développer le projet avec une idée fausse sur la convolution : je pensais calculer la corrélation en sommant le produit des deux FFT ($\sum_i fft_1(i) * fft_2(i)$), alors qu'il faut non pas sommer, mais effectuer une FFT inverse puis rechercher le maximum pour avoir la corrélation. Car la convolution de deux signaux n'est pas un scalaire mais bien un signal temporel. Mon calcul de corrélation était donc biaisé par les lobes secondaires qui apparaissent normalement sur le signal convolué, et dont la puissance peut être importante. Malgré tout, cette méthode fonctionnait, bien qu'avec un contraste (différence entre la corrélation au point source et celle sur les autres points) plus faible sur les résultats. Nous l'avons laissé en place pendant tout la durée du projet, car je me suis aperçu de mon erreur vers la fin de la période de développement, et nous n'avons pas voulu modifier un code fonctionnel et prendre le risque de casser des choses à quelques jours de la présentation. Cependant, je compte implémenter ceci à titre personnel après la fin du projet.

Nos autres problèmes sont venus du découpage et de la concaténation des données suivant le mode de capture utilisé (16 bits, stéréo...), car ils étaient mal documentés dans l'application de démonstration de Sun, et nous avons mis quelque temps pour nous comprendre tout seuls comment étaient organisées les données venant de la carte son. Nous avons utilisé Scilab pour pouvoir afficher et manipuler facilement les données. En affichant les données sur la sortie standard, puis en la redirigeant dans un fichier, ou peut facilement les importer dans un vecteur scilab à l'aide de la fonction fscanfMat qui prend en paramètre un nom de fichier.

Pistes d'amélioration

temps d'exécution La principale piste à suivre pour améliorer le temps d'exécution du programme serait de réduire le temps de capture au temps de Heisenberg (durée pendant laquelle tous les paquets d'ondes passent par un micro), mais je ne sais vraiment pas comment l'évaluer pour une surface donnée. On pourrait essayer de s'en rapprocher en procédant par réductions successives du temps de capture en vérifiant que le contraste reste correct. En dehors du temps de capture, l'algorithme est peu gourmand en ressources, mais on peut l'améliorer tout d'abord en utilisant un langage plus proche de la machine, comme le C, et en travaillant sur l'algorithme de la FFT[3] (ou en utilisant les implémentations optimisées disponibles en C, comme FFTW[4]).

[2] FFT en java : http ://www.ling.upenn.edu/~tklee/Projects/dsp/

[3] le site suivant décrit quelques pistes pour obtenir un algorithme de calcul de FFT plus efficace : http ://cnx.org/content/m12021/latest/

[4] http ://www.fftw.org/

précision Pour augmenter la précision de détermination des zones tapées, on pourrait commencer par réduire la bande passante d'analyse des signaux, pour éviter de prendre en compte du bruit à des fréquences plus élevées. Il faudrait pour ça effectuer une série de tests en mesurant la fréquence des signaux reçus, et en observant les résultats tout en réduisant la plage de fréquences de l'étude.

On pourrait aussi utiliser un algorithme d'apprentissage, qui permettrait de différencier les boutons non pas seulement à partir de leur corrélation, mais aussi à partir des corrélations des autres boutons. Ainsi, on pourrait différencier deux boutons très proches si leur corrélation avec un troisième bouton varie de façon significative. Il est aussi possible que cette piste ne mène à rien, sachant que nous n'avons effectué aucun test sur ce point.

On peut augmenter la précision de la FFT, en utilisant pour découper le signal une fenêtre un peu plus évoluée qu'une simple porte (ou créneau), qui abîme la FFT du signal en la convoluant par un sinus cardinal. La fenêtre de Blackman semble

5.3 Intégration au projet

Le dispositif présenté ci dessus permet de rendre n'importe quelle surface tactile avec quelques micros et après avoir calibré les boutons sur cette surface. Son principal avantage est qu'en plus d'être bon marché, le coût de cette technologie ne dépend pas de la taille de la surface à rendre tactile. De plus, il s'agit d'une installation rapide à mettre en place, facilement adaptable à de nombreuses configurations : un bouton peut être rajouté ou supprimé à n'importe quel moment, et adaptée à un environnement hostile : si on place les micros sous une table par exemple, ou peut renverser un verre dessus sans endommager le système ; elle est donc particulièrement bien adaptée à une solution de domotique.

Nous avons choisi dans notre projet de l'utiliser pour permettre un changement de profil dans une pièce, ou dans la maison. Cela équivaut à utiliser des interrupteurs intelligents, car contrôlant plusieurs dispositifs à la fois, modulables, et avec une occupation de l'espace nulle, car les surfaces restent utilisables pour d'autres usages. Dans une application commerciale, on peut imaginer utiliser des micros sans fil, permettant par exemple au cuisinier de contrôler sa gazinière et son four depuis la surface de sa chaise sans quitter ses convives.

6 Communication

6.1 Communication avec les interfaces Homme-Machine

HashMap partagée

Les informations concernant la maison sont stockées dans des HashMap partagées entre les différentes parties du projet (interface Web, interface tactile et communication). Une HashMap est une structure de données permettant d'associer une clé à un objet Java. Dans ce projet par exemple, une HashMap de salles associe le nom des salles et les objets Salle correspondant. On peut alors avoir accès à toutes les informations dépendant des salles (type et valeur des capteurs, type et valeur des prises) à partir de leur nom. Lorsqu'une nouvelle information est connue, qu'elle vienne de l'utilisateur par l'intermédiaire de l'interface Web ou bien par la réception de données provenant d'un module de capteurs, cette HashMap est mise à jour. On travaille donc en permanence avec des données à jour.

L'interface Web utilise la HashMap pour afficher les valeurs de température et luminosité actuelles de chaque pièce. Elle propose aussi un état à jour de la configuration de la maison (le nombre de multiprises, le nombre de modules de capteurs, leur situation, la configuration des multiprises).

Interface tactile

Quand on crée un profil via l'interface Web, l'utilisateur a la possibilité d'y associer une zone de l'interface tactile. Lorsque l'interface tactile reconnaît un bouton prédéfini, celle-ci envoie à la classe Interfacage un évènement, qui déclenche un changement de profil vers le profil associé au bouton tapé.

L'écoute de l'interface tactile est contenue dans la méthode run() d'un Thread. Les Threads sont des morceaux de programme pouvant être exécutés en parallèle. Ceci permet que les boutons associés à un changement de profil soient détectés bien que la communication avec la multiprise (envoi et réception de messages) fonctionne en parallèle.

6.2 Communication avec la multiprise

La connexion

La multiprise communique avec le serveur par l'intermédiaire d'un réseau de type ethernet. Ce réseau utilise le CPL (Courants Porteurs en Ligne) ce qui évite de faire traîner des fils supplémentaires. La multiprise possède un module XPORT de Lantronix. C'est un adaptateur série↔ethernet qui fonctionne comme une interface réseau classique : elle est identifiée par une adresse MAC dont le début est spécifique au constructeur, qui est toujours de la forme "00 :20 :4A", ainsi que par une adresse IP.

Un logiciel fourni par Lantronix est nécessaire pour se connecter une première fois au module XPORT en indiquant l'adresse MAC, quand il est dans sa configuration "sortie d'usine". Sans celui-ci il n'est pas possible d'y accéder, son adresse IP étant fixé par défaut à 0.0.0.0. Ce point a posé un problème car l'adresse MAC, normalement écrite sur le module lui-même, était effacée. Après une perte de temps non négligeable, l'adresse MAC a pu être récupérée par la liaison série.

Lorsqu'une multiprise est créée, par exemple lorsque l'on charge le fichier de configuration de la maison en mémoire, un Thread Communication est instancié avec en paramètre l'adresse IP correspondant à la multiprise traitée. On crée la connexion à l'aide du code présenté dans l'extrait de code 3.

Extrait de code 3 Ouverture d'une connexion

```
new Socket(adresse IP, port)
```

Une fois la connexion créée, une boucle va dérouler les étapes d'écoute et d'envoi des données entre le serveur et le XPORT :

1. Vérification de la taille de la queue de messages à envoyer
   Si elle n'est pas nulle, on envoie le message qui est en haut de la queue
2. On attend de recevoir un message ou que le temps d'écoute maximum fixé soit dépassé
3. On regarde si le dernier message reçu est un accusé de réception
   Si c'est le cas, on supprime le message qui est en haut de la queue
   Sinon si le temps d'écoute maximum est dépassé, on essaie de renvoyer le message qui est en haut de la queue

Cette boucle est exécutée continuellement et parallèlement au reste du programme puisqu'elle est contenue dans la méthode run() du Thread Communication. Une queue de type ConcurrentLinkedQueue est utilisée pour stocker les messages devant être envoyés à la multiprise.

Avec ce système, si jamais un message n'est pas reçu par la multiprise, il est renvoyé et aucune information n'est perdue.

Le protocole

Un protocole de communication des données a été défini en concertation avec l'équipe de la partie électronique. Les messages envoyés et reçus sont formatés de façon à être reconnus par les programmes communiquant (dans le serveur et dans le microcontrôleur de la multiprise).

Voici les différents types de messages définis. Tout message n'étant pas formaté de cette façon n'est pas traité :

"reçu" correspond au sens multiprise→serveur
"envoyé" correspond au sens serveur→multiprise
"0011xxxxx" représente l'adresse d'un module de capteurs où "x" est une valeur numérique

- `/0011xxxxx : : : : :NOUVEAU : : : :\` est reçu lorsqu'un nouveau module de capteurs est installé dans une pièce
- `/0011xxxxx\` est envoyé à la multiprise qui doit relayer les données du module 0011xxxxx
- `/ACK :0011xxxxx\` est reçu pour accuser réception de l'ajout d'un nouveau module de capteurs
- `/0011xxxxx :d0 :d1 :d2 :d3\` est reçu toutes les deux secondes et contient les données des quatre capteurs du module 0011xxxxx
- `/REQ :prise :valeur\` est envoyé pour changer la valeur d'une prise
- `/ACK :prise :valeur\` est reçu pour accuser réception d'une commande

Les messages sont découpés pour être envoyés octet par octet. Ils sont lus de l'autre coté du XPORT par un microcontrôleur utilisant une liaison série.

6.3 Reconnaissance de matériel

Une fonctionnalité importante du projet est la possibilité d'ajouter de nouveaux appareils sans nécessiter de modification dans l'installation. Ceci est possible par son architecture modulaire.

Des capteurs peuvent être ajoutés sur les modules de capteurs tant que de la place est disponible. Si ce n'est pas le cas, il faut ajouter un nouveau module de capteurs. Lorsqu'un module est ajouté et mis sous tension, celui-ci va envoyer un message dans les airs. Les multiprises avoisinantes vont le recevoir et le transmettre au serveur. Le premier message reçu va permettre d'associer le nouveau module à la multiprise et il va être ajouté dans l'interface Web pour que l'utilisateur puisse paramétrer dans quelle salle il se trouve.

Il est aussi possible d'ajouter une multiprise. Lorsque celle-ci est branchée sur le réseau, elle envoie sur tout le réseau un message DHCPDISCOVER contenant l'adresse MAC de la multiprise. Un serveur DHCP tourne sur notre serveur. C'est un logiciel qui permet d'attribuer une adresse IP à une interface détectée sur le réseau. Le serveur DHCP va recevoir ce message DHCPDISCOVER et l'écrire dans le fichier syslog du système. Ce fichier log est lu par notre programme, qui vérifie s'il s'agit bien d'une adresse MAC d'un module XPORT. Si oui, il va modifier le fichier de configuration du serveur DHCP afin d'y ajouter la possibilité d'attribuer une adresse IP à la nouvelle multiprise.

"n" représente le numéro du XPORT ajouté
"x" représente un chiffre hexadécimal

Extrait de code 4 Extrait du fichier de configuration du dhcp

```
host XPORTn {
  hardware ethernet xx :xx :xx :xx :xx :xx; fixed-address 192.168.0.n;
}
```

De cette manière, si un ordinateur ou toute interface réseau n'étant pas un XPORT est reliée au réseau, le serveur DHCP du serveur domotique n'en tiendra pas compte et ne lui attribuera pas d'adresse IP.

Comme pour les modules de capteurs, on ajoute ensuite la nouvelle multiprise au fichier de configuration de la maison et l'utilisateur pourra paramétrer cette multiprise dans l'interface Web.

7 Ce que nous n'avons pas eu le temps de développer

Tout d'abord, la finalité du projet est bien sur de gérer plus d'objets que simplement des radiateurs et des lampes : on peut imaginer intégrer un contrôle des stores électriques, travaillant de concert avec les lampes pour la régulation de la luminosité, des climatiseurs, un contrôle de l'ouverture des fenêtres pour réguler la température de la maison en fonction de la température extérieure, etc...

Avec plus de temps nous aurions pu synchroniser le changement de profils de la maison avec un calendrier Google Calendar. L'API disponible nous permettrait assez simplement d'utiliser l'interface en ligne ergonomique de Google pour changer automatiquement de profil. De plus il serait envisageable de synchroniser le calendrier des profils de la maison avec le calendrier professionnel et personnel de l'utilisateur,

et ainsi par exemple de couper le chauffage lorsqu'il est au travail, et de commencer à réchauffer la maison juste avant son retour. Cela permettrait d'allier confort et économie d'énergie.

Nous aurions aussi aimé utiliser des capteurs de présence à l'entrée des salle, toujours pour automatiser les changements de profil, ce qui aurait accru le confort pour l'utilisateur, ainsi que l'acpect écologique du projet : en sachant qu'il n'y a personne dans la pièce, on s'empresse d'éteindre de la lumière, voire de baisser la température d'un degré, et si quelqu'un arrive, la pièce se configure à ses attentes sans attendre d'ordres.

Dans notre vision finale du projet, nous aurions utilisé des micros sans fil pour placer des interfaces tactiles vraiment partout sans encombrement. Ces micros sans fils auraient pu communiquer avec les multiprises, à la manière des modules de capteurs, qui auraient relayé l'information vers l'ordinateur pour être traitée.

Toujours dans une optique écologiste, nous aurions pu intégrer une facette de sensibilisation de l'utilisateur, lui proposant de réduire la température du chauffage si le programme l'estime trop élevée, de couper la chaudière quand l'été arrive, et nous aurions même pu analyser les comportements de l'utilisateur et lui proposer de changer ses horaires pour profiter plus longtemps de la lumière du soleil par exemple.

8 Conclusion

Nous avons mis beaucoup de nous mêmes dans ce projet, parfois sans prendre assez de recul, mais globalement, nous sommes tous sortis assez fier de notre travail, et certains d'entre nous vont continuer le projet pour essayer de la mener à son terme. Ce projet, qui paraissait ambitieux à plusieurs professeurs, l'était peut être un peu trop, vu que la partie électronique n'était pas prête pour le Jour des Projets, mais nous a tous beaucoup appris, ce qui était finalement son but.

Nous avons commencé par planifier l'avancement du projet avec un diagramme de Gantt, que nous n'avons finalement pas vraiment respecté, mais le faire au début du projet nous a permis de bien voir les tâches à accomplir et d'avoir des idées claires sur le travail à réaliser.

Nous avons découvert et appris à utiliser le contrôle de version sous subversion (SVN) en utilisant les serveurs offerts par Google Code[5], ce qui nous a bien servi, surtout dans les derniers jours du projet pendant lesquels nous travaillions à peu près tous sur les mêmes classes dans lesquelles il restait des problèmes à régler.

Nous avons pu appliquer les enseignements de l'ESIEE avec les transformées de Fourier, l'analyse linéaire, même si nous ne l'avons pas utilisée au final, et bien sur les enseignement de programmation : Java et multithreading. Nous avons également pu réutiliser ce que nous avions acquis pendant le stage effectué en début d'année, par exemple les design patterns ; et aller plus loin dans les domaines que nous ne connaissions que partiellement comme les serveurs Web Apache, la génération de code HTML avec les Java Servlet, et les communications réseau.

Nous avons travaillé avec un gros groupe de 7 personnes, ce qui n'a pas toujours été facile, mais chacun a su trouver sa place et s'épanouir dans ce projet, et nous ressortons tous enrichis de cette expérience, avec de bons souvenirs en prime.

[5] http ://code.google.com/p/sensitive-home/

In solid localization of finger impacts using acoustic time-reversal process

Ros Kiri Ing and Nicolas Quieffin
Sensitive Object, 8 rue d'Anjou, 92517, Boulogne-Billancourt, France

Stefan Catheline[a] and Mathias Fink
Laboratoire Ondes et Acoustique, ESPCI, Université Paris VII, U.M.R. C.N.R.S. 7587, 10 rue Vauquelin, 75231 Paris cedex 05, France

(Received 26 May 2005; accepted 20 September 2005; published online 8 November 2005)

Time reversal in acoustics is a very efficient solution to focus sound back to its source in a wide range of materials including reverberating media. It expresses the following properties: A wave still has the memory of its source location. The concept presented in this letter first consists in detecting the acoustic waves in solid objects generated by a slight finger knock. In a second step, the information related to the source location is extracted from a simulated time reversal experiment in the computer. Then, an action (turn on the light or a compact disk player, for example) is associated with each location. Thus, the whole system transforms solid objects into interactive interfaces. Compared to the existing acoustic techniques, it presents the great advantage of being simple and easily applicable to inhomogeneous objects whatever their shapes. The number of possible touch locations at the surface of objects is shown to be directly related to the mean wavelength of the detected acoustic wave. © *2005 American Institute of Physics*. [DOI: 10.1063/1.2130720]

Draeger[1] experimented the time reversal invariance of acoustic waves in chaotic cavities. In this experiment, an ultrasonic pulse is sent from a transducer (the source) and propagates inside a silicon plate. The acoustic trapped energy is reverberated a very long time. The acoustic field, measured on a second transducer (the receiver) lasts hundreds of times the initial pulse duration. This long and complex field is then time reversed and re-emitted from the receiver. The acoustic field is shown to propagate back to the source, recreating the short initial pulse. This experiment, transposed in the audible frequency range, is shown to be an efficient localizing technique.

In the following interactive experiment, the source transducer used in the ultrasonic experiment is replaced by a mechanical impact at the surface of an object. Once the impact position is determined, any corresponding action is executed. For example, a virtual keyboard can be drawn on the surface of an object; the sound made by fingers when a text is captured, is used to localize impacts. Then, the corresponding letters are displayed on a computer screen.

In the following example, the sound propagates through a glass plate, with the dimensions $400\times300\times5$ mm^3. The receiver is a passive sensor (Murata PKS1-4A type), with a working bandwidth ranging from 0.1 to 5 kHz. The sensor is glued next to one side and connected to the input line of a personal computer, 700 MHz with a 256 Mo random access memory. Typically, 100 ms acoustic signals are digitized by a standard sound card, with a 44.1 kHz sampling rate, and a 16-bit dynamic. This signal can mathematically be expressed as a convolution product between the emitted signal $e(t)$ from the touched point P, and the impulse response to the sensor S, $h_{PS}(t)$:

$$S(t) = e(t) \otimes h_{PS}(t). \tag{1}$$

Since the touch is brief enough in consideration with the working frequency bandwidth, the emitted signal can be approximated to a delta function $\delta_P(t)$. Thus, Eq. (1) becomes:

$$S(t) \cong h_{PS}(t). \tag{2}$$

It means that the signal measured by the sensor is simply the impulse response between points P and S.

The first step of the experiment is a training step. At the surface of the plate, a 35×28 "tactile points" are chosen on the nodes of a 1 cm square grid, Fig. 1. These 980 tactile points P_i are sequentially knocked; $i=1,\ldots,980$ is defined as the point index. The corresponding impulse responses, $h_{PiS}(t)$, detected by the sensor are recorded in the computer.

In a second use step, as one of the previous points is knocked again, say P_j, the new impulse response $h_{PjS}(t)$, is transferred to the computer. Then, a time reversal experiment is processed in the computer. The new impulse response is time reversed and virtually reemitted by the sensor as if it

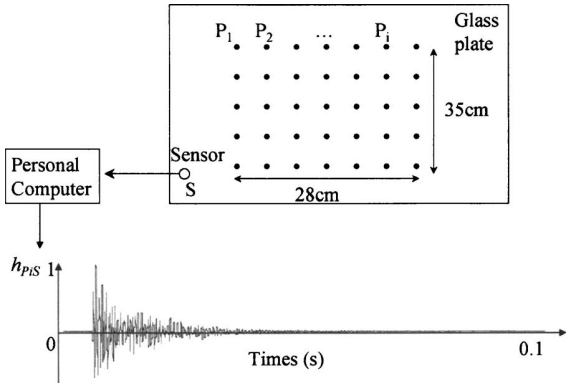

FIG. 1. Training step of a time reversal interactive experiment. Tactile points in a 35×28 matrix are sequentially knocked on a glass plate. Detected by a sensor S, 980 acoustic signals $h_{PiS}(t)$, are stored in the computer memory.

[a]Author to whom correspondence should be addressed; electronic mail: stefan.catheline@espci.fr

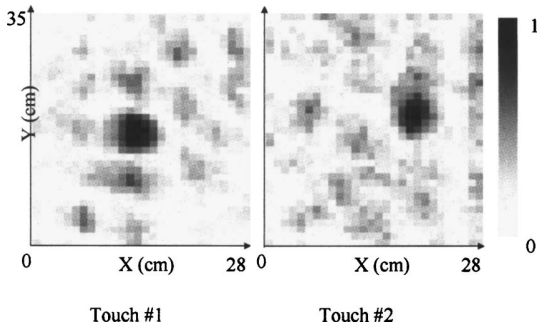

FIG. 2. During the use step, the signal from a new touch impact is correlated to the acoustic signals recorded in the training step. On the correlation map, the localization of two touch impacts clearly appears as black dots.

was able to act as a source. The acoustic field $S_i(t)$ that would be observed with sensors on points P_i is deduced from Eq. (1) where the emitted signal $e(t)=h_{PjS}(-t)$:

$$S_i(t) = h_{P\,j\,S}(-t) \otimes h_{P\,i\,S}(t). \tag{3}$$

The field $S_i(t)$ is entirely computed in software. It is indeed a correlation of impulse responses with a maximum when $i=j$. This is a consequence of the reversibility property of the wave equation which implies that a maximum of energy is found on the point where the source was. From another straightforward point of view, the correlation can be interpreted as a recognition processing that does identify the touched point among the reference signals.

This is illustrated on the normalized correlation map (i.e., the focusing acoustic pattern) represented on a grey scale, Fig. 2. Black dots indicate a correlation coefficient bigger than 0.95 whereas white areas correspond to a correlation coefficient smaller than 0.55. The technique can clearly localize the two touch impacts, moreover in real time since the whole computation in this example takes 20 ms.

Figure 3 represents the correlation coefficient along a line of Fig. 2 at $Y=17$ cm. The impact point is centered at $X=14$ cm. Such focusing patterns of acoustic waves involving time reversal technique have been described quantitatively in numerous papers.[2,3] Its −3 dB width and the maximum peak to ground level ratio are the two main characteristics. They define the resolution and the contrast respectively. The resolution is highly related to the question of the maximum number of tactile points. Indeed, it sets a boundary limit between points having different acoustic signatures with a correlation coefficient smaller than an arbitrary value chosen here at 0.7 (−3 dB). In other words, it defines a resolution surface.

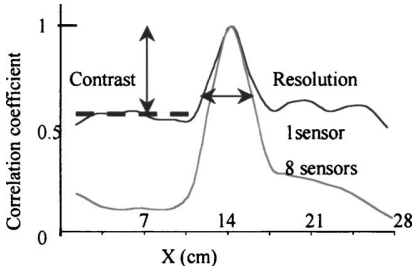

FIG. 3. Typical normalized correlation pattern centered on a touched point at 14 cm. Compared to a single sensor, using eight sensors mostly improve the contrast.

The acoustic energy mostly propagates as antisymmetric Lamb waves and more precisely as flexural waves for low value of the frequency-thickness product. In this experimental configuration, the flexural wave speed, absorption time, and wavelength at a 4 kHz frequency range are $\approx 230\ \mathrm{ms}^{-1}$, ≈ 25 ms, ≈ 5.7 cm, respectively. The latter figure is almost twice the −3 dB width observed in Fig. 3. This is in perfect agreement with the diffraction theory, stating that the resolution limit is reached when the −3 dB width equals half of the wavelength. Indeed, thanks to multiple reflections on boundaries (more than 30 in the plate experiment), there is a whole set of virtual sensors surrounding the focal spot and responsible for its shape (cylindrical symmetry) and its width (one-half of a wavelength). This observation is crucial since it states that the surface of resolution can be deduced from the flexural wavelength estimate λ_F. Theoretically, this latter value depends on the central frequency, the width, the transverse, and compression sound speeds of the plate. As a consequence, the maximum number of tactile points increases with the surface of the object (which is obvious) and with the frequency whereas it decreases with thickness. It may be noted that dispersive effects of the flexural wave improve the efficiency of the localization.[4]

The question of the number of receivers in time reversal experiments has been extensively studied on a wide class of multiple scattering media as well as in parallel steel rod immersed in water,[5] two-dimensional silicon wafer,[6] or reverberant room.[7] In order to investigate this question in time reversal interactive experiments, we used a Soundscape system (SS810-3 type). Simultaneous acoustic fields on different points (up to eight receivers) could be recorded. The contrast increases but the resolution remains unchanged by adding sensors. The contrast enhancement can be explained by the superposition principle: When N emitters focus their energy on the same point, the amplitude on the focal point is the sum of the amplitudes of each individual emitter. Since the resulting acoustic field of each individual emitter is uncorrelated, the amplitude increases according to the square root of the number of emitters everywhere else. The resulting contrast is the ratio of the amplitude on the focal point and the average amplitude elsewhere that is to say $N/\sqrt{N}=\sqrt{N}$. The reciprocity property of the wave equation stipulates that the same result applies to the number of receivers. Indeed, from Fig. 3, the contrast with eight sensors, $\cong 4.9$, is roughly $\sqrt{8}$ times better than the estimation of the contrast with one sensor, $\cong 1.8$. As far as the resolution is concerned, no improvement is expected in nicely reverberating objects. However, for weak reverberating media, the focal spot only partially surrounded by virtual sensors, may lose its cylindrical symmetry and be enlarged. Adding sensors in such situation can help to improve the resolution.

In this letter, a new acoustic time reversal technique has been proposed to transform most of everyday life objects as interactive interfaces. It appears that the number of tactile points is dependent on the mean wavelength of acoustic waves involved in the experiments. Since in numerous cases the acoustic energy propagates as flexural waves, the tactile points are centimeter sized, which is ideal for finger uses. A single receiver is sufficient for localizing impacts, but others can be added to improve the correlation map contrast.

Compared to time-of-flight techniques, the time reversal technique presents the great advantage of being efficient in most solids where sounds propagate, without any knowledge

of sound speed or receiver positions. Moreover, the technical and processing requirements make the experiment possible in real time with a basic personal computer.

This work was partially financed by the European FP6 IST Project "Tangible Acoustic Interfaces for Computer-Human Interaction (TAI-CHI)." The support of the European Commission is gratefully acknowledged.

[1]C. Draeger and M. Fink, Phys. Rev. Lett. **79**, 407 (1997).
[2]D. Cassereau and M. Fink, Acoust. Imaging **19**, 141 (1991).
[3]N. Quieffin, S. Catheline, R. K. Ing, and M. Fink, J. Acoust. Soc. Am. **115**, 1955 (2004).
[4]R. K. Ing and M. Fink, IEEE Trans. Ultrason. Ferroelectr. Freq. Control **45**, 1032 (1998).
[5]A. Derode, A. Tourin, and M. Fink, J. Acoust. Soc. Am. **85**, 6343 (1999).
[6]C. Draeger and M. Fink, J. Acoust. Soc. Am. **105**, 611 (1999).
[7]S. Yon, M. Tanter, and M. Fink, J. Acoust. Soc. Am. **114**, 3044 (2003).