MASTER EN SCIENCES INFORMATIQUES À FINALITÉ SPÉCIALISÉE EN INFORMATIQUE DES ORGANISATIONS

Eye tracking as a modality

état des lieux de l'interaction eye-based en vue d'une intégration aux interactions proxémiques

Lemal, Amélie

Award date:
2018

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2017–2018

Eye tracking as a modality : état des lieux de l'interaction eye-based en vue d'une intégration aux interactions proxémiques.

LEMAL Amélie

Maître de stage : GWIDZKA Jacek

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
DUMAS Bruno

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Remerciements

Avant toute chose, je souhaiterais remercier toutes les personnes qui ont contribué de près ou de loin à l'écriture de ce travail.

Je souhaite tout d'abord remercier mon promoteur, le Dr. Dumas, pour son implication, son aide et son soutien tout au long de la rédaction de ce travail, ainsi que pour m'avoir donné l'opportunité d'étudier un champ de l'informatique aussi intéressant que celui des interactions avec le regard.

Je remercie aussi mon maitre de stage, le Dr. Gwidzka, qui m'a permis de travailler à l'Univesité du Texas d'Austin, pour son accueil et pour m'avoir permis d'acquérir les connaissances théoriques nécessaires à la réalisation de ce mémoire en me laissant assister à son cours.

Je souhaite également remercier ma famille pour son soutien et la motivation qu'ils m'ont insufflée, et tout particulièrement ma maman, qui m'a aidée lors de la correction orthographique de ce travail.

Enfin, je voudrais remercier toutes les personnes qui ont contribué à mon stage ou à la rédaction de ce mémoire.

Abstract

Ubiquitous computing, which is the innovative technological era, continues to invade human existence as they willingly incorporate new devices into their everyday lives. This willingness creates an acceptance for computer scientists to decode people's tactic behavior when interacting with peers. Doing this will help better merge technologies with everyday life. If there is tactic behavior through a series of gestures and distances between people this is also reflected by their gaze. Therefore, eyes play a fundamental role in human behavior and are necessary to be used as a modality in ubiquitous systems. The purpose of this work is to improve the understanding of human gaze, how technology can analyze it and better integrate it into a proxemic ubiquitous system: an interactive panel in a public place.

Résumé

L'informatique ubiquitaire, la nouvelle ère technologique, envahit petit à petit le quotidien des hommes. Ceux-ci intègrent volontiers de nouveaux dispositifs dans leur vie de tous les jours jusqu'à parfois même considérer la technologie comme son égal. Il n'est pas alors insensé de permettre à l'informatique de décoder le langage tacite de l'homme dans ses interactions avec ses semblables afin que la technologie se fonde de plus en plus dans ce quotidien. Et si ce comportement tacite passe par toute une série de gestes et de distances entre les personnes, il se traduit aussi et principalement par le regard. Les yeux jouent donc un rôle fondamental dans le comportement humain et il est donc nécessaire de l'intégrer comme modalité dans un système dit ubiquitaire. L'objectif de ce travail a pour but de mieux comprendre le regard de l'homme et son analyse par la technologie ainsi que de permettre une meilleure intégration de ces données dans un système ubiquitaire proxémique : un panneau interactif dans un lieu public.

Table des matières

Chapitre 1

Introduction

Un monde totalement informatisé, où la télévision s'allumerait à peine entré dans la pièce pour se brancher sur la chaine favorite de son utilisateur ; où, lorsqu'il souhaite changer de chaine, il n'aurait plus qu'à balayer l'espace d'un mouvement de la main ; où le son se couperait lorsque ce dernier reçoit un appel téléphonique. Dans ce même monde, l'homme pourrait contrôler les panneaux publicitaires d'un regard, avoir l'horaire de sa séance de cinéma en balayant l'écran des yeux ou encore pourrait zoomer sur un plan électronique du centre commercial sans avoir à aller appuyer sur un bouton. Et tout cela s'effectuerait sans que l'homme ait même conscience que l'informatique se cache derrière. Toutes ces tâches s'intégreraient à son quotidien de façon à ce qu'il utilise chacun de ces objets informatiques sans plus se poser de question. Ne serait-ce pas quelque peu magique ?

Si cette vision semble totalement futuriste, c'est cependant ce vers quoi tend le futur. Et rien de magique ne se cache derrière tout cela, c'est simplement le résultat de la combinaison de principes sociologiques (les interactions proxémiques) avec la technologie embarquée poussée à l'extrême (l'informatique ubiquitaire).

Les sujets de l'informatique ubiquitaire, très en vogue ces dernières années, ont déjà fait l'objet de nombreuses recherches, et, bien qu'actuellement aucun objet ne masque complètement l'intégration de la technologie, certains concepts plongent tout de même l'humain au cœur de certains films de science-fiction. Cependant, il a plus rarement été couplé aux interactions proxémiques afin d'inclure la technologie au même stade que l'homme et de ramener leurs interactions sur un pied d'égalité.

Bien que le croisement de ces deux concepts pourrait certainement faire avancer l'ère technologique actuelle, il est difficile d'imaginer l'homme commencer à interagir avec les objets technologiques comme s'ils étaient un de leurs semblables, surtout dans des lieux publics. Alors, plus discrètement, serait-ce une option envisageable de coupler le regard avec ces systèmes ubiquitaires proxémiques ?

Le but de ce travail est alors de se pencher sur les challenges que présentent

la réalisation d'un panneau interactif placé dans un lieu public avec lequel les passants seraient capables d'interagir via le regard, sans même une formation ou une explication donnée par un professionnel. Un tel énoncé présentant des besoins aussi particuliers permettait d'ouvrir la porte sur une multitude de questions.

Il sera donc envisagé en quoi et comment le regard peut être une modalité dans une informatique ubiquitaire couplée aux interactions proxémiques, où en sont les avancées dans le domaine actuel, quelles sont les difficultés auxquelles les chercheurs vont être confrontés afin de résoudre un tel challenge.

De plus, une seconde question sera posée : en quoi la situation awareness (la conscience de la situation), un concept fort présent dans les études d'utilisateurs mais rarement employé pour les ressources qu'il met en œuvre lui même, pourrait présenter un avantage dans le cadre de l'intégration du regard dans ces systèmes informatiques plus que poussés.

Pour tenter de répondre à ces questions, plusieurs travaux ont été réalisés : le premier dans le cadre d'un stage réalisé à l'Université du Texas de Austin, dans la section de l'école d'Information, combinant eye tracking et situation awareness et permettant la maitrise de ces deux premiers concepts, et le second réalisé en vue de l'écriture de ce travail et tentant de lier la partie déjà réalisée avec l'interaction proxémique et l'informatique ubiquitaire.

Ce travail tentera de répondre aux questions posées en présentant alors la structure suivante : tout d'abord un état de l'art complet définira successivement l'eye tracking, en expliquant d'abord le fonctionnement biologique et psychologique du regard humain avant de présenter les dispositifs de tracking disponibles actuellement. Une fois cette technologie comprise, une explication détaillée de la situation awareness sera apportée afin de comprendre ce dont il s'agit, ainsi que les différentes manières de la calculer. L'état de l'art se terminera enfin en présentant ce phénomène psychologique appelé interactions proxémiques ainsi que le concept d'informatique ubiquitaire avant de lier toutes ces parties ensemble pour montrer les particularités théoriques d'un tel projet. Ensuite, la partie de réalisation sera abordée afin de présenter les différents travaux réalisés dans le cadre de cet écrit. Enfin, avant de conclure, une partie permettant à de futures recherches d'être entamées sera abordée afin de présenter les challenges qui ont ou non été résolus lors de ce travail.

Chapitre 2

État de l'art

Cette partie permettra de poser et d'établir des connaissances théoriques de la matière avant de pouvoir aborder le côté pratique de ce travail. Différentes notions y sont décrites afin que le lecteur ait en main toutes les données nécessaires à la bonne compréhension de la suite de ce travail.

De ce fait, le premier élément qui sera abordé sera la compréhension de l'eye tracking et du regard de l'homme en général. Ensuite, les mesures d'attention seront abordées en focalisant plus particulièrement sur celles qui ont été employées dans le cadre du stage réalisé à l'Université du Texas, à Austin. Enfin, la question des interactions proxémiques sera évoquée, d'abord de façon plus générale avant d'y incorporer les notions d'informatique ubiquitaire et d'eye tracking.

2.1 Eye tracking

L'homme est comme un animal en quête de nourriture : il est « infomnivore[1] ». Il désire trouver des informations rapidement et ne souhaite pas déployer trop d'efforts sous peine d'abandonner une piste qui ne lui fournira pas assez rapidement la sensation qu'il se rapproche de son but[10]. C'est pourquoi les études d'utilisabilité (UX) deviennent une tendance incontournable à l'ère actuelle. Le fournisseur de contenu souhaite offrir à l'utilisateur ce qu'il désire afin de le fidéliser.

Cette quête d'information s'effectue principalement par le regard, qui va fournir à l'utilisateur un premier point d'entrée face au dispositif parcouru, comme un site internet, par exemple. Or, le mouvement de l'œil est un mouvement qui n'est ni facilement contrôlable, ni facilement observable[3]. Parfois conscient, parfois inconscient, il offre aux études d'UX, notamment, une meilleure compréhension du comportement et des pensées de l'utilisateur. C'est pourquoi il est intéressant d'étudier le fonctionnement d'un eye tracker et les informations qu'il fournit.

[1] Qui consomme des informations.

Le procédé de traquage de l'œil est un procédé d'identification qui permet de recenser où une personne regarde et comment elle regarde cet endroit. Ce processus sera expliqué plus en détails un peu plus loin dans ce chapitre. Cependant, il est important de faire le point sur quelques notions basiques concernant l'œil avant de continuer. Ensuite seront expliqués le fonctionnement d'un eye tracker et son utilité. Enfin, une description des différents procédés de tracking seront abordés.

2.1.1 L'œil et le regard

Avant de plus entrer dans les détails concernant l'eye tracking, il est important de comprendre pourquoi l'œil bouge ainsi que son fonctionnement. En effet, nous ne pouvons étudier leur mouvement que parce que les yeux ne restent pas fixes. Or, quelle en est la raison ? Il est donc impératif de comprendre le fonctionnement biologique du regard pour pouvoir le traquer de façon efficace.

Le mouvement de l'œil

Pour des raisons de survie, l'œil humain possède une vision périphérique énorme (90° sur le plan vertical et 90° par œil sur le plan horizontal), comme il est possible de le voir dans la Figure 2.1. Cependant, si tout ce que l'homme voyait dans ce champ de vision était net, son cerveau serait surchargé d'informations plus ou moins utiles. Dès lors, seul un foyer d'approximativement 2° est parfaitement net (vision fovéale) ; de 2 à 5° un premier flou commence à recouvrir les détails (vision parafovéale) ; et au-delà des 5° tout est complètement flouté (vision périphérique)[3]. Ce flou progressif est illustré à la Figure 2.2.

Pourtant, il n'y a pas besoin de faire l'exercice pour se rendre compte que, lorsque l'homme regarde devant lui, il est très rare que la majorité des éléments qu'il voit soient flous tandis que seule une infime ouverture lui semble nette. C'est grâce au mouvement des yeux. En effet, l'œil humain saute d'emplacements en emplacements jusqu'à plusieurs fois par secondes. Ces mouvements sont appelés des saccades oculaires et durent généralement moins de 50ms. Lors d'arrêts temporaires, appelés fixations oculaires, le cerveau va profiter pour extraire les informations nettes de la vision.

Ainsi, l'œil bouge afin de récolter une multitude de petites images nettes qui seront assemblées par le cerveau telles des pièces de puzzle afin de laisser croire à l'homme qu'il voit tout de façon nette.

L'attrait du regard

Précédemment, il a été établi que l'œil bougeait afin de fournir une image nette et précise des alentours. Cependant, tous les hommes ne regardent pas les mêmes choses dans le même ordre ou en leur accordant la même importance. C'est là que se trouve alors tout l'intérêt du traquage de l'œil.

En effet, le comportement visuel est influencé par ce qui attire subitement l'homme (basé sur des stimuli, des objets qui contrastent) mais aussi par l'intention volontaire de regarder quelque chose (basé sur la connaissance et les

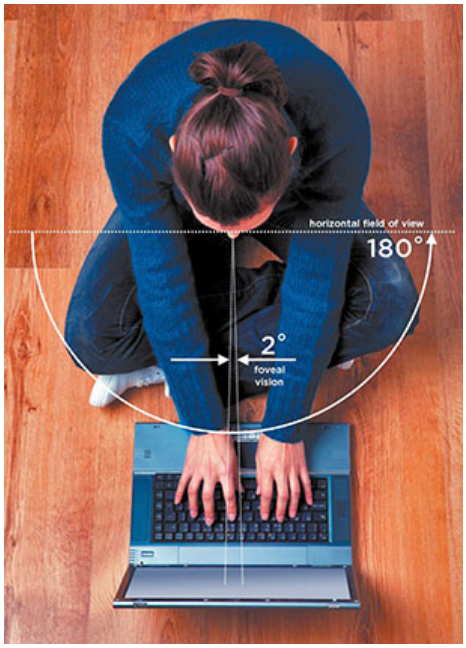

FIGURE 2.1: Angle de vision permis par les yeux[3].

FIGURE 2.2: La vision fovéale de 2°.[3].

attentes). Dès lors, deux personnes regardant la même scène ne verront pas les éléments dans le même ordre, voire ne verront pas du tout certains éléments qui ont pu en interpeler d'autres.

Puisque le regard et l'attention sont liés, il est facile de comprendre l'utilité de traquer le regard des utilisateurs lors de certaines recherches. Capturer où leur regard se pose et comment il bouge permet de savoir où et comment leur attention est répartie et, dès lors, de nombreuses actions peuvent être menées afin d'amener les utilisateurs à préférer un moyen d'obtenir des informations à un autre, tel qu'il le sera expliqué par la suite.

2.1.2 Les eye trackers

Maintenant que ce point biologique a été posé et que le fonctionnement de l'œil a été compris, le cœur de ce travail peut être abordé : le fonctionnement des eye trackers.

Comme expliqué ci-avant, l'eye tracking consiste en l'identification de l'endroit et de la façon dont une personne regarde. De ce fait, le tracker va mesurer les caractéristiques liées au regard (localisation, temps passé à fixer certains points, mouvements des yeux) comme celles de l'œil en lui-même (taille de la pupille)[3].

Qu'est-ce qu'un eye tracker ?

Il faut savoir qu'un eye tracker, en tant que dispositif spécifique[2], est un objet capable d'enregistrer le mouvement des yeux d'une personne. Son fonctionnement est assez simple : le dispositif est muni de lumières infra-rouges qui se reflètent alors sur le visage de l'utilisateur. Le choix de l'infra-rouge n'est pas anodin puisqu'il est non visible par l'homme, ce qui ne le déconcentrera pas de sa tâche, et peu dangereux pour l'œil, à condition de n'y être exposé qu'à faibles doses. À l'aide d'une caméra sensible à cette fréquence, l'appareil va pouvoir capturer la réflexion rétinienne (réflexion des canaux sanguins afin de capter le centre de la pupille, voir Figure 2.3) et la réflexion cornéenne (l'espèce de point blanc présent sur l'œil)[3].

Quand l'appareil possède ces deux données, il n'a plus qu'à les soustraire l'une à l'autre pour obtenir le ratio entre les deux. C'est lorsque ce ratio change qu'il est alors possible de savoir que le regard n'est plus au même emplacement et de calculer quel est le nouveau point que l'utilisateur regarde. Quelques particularités sont tout de même à relever : lorsque l'œil bouge, la réflexion cornéenne reste au même endroit alors que la pupille change clairement de position (voir Figure 2.4), tandis que lorsque la tête de l'utilisateur se déplace alors qu'il fixe toujours le même endroit, le ratio entre les deux ne change plus (voir Figure 2.5)[3].

Afin de pouvoir obtenir ces deux données et de les associer à quelques points initiaux, ce qui permet de fixer une base aux calculs du tracker, une calibration

[2]Comme il le sera expliqué un peu plus loin, une webcam peut servir d'eye tracker. Cependant, son fonctionnement n'est pas du tout similaire à celui décrit ici.

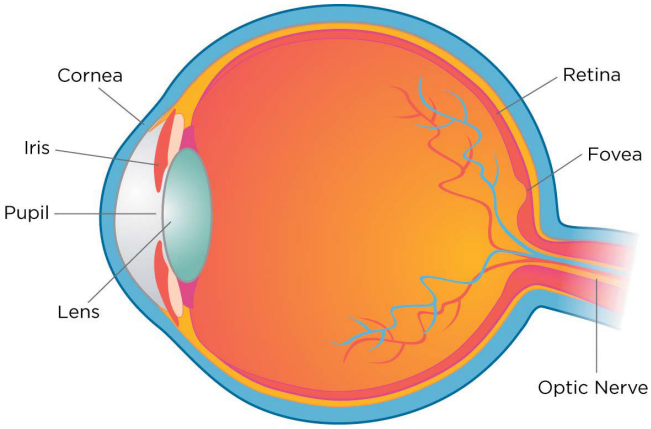

FIGURE 2.3: Schéma d'un œil[3].

FIGURE 2.4: La réflexion cornéenne ne change pas quand l'œil bouge[3].

FIGURE 2.5: Le ratio entre les réflexions reste identique quand la tête bouge[3].

doit être effectuée. Les eye trackers modernes sont désormais tous capables d'effectuer la tâche en quelques secondes[24]. La calibration consiste à demander à l'utilisateur de regarder des points prédéfinis, par exemple (voir Figure 2.6). De la sorte, le tracker peut associer les réflexions détectées à une position connue. Comme tout homme est constitué différemment, ses « paramètres » de regard sont différents (vitesse, précision, etc.). La calibration va permettre de réduire les erreurs de traquage induites par ces différences-là.

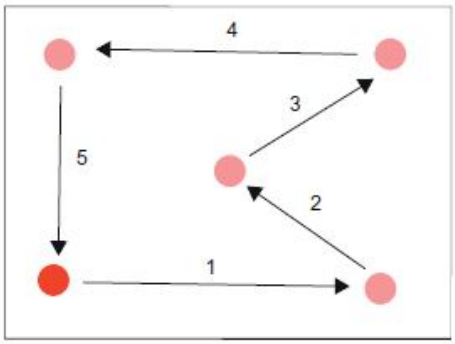

FIGURE 2.6: Une séquence de calibration typique[24].

À quoi sert un eye tracker ?

Il a été expliqué plus haut ce qu'était un eye tracker et comment l'appareil fonctionnait. Cependant, il est important de se poser la question de l'utilité de ce dernier.

Comme évoqué très brièvement en début de chapitre, ce dispositif, qui permet d'enregistrer le regard de l'utilisateur, est employé par de nombreuses personnes, que ce soient des chercheurs, des ergonomes ou des experts des études d'expérience utilisateurs (UX)[24]. Par la suite, seul le terme « chercheurs » sera employé, cependant, il réfère à toutes les personnes menant une étude avec un eye tracker, quelle qu'en soit la finalité.

Bien qu'un eye tracker ait des utilités diverses, ses principales utilités sont les suivantes : la recherche, les études UX et les études de marketing. Si la première a pour but d'étudier le comportement du regard et de trouver des applications nouvelles à l'eye tracking, les suivantes trouvent plus souvent leur place en tant que conseil dans les entreprises. En effet, bien que les études UX visent un contenu virtuel, ces deux formes d'études se complètent afin de prouver, par exemple, à un PDG et ses investisseurs à quel point il est primordial de refaire le design de l'application ou du site web de la compagnie, voire même le pakaging d'un objet mis sur le marché. Les études de marketing viseront surtout à montrer que l'utilisateur est perdu face aux informations qui sont mises à sa disposition, tandis que les études UX auront plutôt pour but de comprendre où l'utilisateur s'est trompé, pourquoi n'a-t-il pas été plus attiré par certains contenus, permettant alors aux ergonomes et aux designers de retravailler le style d'un site web, par exemple. Pour résumer, l'eye tracking est donc une méthodologie

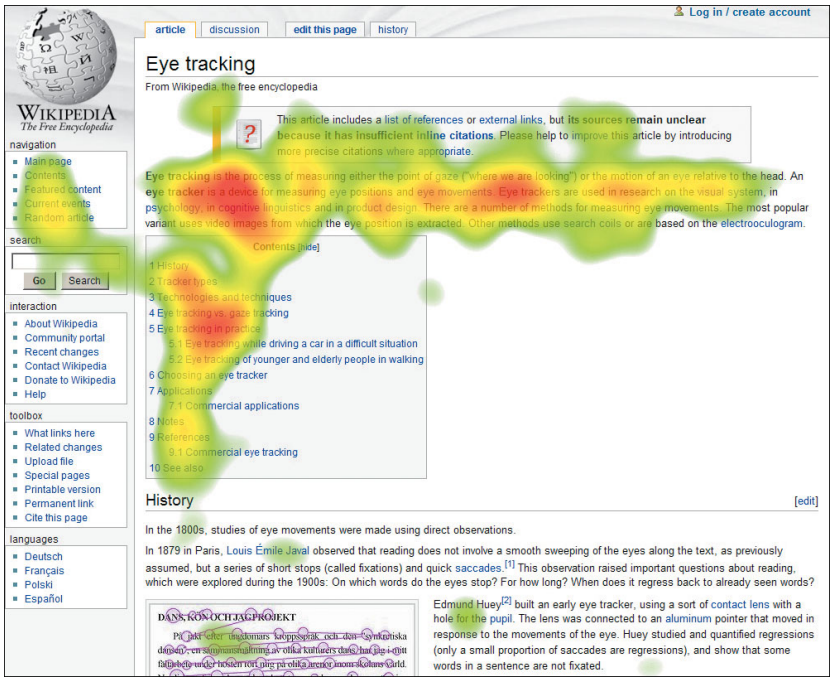

FIGURE 2.7: Visualisation de type heatmap [3].

qui aide les chercheurs à comprendre l'attention visuelle des personnes en quête d'information[24].

Lors de l'utilisation d'un eye tracker, le chercheur va récupérer plusieurs types de données, comme la position de la pupille droite, de la pupille gauche, leur taille respective, ... Selon la marque et le type d'appareil employé, un tableau plus ou moins complet va lui être fourni (Voir Annexe B.3). Ces données peuvent être traitées manuellement, par un logiciel « maison » ou encore par des logiciels industriels puissants (fournis ou non par les développeurs même de la technologie).

Tandis que les deux premières options ne seront pas détaillées plus, car elles dépendent de besoins spécifiques à chaque chercheur, la troisième mérite d'être analysée un peu plus en profondeur. En effet, ces logiciels permettent de générer presque instantanément des visualisations de données récoltées au cours de l'expérience. Celles-ci s'expriment sous forme de graphiques, dont les deux types les plus employés dans le cadre des expériences utilisateurs sont les heatmaps et les gaze plots.

Une heatmap (voir Figure 2.7) consiste en un genre de cartographie en couleurs afin de pouvoir placer où les fixations ont été les plus nombreuses ou les plus longues. Typiquement, les couleurs qui y sont employées présentent un dégradé du rouge, pour les zones de condensation forte, au vert, symbole de la condensation la plus faible[24]. Les zones sans couleur sont des zones où aucune fixation n'a été enregistrée. Bien que cela semble étrange, une zone qui n'en-

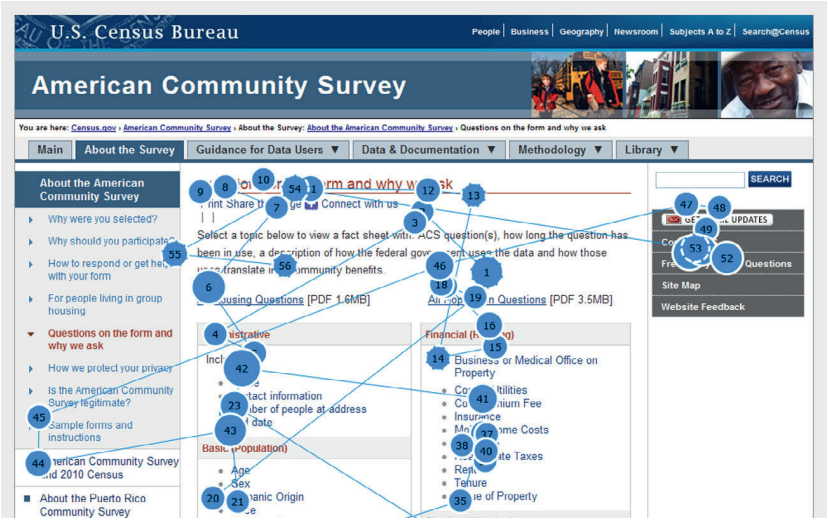

FIGURE 2.8: Visualisation de type gaze plot[20].

registre pas de fixations ne signifie cependant pas que l'emplacement n'a pas été examiné au cours de la collecte d'informations. En effet, il a été expliqué précédemment que l'homme ne percevait de façon nette qu'environ 8% de son champ de vision. Cependant, il est capable de distinguer des éléments présents dans ses visions parafovéales et périphériques. Ainsi, ce n'est pas parce que l'eye tracker n'a pas détecté de fixations ou de saccades dans une zone que cela signifie que l'utilisateur ne l'a pas vue. Il l'a surement captée de façon générale mais cet endroit n'a pas suscité suffisamment son attention pour qu'il s'y attarde[24]. Un exemple typique d'utilisation d'une heatmap serait, par exemple, de présenter au client ce que l'utilisateur voit ou pas sur son site pour lui prouver qu'il doit revoir le design de ce dernier[20].

Un gaze plot (voir Figure 2.8) consiste en un graphique de fixations et de saccades pour un laps de temps particulier. Typiquement, les fixations sont représentées par des ronds et les saccades par les barres les reliant. Plus une fixation est longue, plus le rond qui la représente est étendu[24]. De plus, l'ordre d'enchainement des fixations est notifié au sein des ronds correspondants, ce qui permet de pouvoir retracer avec certitude la façon dont la page a été scannée par un utilisateur[20]. Cela permet de distinguer des points d'entrée sur un site, par exemple, ou encore de voir comment il parcourt un texte littéraire.

Les différents appareils

L'homme s'intéresse au regard de ses semblables depuis la fin des années 1800 et on retrouve même un premier appareil du début du 20^{e} siècle ayant des apparences de machine de torture (voir Annexe A.1). Fort heureusement, les eyes trackers ont bien évolué depuis cette époque.

Actuellement, les appareils se distinguent en deux grandes catégories : les dispositifs portables et les dispositifs distants[3] (voir Figure 2.9). La princi-

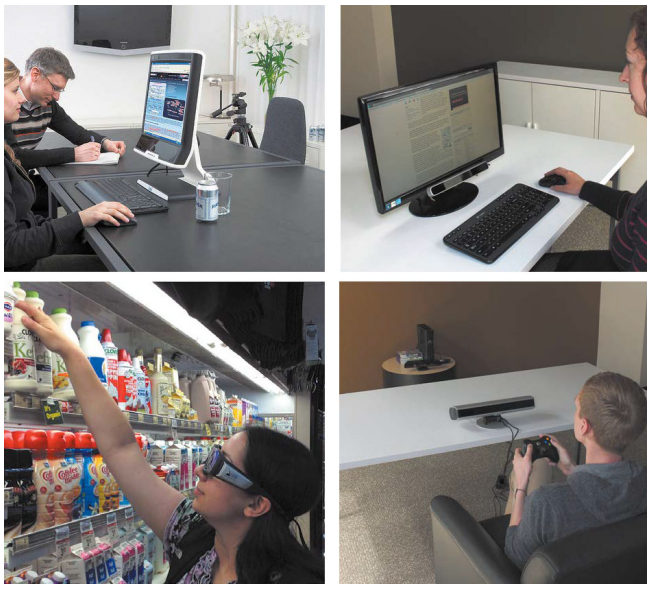

FIGURE 2.9: Différents types d'eye trackers : en haut à gauche, un eye tracker distant intégré ; en haut à droite, un eye tracker distant externe ; en bas à gauche, un eye tracker portable ; en bas à droite : un eye tracker distant indépendant[3].

pale différence entre les deux est évidemment la localisation de l'équipement pendant l'expérience puisque les premiers, mobiles, se situent sur la tête de l'utilisateur, tandis que les seconds restent fixes. De même, au sein des eye trackers distants, on peut retrouver plusieurs grandes catégories. D'une part, les appareils qui sont reliés à un ordinateur (intégrés ou à brancher), et, d'autre part, les appareils indépendants, qui peuvent alors tracker le regard pour n'importe quel équipement. Chaque type de dispositifs possède des avantages sur les autres ; ainsi, les trackers connectés à l'ordinateur sont plug and play tandis que les indépendants sont plus simples à configurer et, enfin, les trackers portables sont plus écologiques mais offrent un moins bon contrôle quant à l'analyse des données[3].

Il existe plusieurs mesures pour comparer les différents appareils d'un même type et choisir le meilleur. Bien qu'elles ne présentent pas toutes le même stade d'utilité et que, souvent, le choix du tracker se fasse plutôt à la réputation de la marque, il est important d'en évoquer quelques unes qui s'avèrent être les plus employées.

Le critère de comparaison le plus populaire, utile pour comparer les modèles au sein d'une même marque, est le taux d'échantillonnage (Hz)[3]. Ce taux se définit comme étant le nombre de fois où le tracker enregistre la localisation du regard de l'utilisateur en une seconde. Par exemple, un appareil à 250Hz enregistrera 250 points pour 1 seconde écoulée. Le taux minimum d'échantillonnage est de 25Hz tandis que le taux le plus élevé se situe aux environs de 2000Hz. Pas besoin de posséder un eye tracker à un taux d'échantillonnage très élevé pour faire de simples études. En effet, typiquement, l'appareil tournera aux alentours de

50 à 120 Hz. Cependant, lorsqu'il est important d'avoir une meilleure précision quant aux données récoltées, comme, notamment, la durée des fixations, un eye tracker soumis à un plus grand taux d'échantillonnage est nécessaire (le taux d'erreur lors de l'estimation du temps de fixation s'approcherait des 20ms à 25Hz et seulement des 2ms à 250Hz). Un appareil qui tourne à plus de 250Hz sera typiquement employé pour la reconnaissance de saccades et les clignements des yeux, en plus des fixations.

Les autres critères jouent un rôle moins important dans le choix du tracker, cependant, il est toujours utile de les mentionner. Ainsi, il existe des critères d'exactitude et de précision[3]. L'exactitude se mesure par la différence entre ce que l'œil regarde réellement et la position enregistrée par l'appareil, tandis que la précision mesure la capacité de l'eye tracker à reproduire exactement un mouvement des yeux effectué plus d'une fois à la suite (voir Annexe B.2). Une autre mesure serait la taille de la boite de tête, c'est-à-dire la boite virtuelle dans laquelle la tête de l'utilisateur peut se déplacer tout en sachant que le tracker sera encore capable de traquer le mouvement des yeux[3]. Il est aussi possible de comparer les dispositifs qui n'enregistrent la position que d'un seul œil contre ceux qui se calquent sur les deux ou encore la façon dont ils illuminent les pupilles pour les faire contraster avec les iris (il est possible de noircir la pupille ou de l'éclaircir)[3]. Cependant, ces deux dernières caractéristiques ne sont pas assez importantes que pour permettre de faire un choix décisif de l'appareil à utiliser.

En outre, bien que ne faisant pas partie de la famille des eye trackers, une webcam peut aussi servir de dispositif pour enregistrer le mouvement de l'œil de l'utilisateur. Au lieu d'utiliser des lumières infrarouges qui vont illuminer le visage et permettre de capter les différentes réflexions des yeux, tel qu'expliqué plus tôt dans ce chapitre, la caméra se contente d'enregistrer une succession d'image à un rythme plus ou moins important (frames per second). Cet appareil va alors extraire des images les données caractéristiques du visage, afin d'identifier les yeux et les traquer. Afin d'accomplir cette prouesse, diverses techniques peuvent être employées : demander à l'utilisateur de pointer lui-même son œil, jouer avec le contraste et la luminosité afin de ne plus percevoir que certaines caractéristiques du visage, utiliser des techniques de machine learning afin de détecter l'œil, ... Ces méthodes ne permettent que la détection des yeux sur le visage d'un utilisateur, elles ne sont donc pas exemptes d'une calibration, au même titre que les eye trackers.

Selon le taux de rafraichissement de la caméra, sa résolution, la luminosité ou l'éclairage de la pièce, la position de l'utilisateur et bien d'autres facteurs, ces techniques seront plus ou moins efficaces pour capter le regard d'un utilisateur efficacement. Bien que les algorithmes de machine learning permettent une meilleure précision, ils nécessitent d'y consacrer beaucoup de temps et de moyens, ainsi que la présence de nombreux « cobayes » permettant d'entrainer le logiciel. Ainsi, les autres techniques seront plus souvent privilégiées, au détriment de la précision.

L'emploi d'une webcam peut être bénéfique et justifié dans certaines études. En effet, bien que sa précision soit amoindrie comparativement à celle des eye

trackers[3], ce dispositif, peu cher, offre une solution accessible à un grand nombre d'utilisateurs. De cette popularité de la caméra, ce petit appareil, très répandu dans les foyers, peut offrir une plus grande couverture des études UX : plus besoin de faire venir l'utilisateur jusqu'à une centrale de test où le matériel est présent, il peut se faire interroger depuis chez lui. Par ailleurs, certaines études se basent déjà sur l'utilisation de la webcam (par exemple, la reconnaissance d'émotions) et il peut être plus rentable, lorsque plusieurs métriques sont étudiées, de n'utiliser qu'un seul instrument de mesure[4].

[3]Un écart moyen de 10cm entre ce que l'utilisateur regarde et ce qui est enregistré a été calculé[7].

[4]Plus de détails sur les avantages et inconvénients des webcams peuvent être trouvés sur le blog de iMotions : `https://imotions.com/blog/webcam-eye-tracking-vs-an-eye-tracker/`

2.2 Situation Awareness

Il n'est pas rare, lors de la lecture d'études basées sur l'eye tracking, d'être confronté au concept de « conscience de la situation », référé plus communément dans les travaux scientifiques par sa traduction anglaise : situation awareness (à partir de ce point, ce concept sera toujours abrégé par son sigle, SA).

Dans cette partie, il sera expliqué au lecteur pourquoi ces études sont souvent associées et comment elles se complètent, après avoir expliqué en détail ce concept qu'est la SA. Enfin, il sera détaillé de façon théorique les différentes mesures qui ont été retenues dans le cadre de ce travail. Afin de permettre une meilleure compréhension de ces mesures, un aparté devra être fait afin d'expliquer une notion intimement liée au domaine des eye trackers : les zones d'intérêt.

2.2.1 Compréhension de la SA

L'homme possède une capacité d'attention limitée dédiée aux tâches qu'il doit effectuer. Lorsqu'il accomplit une mission caractérisée par un ou plusieurs objectifs, il doit diviser cette attention entre plusieurs tâches, plusieurs événements, qui, parfois, ont lieu simultanément. Quand la demande totale d'attention excède cette capacité maximale, l'homme doit alors sacrifier son attention sur des tâches qu'il considère comme non prioritaires face aux autres. Si l'attention placée dans une tâche diminue en-dessous d'un certain seuil, alors ses performances s'en trouvent diminuées[11].

Comme il est expliqué de façon quelque peu détournée dans ce cas de figure, la SA se caractérise par la conscience de ce qui se passe autour de soi (« savoir ce qu'il se passe »[8]) et par la compréhension d'une information et des conséquences qu'elle aura sur la situation actuelle et future[9]. La définition la plus reprise est celle de Ensley[8] : « C'est la perception des éléments dans un environnement étant donné un volume de temps et d'espace, la compréhension de leur signification et la projection de leur statut dans un futur proche. »

Il est important de noter que la SA est toujours définie dans des termes opérationnels, c'est-à-dire en vue d'un but à accomplir, dans une situation précise, afin que l'agent puisse prendre les décisions qui y sont liées[9]. Le terme SA est emprunté au vocabulaire militaire qui l'utilisait lors de missions de pilotages périlleuses et difficiles ; toute leur attention était alors requise afin de prendre des décisions judicieuses[9]. Il est alors évident qu'il ne faut pas confondre conscience de la situation avec prise de décisions : le premier est un des facteurs précurseurs influençant et étant influencé par le second (voir Annexe A.2)[8].

Les trois niveaux

Cette définition contient beaucoup de concepts importants et il est essentiel de les expliciter un minimum afin d'obtenir une connaissance vaste et complète du sujet. En relisant attentivement la définition donnée ci-dessus, il est visible que le concept soit découpé en trois parties : la phase de perception, la phase

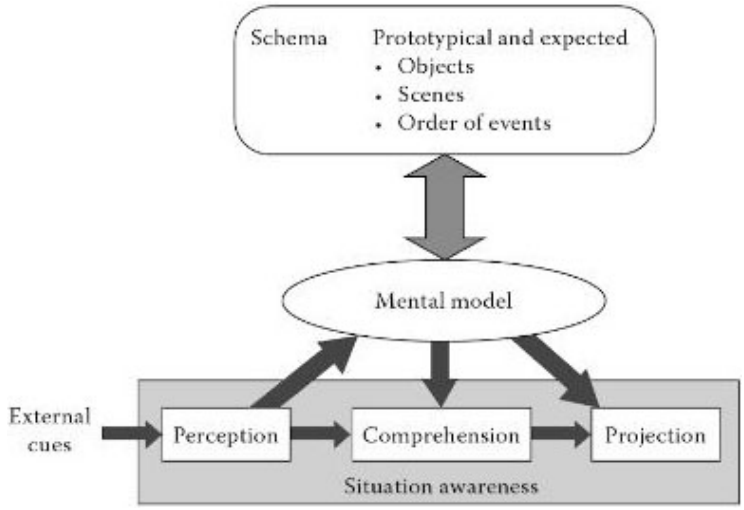

FIGURE 2.10: Schéma de la formation du modèle mental via les 3 niveaux[9].

de compréhension et la phase de projection[18]. Ces trois niveaux sont essentiels dans la création d'un schéma mental (compréhension du fonctionnement des choses à l'aide d'une représentation systématique) qui sera lui-même un élément clé de la prise de décision (voir Figure 2.10).

Le premier niveau défini est la phase de perception. Bien que différente pour chaque tâche spécifique, la perception se fait de la même façon pour tous : au moyen des sens (vue, odorat, ouïe, toucher, goût) et est liée à l'environnement direct de l'utilisateur[9]. Cette tâche est fondamentale, ce qui permet à tout un chacun de se représenter une image correcte de la situation[8]. C'est à ce moment que l'utilisateur acquiert l'information nécessaire pour la suite[18]. Cette étape est tellement primordiale que dans de nombreuses études menées, 76% des erreurs faites lors de la prise de décision proviennent d'une mauvaise perception de l'information (qu'elle soit donnée à la mauvaise personne ou non présente au moment où l'agent en a besoin)[9].

Le second niveau est appelé phase de compréhension. Comme son nom l'indique, c'est à ce niveau que l'utilisateur va comprendre les données qu'il a à sa disposition et leur donner un sens[9]. L'agent, dans la phase précédente, a récolté de nombreuses informations, sans lien apparent entre elles, et doit, à ce niveau, les combiner et les prioriser afin de leur donner une signification sensée[9]. Lorsque l'homme traite l'information, il faut cependant faire attention car il veut que la donnée serve son but. Dès lors, cette dernière possède à la fois une signification subjective (« awareness ») et à la fois objective (« situation »)[8]. Selon les mêmes études que citées au niveau précédent, à ce stade de compréhension, l'agent a un risque de 20% de faire une erreur (qui ne provient pas de la collecte d'information)[9].

Enfin, le troisième et dernier niveau est appelé la phase de projection. C'est l'activité mentale résultante de tout ce procédé[18]. Quand l'utilisateur a com-

pris à quoi servent les éléments, il doit imaginer leur impact concernant le futur[9]. Ce dernier pilier demande un niveau de compréhension de la situation (niveau 2) assez élevé [8].

Par ailleurs, force est de constater que la temporalité joue un rôle crucial dans la SA d'un modèle dynamique. En effet, mis à part son évocation dans la définition de cette dernière (« [...] dans un environnement étant donné un volume de temps et d'espace [...] »), il est possible de décortiquer les niveaux pour y montrer que tous sont assujettis à la problématique. Dans le niveau 3, cela est démontrable de façon très évidente : par exemple, lorsqu'un utilisateur pose un acte, il doit se demander combien de temps est mis à sa disposition avant que ce dernier n'ait un impact sur ses objectifs. Le deuxième niveau est, lui aussi, facilement impacté par le temps puisque l'utilisateur peut passer plus ou moins de temps à apposer une signification sur les différents éléments récoltés. Enfin, le premier niveau est lui aussi soumis à la problématique de la temporalité, même si cette notion est y moins naturellement présente. En effet, le modèle dynamique est basé sur le monde réel, et qui dit monde réel dit informations qui fluctuent en fonction du temps, justement. Dès lors, il est important de noter que la temporalité influencera la façon dont l'utilisateur traite les informations (et donc la façon dont l'utilisateur est conscient de la situation).

Pourquoi mesurer la SA ?

Comme mentionné précédemment, étudier la SA revient à s'intéresser à la compréhension de l'utilisateur de son environnement. Cela permet d'avoir un meilleur savoir concernant l'acquisition des connaissances, de comprendre comment l'homme assemble les différentes pièces du puzzle pour former une « image opérationnelle »[5][8].

Pour ne citer que les utilisations les plus répandues, grâce aux études de SA, il est possible d'évaluer le design, d'un site internet par exemple, ou d'évaluer des techniques d'entrainement spécifiques, comme les simulateurs de pilotage d'avions. Ce genre d'études est rarement employé seul dans le but de pratiquer uniquement la SA ; il est plus courant et naturel d'intégrer ce concept dans des études UX[8].

De même, être capable de mesurer la SA permet de mieux maitriser les facteurs qui l'influencent[8], et dès lors, en construire de meilleurs modèles dynamiques. La Figure 2.11 d'Endsley montre la représentation typique des modèles dynamiques en SA. Ainsi, par exemple, Wortelen[32], se basant sur cette théorie des schémas dynamiques a pu construire une simulation dynamique[6] de l'attention des automobilistes.

[5] Jargon militaire

[6] Dans sa simulation, Wortelen n'emploie pas les mêmes termes que Endsley. Cependant, ses trois étapes sont similaires : il reconnait le procédé cognitif (traitement d'une situation jamais rencontrée auparavant ; où l'utilisateur n'a aucune expérience), le procédé associatif (traitement d'une situation familière où l'agent se base sur des règles) et le procédé autonome (situation tellement bien entrainée que la résolution se fait sans que l'agent soit conscient de celle-ci)[32].

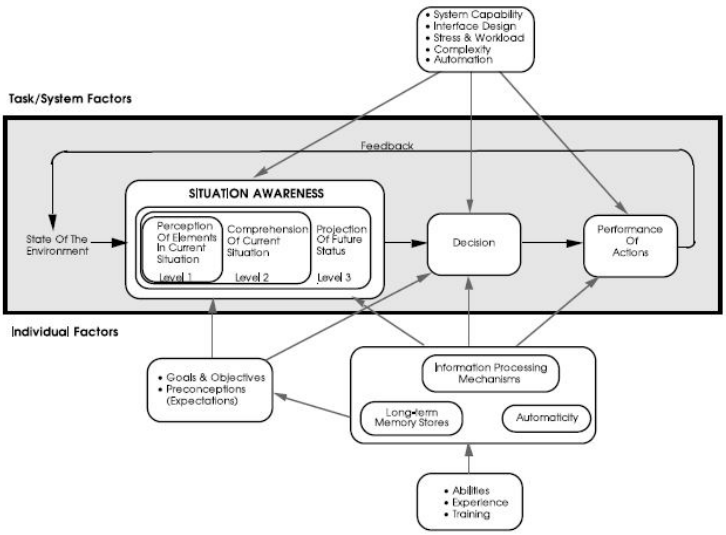

FIGURE 2.11: Schéma d'un modèle de SA dynamique[8].

Dans le cadre de ce travail, la SA a été étudiée afin de pouvoir être intégrée dans les logiciels développés, en tant que modalité. En effet, le but ici était de pouvoir mesurer l'attention de l'utilisateur lorsque son regard était traqué. Cette mise en situation sera expliquée plus en détails dans la partie technique de cet écrit. Néanmoins, les mesures qui ont été employées, et uniquement celles-là, seront expliquées théoriquement dans le point suivant.

2.2.2 Zones d'intérêt (AOI)

Afin de pouvoir traiter les mesures employées pour les études de conscience de l'utilisateur, il est nécessaire de faire un petit saut dans la matière précédente, les études d'eye tracking, et d'expliquer un concept qui avait été laissé de côté : les zones d'intérêt (ou AOI[7]).

Comme cela a été évoqué précédemment, la recherche visuelle est différente selon les personnes. Ainsi, même si des comportements assez identiques se dégagent, il n'est pas toujours évident de comprendre ce processus parmi les utilisateurs. Par exemple, l'homme aura tendance à examiner les objets en haut à gauche d'une page internet et de les examiner dans la forme d'un F (aussi appelé triangle doré, illustré à l'Annexe A.3). De même, lorsqu'un utilisateur navigue entre les sites web, il peut soit scanner rapidement le contenu pour savoir s'il lui plait (mode par défaut), soit diriger son attention en quête de certaines informations précises (influencé par une stratégie de recherche)[6]. De plus, un utilisateur peut très bien devenir, de façon inconsciente, « aveugle » face à certaines formes de contenu. En effet, ce dernier ignore totalement toutes les bannières que son cerveau identifie comme s'apparentant à de la pub, quand bien même cette bannière contiendrait des informations importantes et utiles[6]. En-

[7]Areas of Interest

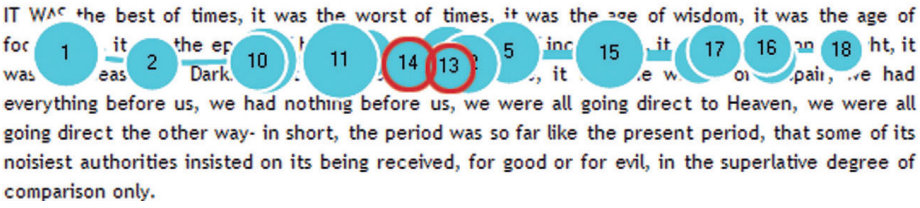

FIGURE 2.12: Lecture non linéaire d'un paragraphe[10].

fin, lorsque l'homme décide de s'arrêter sur du contenu dans le but de le lire, il a été constaté que sa lecture n'était pas linéaire (voir Figure 2.12) mais qu'il sautait des mots, revenait en arrière, fixait des parties de certains mots plus ou moins longtemps, etc.[10].

Il est donc assez simple de voir que tous les objets présents sur un design (site web, simulation d'un cockpit en aviation, etc.) se font concurrence et tentent tous d'accaparer l'attention de l'utilisateur qui les regarde[6]. Pour pouvoir étudier cette concurrence et les stratégies de scan mises en place par le cerveau humain et par l'œil, il a été intéressant de définir des zones d'intérêt (AOI).

Une AOI est une zone spécifique sur l'interface qui intéresse l'équipe menant l'étude UX (voir Figure 2.13)[20]. Cartographier les affichages en régions intéressantes[24] présente bien des avantages. En effet, cela permet d'extraire des informations beaucoup plus facilement qu'en étudiant « manuellement » les données récupérées de l'étude d'eye tracking. Par exemple, l'équipe en charge pourra déterminer aisément où l'utilisateur a regardé en premier (quel était son point d'entrée), comment et combien de temps a-t-il regardé certaines zones, est-il passé rapidement à autre chose ou s'est-il attardé sur cette zone, combien de fois est-il revenu pour observer ceci ou cela, dans quel ordre a-t-il analysé les choses, etc.

Grâce aux zones d'intérêt, toutes ces questions trouvent une réponse beaucoup plus facilement et permettent une meilleure maitrise de la problématique énoncée ci-avant. De plus, ces zones sont facilement implémentées (il s'agit de la définition de forme) et beaucoup de logiciels d'eye tracking industriels et complets, tel que iMotions, permettent de les intégrer très facilement (traçage des formes à la souris sur les interfaces). Par ailleurs, ces logiciels proposent aussi leur propre métrique d'analyse des AOI via des mesures populaires de SA (voir Annexe A.4). Ceci est à la fois un avantage et un inconvénient car l'équipe d'analyse n'a pas besoin d'être experte dans l'étude de la SA, cependant, ces analyses sont limitées par les définitions des développeurs de ces logiciels et ne permettent pas la personnalisation quant aux besoins spécifiques d'une étude.

2.2.3 Les différentes mesures

Cet aparté concernant les AOI permet enfin la possibilité d'expliquer quelles mesures ont été sélectionnées comme utiles dans le cadre de ce travail. Dans cette partie, les mesures seront expliquées de façon très théoriques. Cependant, les détails d'implémentation trouveront eux aussi une place dans la suite ce tra-

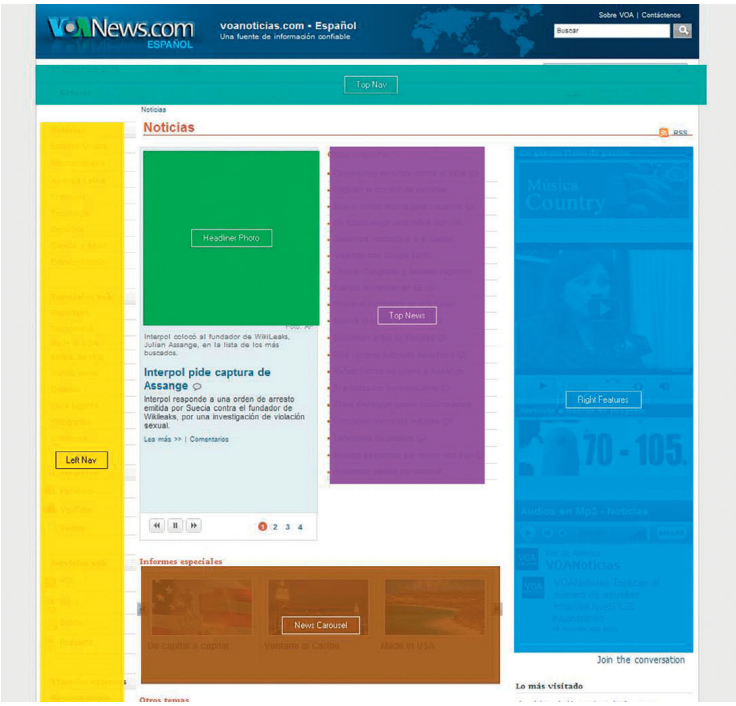

FIGURE 2.13: Exemple de zones d'intérêt sur une interface[24].

vail, étant donné les particularités qu'ils présentent.

Le choix des métriques intéressantes, dans le cadre des recherches menées pour ce travail, servant à mesurer la SA et l'attention de l'utilisateur a été principalement influencé par les travaux de Van De Merwe[18], Steichen[26] et Toker[27].

La première mesure qui a été exploitée est le dwell time ou, en français, le temps de séjour. Cette mesure permet de connaitre le temps de fixations total passé au sein d'une AOI. Au plus le dwell time est élevé dans une certaine zone, au plus l'utilisateur a fixé cette AOI pendant la période de temps étudiée. Lorsque le contexte est combiné à cette mesure, il est aisé de comprendre si l'utilisateur était perdu face à une interface ou s'il fixe telle partie de l'affichage pour certaines raisons. Beaucoup d'auteurs préfèrent calculer le ratio (ou pourcentage[31]) du temps de séjour, ce qui n'est qu'une toute petite manipulation supplémentaire à réaliser et permet une analyse parfois plus simplifiée. Cependant, pour des raisons techniques qu'impliquent le temps réel[8], calculer le ratio, bien que cela aurait été possible, n'aurait pas facilité la tâche dans le cadre de ce travail. Il est cependant important de noter que ce calcul de ratio est souvent la mesure qui est préférée dans les travaux traitant de la SA comparativement au dwell time.

[8]Cette contrainte, comme il le sera expliqué dans le chapitre suivant était une partie majeure de ce travail.

La seconde mesure qui a été intéressante d'analyser est l'entropie. Ce terme désigne le degré d'imprédictibilité d'un contenu (ici, le regard). Si l'entropie est élevée, cela signifie que l'utilisateur est surchargé visuellement et mentalement. De plus, au plus cette mesure est basse, au plus le mouvement de l'œil suit un comportement totalement prévisible (0 étant le seul le plus bas et décrivant un pattern devinable) et si la mesure est élevée, l'œil suit un schéma tout à fait aléatoire[18]. Cette entropie est calculée, pour une même interface, à l'aide des différentes AOI qui y sont définies. Cette mesure permet de voir si à un moment spécifique, lors d'une expérience de simulation, par exemple, l'utilisateur se retrouve submergé par les informations et complètement perdu face à une interface ou pas.

La troisième mesure qui a été travaillée est la distance entre des fixations qui se suivent. En effet, chaque fixation possède des coordonnées cartésiennes (x, y), placées à l'épicentre de la fixation[9]. Il est donc possible de calculer l'écart présent entre deux fixations à l'aide de la formule bien connue de distance cartésienne entre deux points. Il est important de noter que la distance n'est pas calculée entre deux fixations totalement aléatoires mais bien entre celles qui se suivent directement ; cela a de l'importance pour l'interprétation de cette mesure. Ces données permettent de connaitre la dispersion du regard de l'utilisateur, et donc, par extension, de savoir si l'utilisateur suit un certain pattern régulier ou non lors de ses fixations[19].

La dernière mesure exploitée est la distance totale parcourue par les yeux de l'utilisateur. De façon assez similaire à la méthode précédente, cela consiste à calculer les distances cartésiennes entre les fixations et d'additionner toutes ces mesures. De même, l'interprétation qui en résulte reste assez proche des interprétations précédentes et va en leur sens : plus la distance parcourue par le regard est grande, plus l'utilisateur a eu un comportement visuel dispersé, et donc probablement perdu, et vice-versa.

Bien que toutes ces données semblent parfois redondantes dans leur analyse de la SA, elles permettent toutes d'aborder les choses avec des points de vue différents. De plus, la liste ci-dessus ne présente pas une liste exhaustive des mesures à employer lors d'une étude de SA. D'autres se trouvent présentes dans les travaux cités tout au long de cette section, qui sont retrouvables en bibliographie, mais aussi dans d'autres articles non exploités dans le cadre de ce travail[10]. Par ailleurs, il faut rester attentif aux besoins de son étude et il est très aisé de définir de nouvelles méthodes pour mesurer la SA et lui apporter une interprétation qui fait sens. Enfin, il est à noter que les données récoltées sont toujours traitées selon les besoins spécifiques : soit en y posant une analyse mathématique afin de clore une expérience pratique et concrète[11], soit en les traitant ultérieurement (c'est ce qui a été réalisé dans la seconde partie de ce travail).

[9] Le regard se pose rarement 2 fois au même endroit ; même lors de fixations, il ne reste pas de marbre. Une fixation est alors déterminée comme étant l'ensemble des points dans un rayon acceptable pendant une certaine durée.

[10] Il est aussi possible de retrouver certaines mesures dans des logiciels très complets, comme iMotions (voir Annexe A.4).

[11] Pour ce faire, la méthode la plus employée est l'ANOVA et le test de Fisher.

2.3 Les interactions proxémiques

L'homme interagit avec ses semblables quasiment tout au long de sa journée. Qu'il soit dans la rue, entouré d'inconnus, qu'il prenne le bus avec son ami de toujours ou qu'il soit au travail et communique avec ses collègues ; l'homme est toujours en interaction avec ses semblables. Et de la même façon qu'il sait quelle forme de langage employer dans chacune des situations mentionnées ci-avant, il sait aussi comment se tenir en société.

Ces règles tacites qui lui semblent si naturelles ont été étudiées par les sociologues et appelées « interactions proxémiques ». Dans cette partie, ces règles seront étudiées en profondeur d'un point de vue sociologique avant d'être intégrées à l'informatique ubiquitaire. Enfin, cette partie se clôturera par l'intégration du regard dans les interactions proxémiques ; l'axe central de ce travail.

2.3.1 Théorie des interactions proxémiques

Comme mentionné ci-dessus, l'homme est sujet à des règles tacites ; un code de conduite en présence de ses semblables. Ces règles se forment car l'homme a une certaine perception des autres. Ainsi, selon les personnes qui l'entoureront et le contexte dans lequel il se trouvera, il sera plus ou moins proche (physiquement) de certains que d'autres. Ces distances interpersonnelles sont utilisées quotidiennement comme médiatrices dans les interactions entre les personnes[17].

Ce qui vient d'être expliqué ci-dessus présente brièvement ce que sont les interactions proxémiques. Ce terme, mentionné par Hall pour la première fois en 1966, peut être décomposé en « prox », du latin proimitas, qui signifie « ce qui est rapproché », et « émique » qui se rapporte, en sciences sociales, aux recherches du point de vue intérieur d'un groupe social[17]. Les interactions proxémiques désignent donc une des stratégies de canalisation des interactions entre les personnes[15] ; c'est une forme de communication implicite et non-verbale[12][17].

Selon Hall, étudier les informations proxémiques serait utile dans le but de comprendre « l'organisation de l'espace dans les maisons et buildings et, ultimement, le layout dans les villes » et pas simplement dans le but d'en savoir plus sur le comportement interactif de l'homme[17]. En effet, il sera vu un peu plus loin dans cette section que l'arrangement des espaces a une incidence sur le comportement et les interactions entre personnes en société.

Espace personnel

Les interactions proxémiques consistent en toutes les règles comprises implicitement par l'homme lorsqu'il gère ses relations spatiales[2]. Malgré la présence d'autres facteurs, Hall, dans ses études sociologiques, a remarqué la présence d'une corrélation entre la distance physique entre les personnes et la distance sociale qui caractérise ces mêmes individus[17].

[12] Hall appelle aussi cela le « langage silencieux »[17].

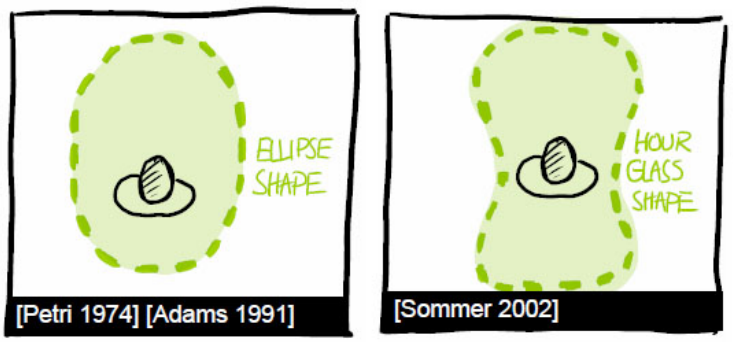

FIGURE 2.14: Exemples de formes que peut prendre l'espace personnel[17].

L'espace personnel représente la bulle d'espace que l'homme désire placer autour de son corps et qu'il autorise à envahir sous certaines conditions (voir Figure 2.14). Ce terme a d'abord désigné, en zoologie, la bulle de sécurité qui entourait un animal avant qu'il soit en danger face à un prédateur[17]. Il est, dès lors, important de distinguer l'espace personnel du territoire. Ce dernier désigne une position fixe munie de limites claires et visibles (dans le cas de l'homme, en tout cas) tandis que la première notion désigne une aire invisible autour du corps qui se déplace alors avec la personne qui a « dessiné » cette bulle[17]. Selon les caractères, les relations, le vécu et d'autres paramètres, cette aire est plus ou moins grande et influence les interactions entre un homme et ses semblables[17].

Selon la vision de Hall, l'homme divise sa bulle d'espace personnel en 4 sous-bulles, chacune correspondant à des relations particulières (voir Figure 2.15) :

- **La zone intime.** De 0 à 50 centimètres[15], c'est la relation la plus proche que Hall considère qu'il puisse exister. A ce stade, la personne est capable d'acquérir des informations sur l'autre assez aisément. En effet, son semblable est à une distance visuelle où les détails sont distinguables, il est possible de percevoir ses paroles, même chuchotées, de capter jusqu'à la sensation de chaleur dégagée par les corps ou d'en sentir l'odeur. La personne est touchable sans qu'il faille étendre totalement le bras[17]. Il est important de noter que, mis à part circonstances exceptionnelles liées à un environnement contraignant, il faut une autorisation des gens pour entrer à ce point dans leur bulle d'espace personnel[17].

- **La zone personnelle.** De 0,5 à 1 mètre[15], l'homme laisse l'accès à cette bulle pour les amis et la famille. Souvent, l'autre personne se trouve à la distance d'un bras, il est encore donc possible de la toucher. Évidemment, l'homme est capable de voir et d'entendre la personne, cependant l'orateur devra augmenter légèrement le volume afin de se faire comprendre[17].

- **La zone sociale.** De 1 à 4 mètres[15], cette bulle laisse place à des relations et interactions plus formelles (comme dans le cadre d'une discussion entre collègues, au bureau). Dans ce cas, les personnes ne sont plus atteignables dans la distance du bras et il n'est plus possible de les toucher. Lors de conversations, l'orateur devra parler avec un volume plus élevé que précédemment et son ton sera beaucoup plus formel[17].

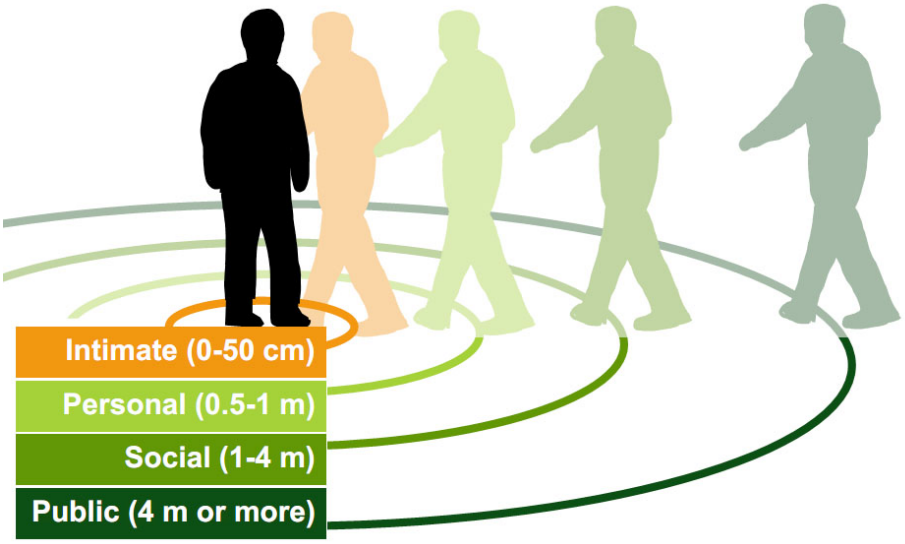

FIGURE 2.15: Catégorisation des distances d'espace personnel selon Hall[17].

- **La zone publique.** De 4 à 7 mètres[15] (et même au-delà), l'homme se situe plus comme un orateur face à une assemblée. Les éléments principaux pour glaner de l'information sont la vue et l'ouïe (à condition que le volume de la voix soit très amplifié)[17].

Si ces distances ne sont pas explicitement définies dans la vie courante et ne font pas l'unanimité auprès des différents sociologues[13], les réactions des personnes aux violations de ces limites sont, elles, très concrètes. Ainsi, l'homme va être sans cesse dans l'ajustement et l'adaptation en réaction à la présence et aux interactions de ses semblables. Plusieurs variables peuvent être modifiées dans le cadre d'une réaction aux intrusions. Ainsi, la distance, l'orientation ou le contact visuel sont des mécanismes simples mais perceptifs qu'une personne peut mettre en place en guise de réaction. De même, ces variables sont interchangeables, si pour une raison particulière une de ces variables ne peut être modifiée, l'homme va en corriger une autre pour maintenir l'équilibre[17]. Ainsi, par exemple, s'il est coincé dans un ascenseur bondé, qu'il ne peut modifier les distances interpersonnelles, il peut choisir de regarder le sol et d'éviter le contact visuel avec les personnes qui l'entourent.

Agencements des personnes et de l'espace

Au début de cette section, il a été évoqué que l'étude des interactions proxémiques pourrait mener, à terme, à une meilleure compréhension de l'agencement des espaces privés et des villes. Cette affirmation est possible car, comme il le sera expliqué ci-après, l'un et l'autre sont étroitement liés.

[13] Aiello, par exemple, considère qu'il n'existe pas un point de transition exact entre les zones et préfère voir une distance proxémique continue et dynamique propre à chaque être[17].

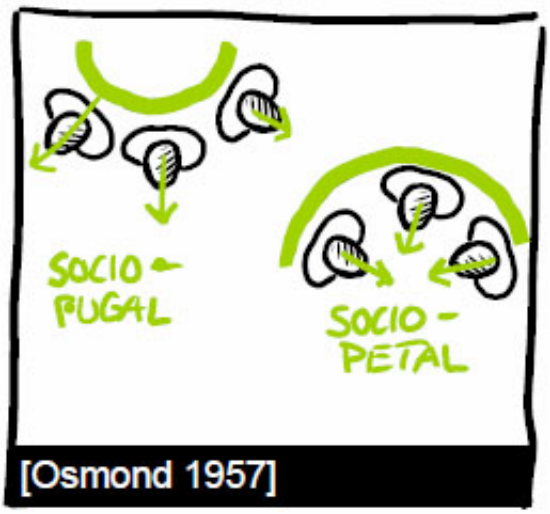

FIGURE 2.16: Schéma de l'agencement sociofuge et sociopète[17].

Plus tôt, il a été énoncé que les distances proxémiques entre les êtres humains n'étaient pas statiques et dépendaient de plusieurs variables. En effet, selon la culture d'une personne[14], son genre, son âge, sa personnalité et sa relation avec l'autre (amitié, connaissance, passé), elle sera plus ou moins prompte à laisser entrer cet autre dans sa bulle d'espace personnel. Cependant, il existe un dernier facteur qui influence cette donnée : l'agencement de l'espace[17].

Dans une pièce, il y a 2 facteurs qui en influencent l'agencement : les propriétés fixes et les semi-fixes. Les premières désignent tout ce qui est immobile, c'est-à-dire la forme de la pièce ou du bâtiment, la position de ses murs, portes et fenêtres, etc. Les secondes désignent tout ce qui caractérise une pièce mais qui peut être déplacé. Typiquement, cette proposition évoque les meubles tels que les tables, les chaises ou les canapés, par exemple[17].

Si l'homme ne peut modifier aisément la structure fixe d'une pièce, il peut ré-agencer le mobilier semi-fixe afin que le design soit plus propice à la tâche qu'il souhaite effectuer. Ainsi, certains agencements sont plutôt prompts à réunir les gens (sociopètes) tandis que d'autres tendent à les séparer (sociofuges)[17], tel qu'illustré à la Figure 2.16. Lors d'une interaction groupée, l'homme peut choisir le style d'agencement préféré. Souvent, cet agencement se fait de façon à ce que la tâche à accomplir se déroule de façon simple et efficace. Par exemple, il préférera une structure face-à-face pour une tâche de nature plutôt compétitive, côte-à-côte pour quelque chose de plus coopératif et d'un coin à un autre pour des conversations[17].

[14] Il a été prouvé que les personnes méditerranéennes et latines sont plus promptes au contact physique que les personnes issues de la culture anglo-saxonne[17].

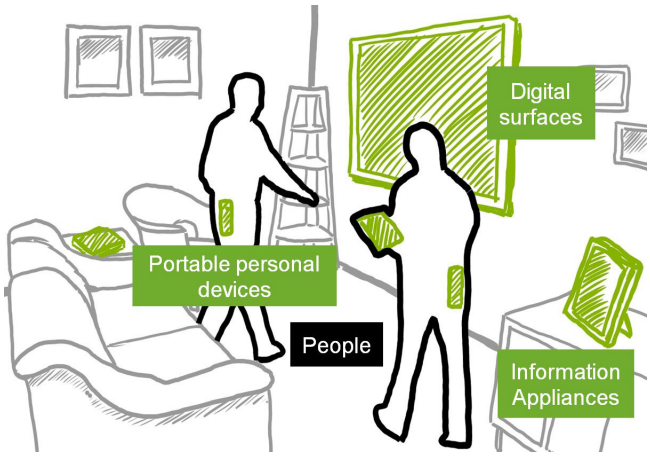

FIGURE 2.17: Informatique ubiquitaire avec des appareils de toute taille[2].

2.3.2 Interactions proxémiques et ubicomp

L'homme interagit, il le fait avec ses semblables, depuis toujours et de façon presque inconsciente. Mais que penser d'un monde où l'être humain interagirait avec les appareils électroniques de la même façon ? N'est-ce déjà pas le cas, lors de l'observation de la relation qui le lie à son ordinateur ou son téléphone portable ? Un monde où les relations de personne à personne, de personne à appareil et d'appareil à appareil seraient sur le même pied d'égalité est-il envisageable[2] ?

Informatique ubiquitaire

Pour tenter de répondre aux questions posées ci-dessus, il est nécessaire de comprendre ce qu'est l'informatique ubiquitaire (aussi appelée ubicomp). Ce terme est apparu pour la première fois dans les années 90, dans un article scientifique écrit par Weiser. Ce dernier y expliquait sa vision de l'informatique et y découpa celle-ci en 3 grandes ères. La première était celle des ordinateurs centraux caractérisée par un ordinateur pour plusieurs personnes ; la seconde était celle des ordinateurs personnels décrite par le fait qu'un ordinateur était désormais disponible pour une seule personne ; la troisième était une « prédiction » que Weiser appela informatique ubiquitaire et qui prévoyait qu'il y aurait plusieurs appareils digitaux pour une seule et même personne[17].

A l'heure actuelle où l'accès à l'information digitale est en hausse[15], il peut être affirmé sans se tromper que la vision de Weiser était correcte. L'homme possède des appareils de toute sorte et de toute taille, comme illustré à la Figure 2.17 (Weiser définit lui-même 3 échelles de classement allant des grands objets tels que des panneaux interactifs de plus d'un mètre aux petits appareils d'environ 2 centimètres qui peuvent être tenus dans une main en passant par les appareils de taille moyenne comme un ordinateur portable[17]).

Cependant, l'ère actuelle n'est pas encore pleinement dans l'ère ubiquitaire telle qu'elle est décrite par Weiser. En effet, dans la vision de ce dernier, il prédit une informatique embarquée à son apogée. Pour lui, dans l'ère ubicomp, l'ap-

pareil digital doit devenir un outil tellement intuitif et intégré qu'il en devient invisible. Il entend par là que l'utilisateur ne doit plus avoir conscience qu'il utilise un appareil digital pour accéder à l'information et que sa tâche doit se faire naturellement, telle qu'elle s'est toujours déroulée par le passé[17]. L'ubicomp doit prendre en compte l'environnement naturel de l'homme pour s'y intégrer de façon tellement fluide que l'ordinateur serait complètement caché dans le décor de cet environnement.

Bien que les systèmes ne soient pas encore totalement intuitifs et cachés dans les objets du quotidien, certains dispositifs possèdent déjà un stade de reconnaissance avancé et permettent alors deux modes d'interaction. Il y a les interactions explicites, très répandues au quotidien et qui nécessitent des actions de contrôle explicites initiées par l'utilisateur (emploi d'une télécommande, gestuelle à effectuer, commandes vocales tarabiscotées), et les interactions implicites, intuitives, où l'utilisateur se comporte de façon tout à fait naturelle (entrer dans une pièce, jeter un coup d'œil sur un écran, commandes vocales intuitives et simples) pendant que l'ordinateur en déduit les actions à effectuer dans ces situations (allumage d'un appareil, agrandissement d'une image, changement de chaine de télévision). Ce mode ne devrait pas être lié qu'à l'utilisateur mais initierait automatiquement la connexion avec les appareils digitaux à proximité afin de faciliter le transfert de données, par exemple. Idéalement, dans la vision d'un écosystème ubicomp parfait, les interactions intuitives seraient le seul mode existant et l'utilisateur utiliserait alors les systèmes de façon plus naturelle. Au lieu d'allumer la télévision par une succession de pressions sur une télécommande, le fait d'entrer dans un pièce et de s'asseoir devant l'écran pourrait déclencher son allumage, par exemple.

Intégration de la théorie proxémique

Il n'est pas facile d'atteindre la vision du futur telle que Weiser l'a décrite. L'implémentation fait face à plusieurs obstacles, parfois de taille. Marquardt en a relevé six[16] dont l'incitation et la facilité que l'homme a à interagir avec une technologie sans interface, la confusion dans les données perçues (ces données constituent-elles des interactions avec le système ou font-elles partie de la routine de l'utilisateur), le respect de l'intimité et de la sécurité tout en permettant l'établissement de connexions entre les différents appareils ou encore la gestion des erreurs d'interprétation dans des contextes délicats[16].

Il est possible de surmonter certains de ces obstacles en intégrant aux technologies émergentes les théories proxémiques. En appliquant les connaissances possédées concernant les interactions entre humains à la technologie, de nouvelles techniques d'interaction vont émerger[17]. Il sera alors possible de définir de nouvelles relations : de personne à appareil digital, d'appareil digital à appareil digital, voire même d'appareil digital à objet classique (fixe ou semi-fixe), comme il est possible de le voir sur la Figure 2.18.

Cette section va présenter quelles informations il faut tirer de la théorie proxémique et comment les appliquer à l'informatique ubiquitaire afin de créer des écosystèmes conformes à la définition de Weiser.

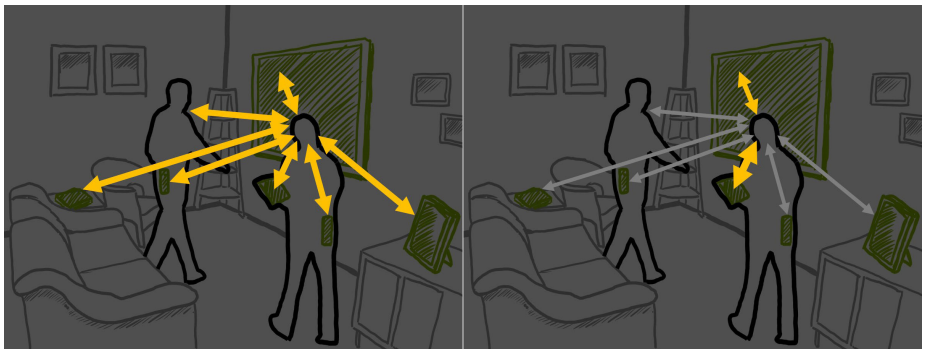

FIGURE 2.18: Interactions proxémiques complètes (entre les hommes, les appareils digitaux et le mobilier non digital) et un zoom sur un cas d'utilisation[17].

L'homme, dans ses interactions quotidiennes, est naturellement muni de capteurs (ses sens) et d'un système capable d'assimiler le contexte autour de lui (son cerveau). Cependant, l'ordinateur ne possède pas naturellement les connaissances nécessaires pour pouvoir appliquer et interpréter des interactions proxémiques. Il est donc nécessaire de lui fournir des senseurs ou des capteurs qu'il est apte à comprendre (ce point sera explicité plus loin), un système et des règles capables de situer le contexte et réagir face aux changements (ce qui sera appelé conscience de contexte).

Pour définir un contexte qui entoure un système, il faut capter 3 aspects différents : où se trouve la personne, avec qui se trouve-t-elle et quelles sont les ressources exploitables à proximité. Ces différentes données vont permettre de designer un environnement réactif qui va pouvoir induire des contextes d'utilisations et influencer les actions du système. Dès lors, c'est cette technologie qui sera utilisée afin de mettre en place un écosystème ubicomp capable d'exécuter des actions pro-activement (mode d'interaction implicite, tel que défini plus haut).

Néanmoins, l'implémentation d'une conscience de contexte n'est pas évidente. Un contexte est une donnée dynamique et instable, il n'est pas possible d'en identifier précisément toutes les informations qui le définissent. Par exemple, certains contextes sociaux dépendent d'un historique entre les personnes qui ne peut être perçu par le système, aussi bien conçu qu'il puisse l'être. De même, l'état d'esprit d'une personne qui rentre en interaction, ses émotions ou son objectif du moment sont des données qui ne sont pas percevables par de simples capteurs. Dès lors, il est impossible de créer des règles qui donneraient au système des indications sur ce qu'il doit interpréter dans tous les contextes possibles.

Greenberg recommande alors de ne pas simplifier un contexte à outrance en le rendant bénin et d'éviter d'employer des systèmes d'environnement réactif quand les actions et décisions d'interprétation sont délicates et peuvent mener à effectuer des actions inadéquates et risquées. Il préconise aussi que tout système explicite fortement les actions qu'il entreprend et qu'il soit ouvert à la réécriture manuelle dans le cas d'actions erronées.

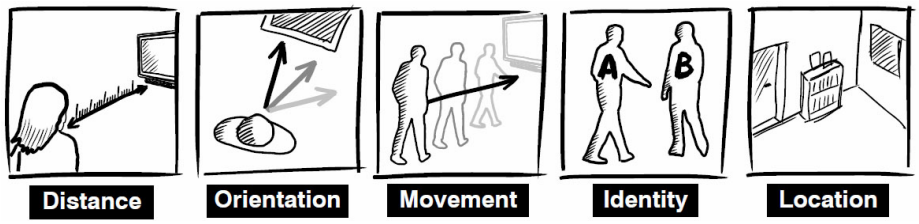

FIGURE 2.19: Les 5 mesures d'intérêt dans les écosystèmes ubicomp[17].

Cependant, la problématique de l'interprétation du contexte s'apparente plus à des problèmes d'intelligence artificielle et ne sera pas traitée plus en profondeur dans le cadre de ce travail.

Comme mentionné plus tôt, la technologie doit impérativement posséder des capteurs afin de pouvoir inférer un contexte. Ces capteurs sont divers et variés, ils peuvent consister en tags RFID, caméras et bien d'autres[15]. Lorsqu'un écosystème ubicomp d'une telle ampleur est créé, il faut s'intéresser à 5 mesures proxémiques en particulier (voir Figure 2.19), qui serviront de base pour tout le système[16] :

- **La distance.** Que ce soit entre les personnes, les appareils digitaux ou les objets simples, la distance se mesure entre les différentes entités. Elle peut être caractérisée par des catégories (comme les distances proxémiques qui sont divisées en 4 groupes, tel que dans la Figure 2.20) ou par une mesure précise de longueur. De même, elle peut être discrète ou continue ; ou encore elle peut être absolue ou relative. Lorsqu'on parle d'une position relative, on se contente de calculer la distance entre 2 entités distinctes en tout genre, dans le cas d'une position absolue, on définit un point fixe qui ne bougera jamais et la distance de chaque entité par rapport à ce point sera calculée.

- **L'orientation.** Cette mesure représente la direction que prend une entité. Pour une personne, cela peut être l'orientation de son corps, de sa tête mais aussi de ses membres ou son regard. Pour un objet, il faut définir un avant différentiable. D'autres faces peuvent être identifiables si les besoins nécessitent vraiment un tel niveau de sophistication. Cette métrique peut être donnée de façon qualitative (telle entité fait face à ceci ou telle entité est de dos, par exemple) ou de façon quantitative (angle précis). La seconde méthode permet de savoir quand deux rayons sont en intersection, ce qui peut être utile, par exemple, lorsqu'un groupe de personnes se font face, l'intersection de leurs rayons corporels définit la zone de travail consacrée à la tâche-objectif de ce même groupe.

- **Le mouvement.** Il est caractéristique du changement à la fois de position et/ou d'orientation et s'applique aux données tant bien relatives

[15] Une liste exhaustive des capteurs qu'il est possible d'employer se trouve dans le Tableau B.4

[16] Bien évidemment, d'autres mesures et capteurs peuvent être ajoutés en fonction des besoins, ne sont présentés ici que les besoins élémentaires dans le cadre d'un tel système.

qu'absolues. Le mouvement se calcule sur une période de temps et pas à un instant précis et permet de connaitre la vélocité de ce changement ainsi que l'accélération d'une entité. Cela permet de savoir, par exemple, comment une personne s'approche d'une autre entité quelconque et de répondre à des questions telles que celles-ci : ralentit-elle, lui fait-elle face, se tourne-t-elle subitement vers une autre entité ?

- **L'identité.** Il est primordial de pouvoir décrire chaque entité de façon unique. Cependant, le niveau de description d'une entité peut varier d'une caractérisation précise (telle personne, le téléphone de tel homme) à une identification vague (objet non digital). Il est possible que le niveau de détail se situe entre ces deux extrêmes en catégorisant l'entité (une personne, une chaise) ou encore de l'affilier à des groupes bien définis (membre de la famille, visiteur). L'identité de l'entité doit être définie selon les besoins du système, ainsi, il n'est pas nécessaire de reconnaitre précisément chaque personne qui passe devant un panneau publicitaire en rue, par exemple.

- **La localisation.** Différente de la distance/position, cette métrique concerne plutôt les aspects qualitatifs et quantitatifs du lieu où les interactions prennent place. Cela prend en compte l'agencement de l'espace (tel qu'il a été défini plus tôt dans ce travail) et le contexte de localisation. Cela doit être vu comme un tout qui fournit des méta-informations concernant les pratiques sociales dans ce lieu, le contexte d'utilisation du système. Par exemple, les interactions et le contexte seront différents selon que l'écosystème ubiquitaire se trouve dans un lieu de travail ou dans une maison.

Il faut cependant noter que l'intégration pour créer un environnement réactif ne se fait pas sur un coup de tête et qu'il faut suivre une certaine procédure logique de traitement des données. En effet, le système est désormais muni d'une technologie sensible capable de capter et traquer certaines données sur les entités (les inputs). La première information que le système va extraire et traiter sera les données de localisation, afin d'être informé du contexte et de l'agencement de l'endroit (quelles sont les données fixes et semi-fixes). Ensuite, le système va tenter d'identifier les entités présentes pour dresser une liste des différents acteurs. Puis, le système va traiter les informations de distance et d'orientation, bien souvent relatives, entre les différentes entités reconnues. De par ces deux données, le système va aussi être capable de traiter les données de mouvement. Collecter la distance, l'orientation et, de la sorte, le mouvement permet de ne pas se cantonner à une vision discrète de l'interaction et, dès lors, d'être plus précis quant à la gestion de l'attention de l'utilisateur[30]. A ce stade, il est en possession des 5 métriques telles que définies ci-avant et il va pouvoir commencer le travail d'interprétation via les règles de comportement qui lui ont été définies. Après tout ce processus, le système va délivrer une réponse adaptée (output). Ce déroulement est synthétisé de façon schématique à la Figure 2.21.

Un système bien travaillé et qui suit les étapes expliquées tout au long de cette partie permettra de fournir une technologie ubiquitaire qui s'intègre de mieux en mieux à la vision de Weiser. Il faut cependant ne pas perdre de vue

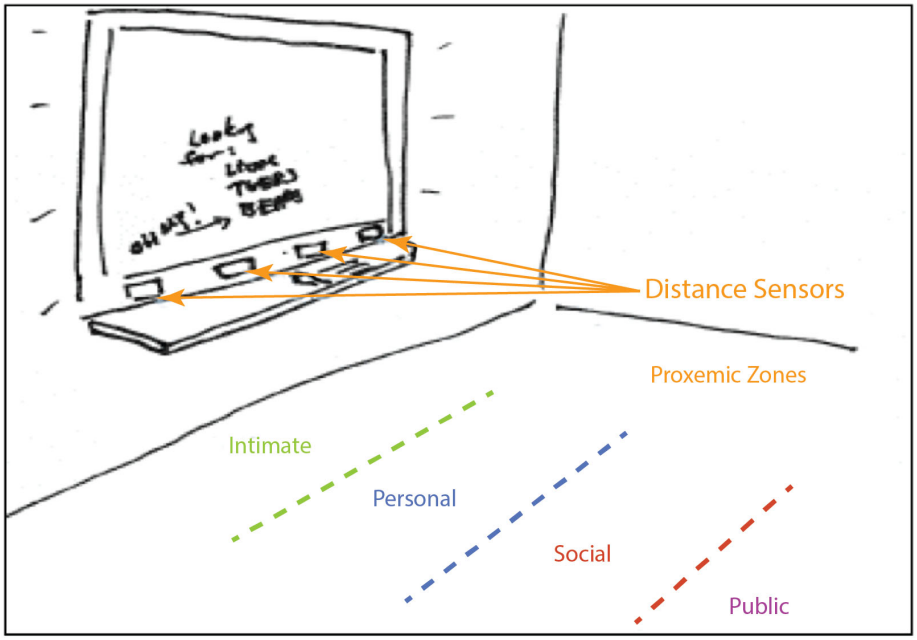

FIGURE 2.20: Capteurs de distance et zones proxémiques[17].

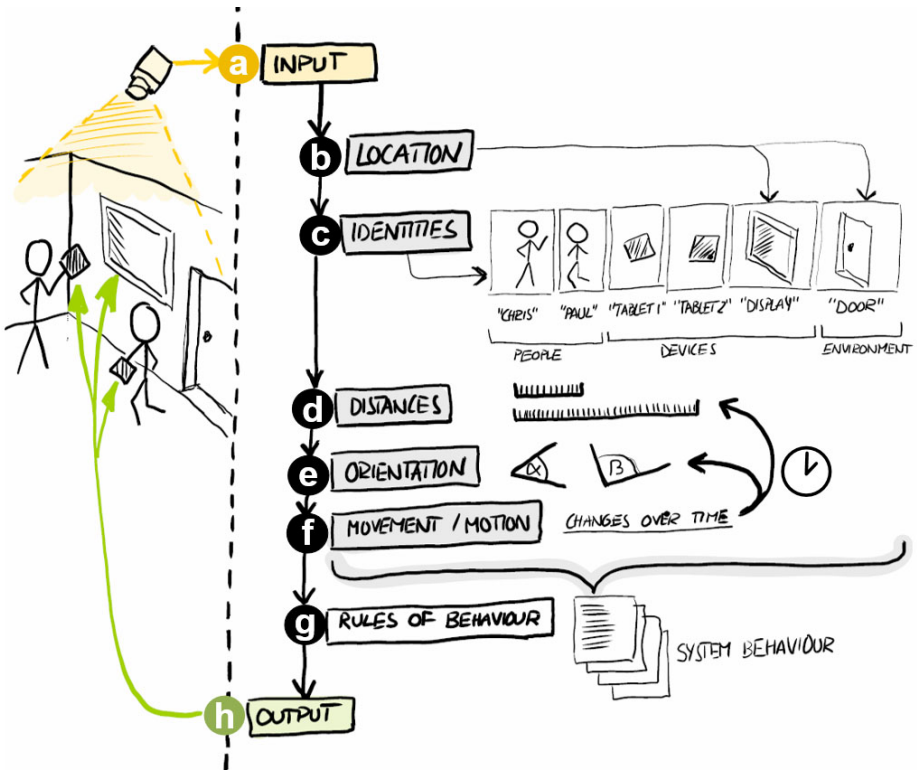

FIGURE 2.21: Schéma de déroulement d'un système ubiquitaire proxémique[17].

les différentes problématiques abordées au long de cette section pour créer un système qui ne sera pas frustrant à employer.

2.3.3 Interactions proxémiques et le regard

Bien que l'intégration des interactions proxémiques à l'informatique semble une bonne idée, il y a quelques désagréments aux techniques d'interactions que cette combinaison propose. En effet, dans le cas d'un panneau interactif public, par exemple, certaines personnes pourraient être intimidées d'aller au devant de ce panneau pour le toucher et interagir, voire même se sentir ridicules de faire des mouvements dans les airs pour donner des instructions à celui-ci[12]. Il peut être intéressant d'envisager un moyen plus discret d'interagir avec un écosystème ubicomp : y intégrer des techniques d'eye tracking.

Les avantages à traquer le regard et l'attention visuelle sont multiples. Comme mentionné ci-dessus, le regard est quelque chose de discret qu'il est facile de diriger. De plus, il reflète bien le niveau d'attention d'un utilisateur face à un système[12]. En effet, en captant simplement les relations de distance et d'orientation d'un utilisateur potentiel face à un système, et que ce dernier ignore totalement l'interaction, le chercheur est alors en droit de se demander si le contenu a été vu et ignoré ou si, justement, l'utilisateur est passé à côté de la fonctionnalité interactive. De plus, ces données lui permettent de savoir quelles étaient la réaction et l'appréciation de l'utilisateur face à ce contenu[30]. Finalement, le regard est parfois aussi plus précis que la gestuelle[1].

De par le nombre limité de références relatives au sujet, il est difficile de citer tous les avantages que présentent l'intégration de l'eye tracking à un système ubicomp proxémique. De plus, si le niveau d'attention visuel a été étudié au travers de tels systèmes[29], il est plus rare d'utiliser la SA comme une input dans les interactions proxémiques. Il faut alors jouer de son imagination et de son esprit créatif afin de voir que la combinaison de ces technologies est intéressante à bien des niveaux.

Les challenges

L'intégration des techniques d'eye tracking et de SA, telles que définies plus tôt dans ce chapitre, aux interactions proxémiques et à l'informatique ubiquitaire n'ont cependant pas encore été beaucoup exploitées au cours des dernières années. Cela peut sembler surprenant puisque tous ces concepts ne sont pas récents et que les prix du marché se démocratisent petit à petit. Néanmoins, une telle prouesse demande de surmonter de nombreux challenges, notamment dans le cadre de systèmes intégrés dans les lieux publics.

Le premier et plus gros problème rencontré dans le cadre de ce couplage est la calibration. Ce problème est pointé par de nombreux travaux s'étant attaqués à l'intégration du regard dans les écosystèmes ubicomp. En effet, lorsque le système présente une interaction dans un endroit restreint (maison ou bureau), il est aisé de calibrer le regard des utilisateurs et de conserver ces données puisque les personnes sont régulières et identifiées. Cependant, dans le cadre d'un écran public, par exemple, il n'est pas possible et nécessaire d'identifier

chaque personne très précisément. De plus, un passant ne va pas prendre le temps de s'arrêter devant l'écran et perdre de précieuses minutes afin de calibrer le tracker avant d'enfin pouvoir découvrir ledit environnement interactif. Si certains chercheurs ont préférés miser sur une calibration implicite et donc plus rapide[13], la solution la plus envisagée est de complètement sauter la calibration, trop consommatrice de temps[4]. Évidemment, cela entraine une perte de précision et de possibilité de mouvements captés[7].

Cette solution amène alors la seconde problématique : la précision[4]. En effet, un eye tracker ne permet pas de pointer exactement ce que l'utilisateur regarde avec une fiabilité totale. Souvent, cependant, ce décalage est vraiment mineur : pour un eye tracker distant, il y a une marge d'erreur de plus ou moins 0,5°[4]. Dans le cas d'un appareil mobile (comme les lunettes), la marge est plutôt de 1°, car il faut gérer les distances et le mouvement de l'utilisateur[4]. Dans le cas de l'emploi d'une webcam dans les interactions proxémiques, la marge d'erreur peut même atteindre les 10cm d'écart avec le point réellement regardé par l'utilisateur[7].

De même, du fait de l'absence de calibration et de ce manque de précision, le mouvement de l'œil ne peut être traité et lié à l'écran que d'une certaine façon. Ainsi, il y a une perte de verticalité dans le contrôle de l'ubicomp par le regard[33]. De plus, en lien avec les imprécisions mentionnées ci-avant, le regard ne peut être lié que relativement à l'écran et non plus absolument, comme c'est le cas lorsque l'utilisateur est seul devant son ordinateur. Dès lors, une solution possible est de lier le regard sur 3 positions : la gauche, le centre et la droite[7].

Une autre problématique évoquée est le guidage de l'utilisateur. Placer un panneau interactif dans un lieu public peut être une bonne chose, mais l'utilisateur peut ne pas être conscient de l'emploi de ce dernier. Évidemment, l'utilisateur ne peut pas suivre un entrainement a priori pour être préparé dans la possible éventualité de rencontrer un tel panneau[33]. Le dispositif doit donc guider l'utilisateur pour lui indiquer quel comportement adopter pour interagir. Par exemple, dans le projet concernant un eye tracker mobile qui suit le déplacement de l'utilisateur, il faut informer de façon autonome à cette personne qu'elle peut continuer à se déplacer totalement normalement, tout en regardant l'écran, sans que cela n'empêche l'interaction. En effet, certains s'imaginaient devoir marcher en crabe pour le bon fonctionnement du panneau[12].

Enfin, un dernier point est à soulever : le problème de Midas. Ceci relève de la difficulté de différencier un mouvement de l'œil intentionnel afin de créer une interaction et un mouvement autre (par exemple parce que l'utilisateur est attiré par un événement en dehors du cadre de l'expérience d'activité)[4]. Pour tenter de résoudre cette problématique, il faut coupler le regard avec d'autres données, d'interactions proxémiques par exemple.

Dans le chapitre suivant, il sera expliqué comment ce travail a tenté de surmonter certains de ces challenges tout en essayant d'intégrer les principes d'eye tracking et de SA aux interactions proxémiques.

Chapitre 3

Implémentation et résultats

Ce travail a pour but de présenter l'intérêt de l'utilisation du regard dans un écosystème ubicomp dirigé par des interactions proxémiques. Précédemment, différents aspects théoriques ont été abordés de façon assez distincts les uns des autres et ne permettant pas de clairement comprendre l'intégration de toutes ces parties ensemble.

Afin d'amener à une meilleure compréhension de cette intégration complète et complexe, un cas d'utilisation a été implémenté. Le contexte de cette implémentation était de réaliser un panneau interactif placé dans un lieu public régi par les principes de l'interaction proxémique auquel la théorie de l'eye tracking avait été intégrée.

Cependant, des hypothèses ont dû être émises afin de simplifier le cas d'utilisation, pour des raisons expliquées ci-après. Dès lors, la réalisation s'en est alors trouvée limitée au cas suivant : un petit panneau restant dans un cadre d'utilisation privé mais qui peut être dirigé par le regard de l'homme.

L'implémentation de cet exemple sera détaillée dans cette partie du travail au travers de trois grandes étapes. Tout d'abord, l'organisation du travail technique sera expliquée afin de séparer et catégoriser ce développement en deux grandes phases : le stage et l'après stage. Ensuite, le développement de chaque partie sera détaillé en deux phases, d'abord le choix de la technologie employée et enfin la partie d'implémentation qui permettra d'expliquer les spécificités de réalisation plus amplement.

3.1 Organisation

Dans le cadre de ce travail, une partie d'implémentation a dû être mise en place. Cette partie a permis l'acquisition de connaissances concernant le sujet traité en plus de la réalisation d'une partie logicielle, et il est important d'en décortiquer quelque peu son organisation.

Ce travail s'est donc implémenté en deux grandes phases, la première étant le stage réalisé à l'Université du Texas d'Austin, dans la faculté d'Information,

et la seconde étant une adaptation de ce premier travail dans le cadre des interactions proxémiques.

Comme expliqué ci-avant, la première partie du travail a pris place pendant la phase de stage. C'est au cours de cette phase que les connaissances concernant l'eye tracking et la SA ont été acquises. En effet, le produit réalisé lors de ce stage était un script Python permettant de capturer le regard et d'en mesurer, en temps réel, les données nécessaires afin de calculer la SA telle que définie théoriquement dans ce travail.

La seconde phase a pris place lors de la fin du stage et pendant toute la seconde partie de l'année. Elle consistait en l'adaptation du travail réalisé précédemment sur d'autres technologies, puis, en l'intégration des interactions proxémiques afin de permettre la création d'un écosystème unbicomp dirigé par le regard. Cette phase a posé quelques problèmes d'ordre technique quant aux emplois de technologies possibles, ce qui sera expliqué dans les sections suivantes.

La suite de ce travail présentera les explications concernant le travail réalisé pendant le stage de façon intégrale et puis repartira sur l'après-stage pour montrer comment la première implémentation a pu être incorporée dans la seconde.

3.2 Eye tracking et SA

La première partie du travail consistait en une phase de recherches théoriques permettant d'apprendre à utiliser un eye tracker, quelles techniques étaient couplées à son utilisation et ensuite d'y incorporer des mesures de SA intéressantes. Le but de ce travail était de fournir un outil de recherche combinable à iMotions, un logiciel d'analyses biométriques complexe servant à faire des études UX poussées et fortement employé sur le lieu de stage. L'outil devait présenter les deux fonctionnalités suivantes : l'ajout de calcul de SA que iMotions ne faisait que partiellement et la possibilité de réaliser ces analyses en temps réel, ce qu'aucun logiciel ne fournit actuellement. Ce complément devait être le plus silencieux possible, c'est-à-dire qu'une fois lancé, l'utilisateur et le chercheur n'avaient pas à se préoccuper de l'outil et pouvaient pleinement se focaliser sur l'étude menée via iMotions.

Dans cette partie, il sera expliqué comment cet outil a été réalisé en montrant d'abord les choix technologiques pertinents qui ont été faits et, ensuite, en expliquant le développement et l'implémentation même de ce travail. À titre d'information, certaines parties du code source présentant des innovations importantes se trouvent en Annexe C, mais il n'est pas nécessaire de les présenter plus en détail.

3.2.1 Technologie employée

Afin de réaliser ce travail, différentes technologies ont été envisagées et comparées avant de faire un choix. Dans cette partie, il sera expliqué comment le choix s'est porté entre les différents matériels, langages de programmation et

librairies employés.

La première partie de ce travail s'est focalisée sur la compréhension et l'emploi d'un eye tracker, puis, la maitrise de la SA. Le matériel fourni a donc été un eye tracker distant intégré à l'ordinateur de la marque Tobii[1]. Cet appareil, bien que n'étant plus de dernière génération, est un appareil d'une qualité supérieure qui permet l'analyse de données de façon très pointue et poussée. Grâce à son taux d'échantillonnage de 300Hz, il permet de détecter très en détail les fixations et saccades oculaires ainsi que d'identifier les clignements de l'œil. Une telle précision dans le cadre de recherches scientifiques poussées est un avantage considérable.

Au niveau du langage de programmation utilisé, c'est Python qui a été choisi. En effet, ce langage a pour avantage d'être compatible avec de nombreux eye trackers, notamment ceux de la marque Tobii, sous peine d'avoir une licence professionnelle. Cette licence, uniquement valable pour le langage Python, permet l'accès à des données plus poussées que d'autres SDK fournis par la marque Tobii, comme il le sera expliqué plus loin.

Le langage utilisé permet de se connecter directement à l'appareil et, donc, de collecter les données en temps réel afin de ne pas passer par un logiciel externe qui pourrait induire du décalage ou filtrer ces données. Cependant, cette solution a pour désavantage que, puisque les données sont fournies en temps réel, le SDK de Tobii ne permet pas de distinguer les fixations, des saccades ou des clignements de l'œil. Il y a donc une perte de l'analyse détaillée que peut fournir une telle technologie couplée à la licence professionnelle.

Pour palier à cela, une librairie non négligeable a été utilisée : PyGaze[2]. Cette solution est une des seules qui permet d'analyser les données de regard en temps réel. Ses algorithmes sont assez simples à comprendre et faciles d'utilisation ce qui a permis de s'y habituer et de la maitriser assez rapidement, à défaut d'avoir une documentation bien détaillée. De plus, cette librairie n'est pas limitée sur son emploi avec les eye trackers de la marque Tobii mais fonctionne aussi bien avec certaines autres marques[5], ce qui offre un possible changement de matériel sans devoir réécrire tout le code. Cependant, cette librairie présente deux désavantages assez conséquents : calculs analytiques moins précis que ceux fournis par Tobii ou iMotions[3] et un manque de stabilité.

En effet, bien que les méthodes définies dans PyGaze permettent de différencier fixation de saccade et même du clignement de l'œil, il se base sur un calcul simple de proximité. Cela signifie que son calcul se contente juste de vérifier si le regard se trouve dans une certaine zone autour du point de départ pour une période de temps définie. Cependant, malgré la présence, au niveau du calcul de saccade, de la vélocité du mouvement de l'œil, ces algorithmes ne sont pas basés sur les standards I-VT pour les fixations[21, 22, 23].

[1] Tobbi Pro TX300

[2] `https://www.pygaze.org/`

[3] Un logiciel d'analyses biométriques.

Quelques librairies supplémentaires ont été nécessaires pour la réalisation de ce travail mais moins importantes et beaucoup plus traditionnelles, ainsi elles ne seront que citées. Les principales librairies employées ont été Numpy, PyGame et PsychoPy en plus de celles nécessaires au bon fonctionnement de PyGaze et le SDK Pro de Tobii.

3.2.2 Développement et produit fini

L'implémentation de ce travail peut être découpée en trois grandes parties : la connexion à iMotions et aux AOI, le lancement de l'eye tracker et enfin le calcul en temps réel de la SA. Dans cette section, il sera expliqué comment fonctionne le programme, quelques subtilités d'implémentation importantes ainsi que les tests de validation qui ont été mis en place.

Fonctionnement général

Pour faire fonctionner le script Python, il est impératif que des AOI soient définies. En effet, comme cela a déjà été mentionné de façon théorique dans le chapitre précédent, et comme il le sera expliqué de façon plus approfondie par la suite, la notion de SA dépend très fortement de ces zones et il n'y a d'intérêt à la calculer si aucune AOI n'est définie. Afin que l'outil puisse extraire les données nécessaires à la définition de ces zones d'intérêt, il faut lui fournir un dossier (à définir dans le code selon les préférences) contenant des fichiers XML portant un nom standard. Ces fichiers sont en réalité des templates générés par iMotions lors de la définition d'AOI et qu'il suffit d'exporter au bon endroit.

Le bon fonctionnement de ce script dépend aussi du fait qu'iMotions soit déjà ouvert sur un réseau accessible par l'ordinateur qui fera tourner le script[4]. Comme l'outil a pour vocation d'être un complément dudit logiciel, il est nécessaire que cette connexion soit faite afin que les deux programmes travaillent sur les mêmes input au même moment. Une fois qu'iMotions est ouvert, le script peut être lancé, et lorsqu'il tourne en arrière plan, l'étude du regard peut être lancée sur iMotions.

Cette manipulation permet de rajouter l'étape du lancement du script Python sans pour autant changer totalement les habitudes d'étude des chercheurs voués à l'employer. Cela permet de rester sur un outil discret et silencieux qui tourne en arrière plan.

Pendant toute l'exécution du script, la SA est calculée en temps réel. Cela permet aux personnes en charge de l'étude d'avoir immédiatement les résultats pour chaque action que l'utilisateur pose et dès lors d'adapter les stimuli de façon nécessaire si besoin. Ce calcul sera explicité un peu plus au point suivant.

[4]Il existe une version du code source qui ne dépend pas d'iMotions, cependant, cette version est assez instable de par la manière dont PyGaze appelle PyGame ou PsychoPy. Néanmoins, cette version est tout à fait capable de tourner seule.

Calcul de la SA

Comme expliqué précédemment, le but du travail était de fournir des mesures de SA en temps réel. Dans le chapitre précédent, ces données intéressantes ont été expliquées théoriquement. Cette partie ajoutera un côté plus concret et pratique sur le calcul de ces données.

La première mesure était le dwell time et représentait le temps passé au sein d'une certaine AOI. La méthode va donc calculer pour chaque AOI son temps de séjour de la façon suivante :

$\sum fixT_{AOI}$, où fixT est le temps de fixation.

La seconde donnée qu'il était souhaitable d'avoir était le calcul de l'entropie. Pour bref rappel, il s'agissait de la disparité du regard et se mesurait sur une échelle allant de 0 à 1. Pour calculer cette dernière, il fallait d'abord définir des matrices de transition :

$$\begin{bmatrix} (AOI_0, AOI_1) & numFix_{0,1} \\ (AOI_1, AOI_2) & numFix_{1,2} \\ (AOI_2, AOI_3) & numFix_{2,3} \\ ... & ... \end{bmatrix}$$

Où $numFix_{i,j}$ est le nombre de fixation entre les AOI_i et AOI_j. Ensuite, les matrices de transitions étaient transformées en matrices de probabilités :

$$\begin{bmatrix} (AOI_0, AOI_1) & proba_{0,1} = numFix_{0,1} / \sum numFix \\ (AOI_1, AOI_2) & proba_{1,2} = numFix_{1,2} / \sum numFix \\ (AOI_2, AOI_3) & proba_{2,3} = numFix_{2,3} / \sum numFix \\ ... & ... \end{bmatrix}$$

Où $\sum numFix$ représente le nombre total de fixations. On divise le nombre de fixations par AOI par le nombre total de fixations pour avoir la probabilité d'avoir une transition entre les AOI_i et AOI_j. Une dernière modification est effectuée sur les matrices avant d'être apte à calculer l'entropie :

$$\begin{bmatrix} (AOI_0, AOI_1) & \log_2(proba_{0,1}) \\ (AOI_1, AOI_2) & \log_2(proba_{1,2}) \\ (AOI_2, AOI_3) & \log_2(proba_{2,3}) \\ ... & ... \end{bmatrix}$$

Enfin, l'entropie est calculée comme qui suit :

$$entropie = -1 * \sum proba_{i,j}$$

La troisième mesure était la distance entre les fixations. Pour ce faire, il suffit de prendre deux fixations qui se suivent, d'en extraire les coordonnées et y appliquer une formule de distance cartésienne :

$$x_i = x_{fix_i} - x_{fix_{i-1}}$$
$$y_i = y_{fix_i} - y_{fix_{i-1}}$$
$$distance_i = \sqrt{x_i^2 + y_i^2}$$

La dernière mesure de SA calculée est la distance totale parcourue par le regard de l'utilisateur. Pour ce faire, il faut calculer préalablement la distance entre les AOI. Comme la majeure partie des distances parcourues par le regard sont les transitions entre ces zones d'intérêt, il n'y a plus qu'à multiplier le nombre de transitions qu'il y a entre ces AOI (calculé précédemment par la matrice de transition) et la distance qu'il y a effectivement entre ces zones. Ainsi, à la fin, il suffit d'additionner toutes les distances parcourues lors des transitions pour avoir la distance totale :

$$distTot = \sum transition_{i,j} * distance_{i,j}$$

Une fois que les formules mathématiques ont été posées, il était aisé de les implémenter comme partie intégrante du code.

```
Participant name: 'Amélie'
Shape not accepted
Shape not accepted
Dwell time = [897.4622413899997, 0, 0]
Entropy = 0
Distance between fixation = [14.142135623730951, 69.3757882838098, 63.63961030678928, 300.9269014229203]
Distance between AoIs = []
```

FIGURE 3.1: Résultats partiels fournis par le script.

La Figure 3.1 montre les résultats tels que présentés lors de l'exécution du script. Lors du lancement de ce dernier, l'utilisateur doit rentrer son nom. Ensuite, les AOI sont encodées depuis les fichiers XML. A savoir que PyGaze n'accepte que certaines formes d'AOI (les rectangles et les ellipses). Ceci explique la présence des lignes 2 et 3 qui, à titre indicatif pour les meneurs d'expérience, signalent que certaines formes d'AOI fournies depuis iMotions ne seront pas prises en compte étant donné la non-conformité de celles-ci. Enfin, les lignes suivantes présentent les résultats du calcul de SA. Ce script s'exécutant en temps réel, ces calculs sont mis à jour toutes les secondes. Cependant, par souci de lisibilité, seuls les premiers résultats ont été affichés dans la capture d'écran.

Le dwell time est affiché sous forme de liste où chaque temps est classé en fonction de l'AOI correspondant. Ainsi, dans cet exemple, l'utilisateur aura fixé pendant près d'une seconde entière la première zone d'intérêt. Ce manque de mouvement de la part de l'utilisateur se traduit par une entropie à 0 et des distances entre les fixations très petites. Ces données sont d'ailleurs, elles aussi, présentées sous forme d'une liste où chaque élément représente la distance parcourue entre une fixation et sa suivante (ainsi, l'élément 1 présente la distance parcourue entre les 1re et 2e fixations). Cette remarque est aussi valable pour la distance entre les AOI, si ce n'est que ce ne sont pas sur les fixations que sont triés les éléments mais bien entre les zones définies.

Validation

Si ce script n'a pas pu être employé par des utilisateurs réels lors d'études UX, il a tout de même été testé de 3 façons différentes afin de garantir la validité des résultats renvoyés et permettre son utilisation optimale. La fonctionnalité principale à tester était le calcul de la SA. En effet, les données de fixations dépendent de la librairie PyGaze et le parsing d'un fichier XML s'est avéré correct assez rapidement, via des comparaisons à iMotions.

La première source de comparaison de ces résultats était iMotions. En effet, bien que ce logiciel ne permette pas le calcul de SA, il renvoie tout de même les données de dwell time. Si le logiciel ne permet pas le calcul en temps réel et que le script développé fonctionnait à l'aide d'une fenêtre de temps coulissante, il a tout de même été possible, à l'aide d'un calcul supplémentaire, de comparer les dwell time mesurés afin de savoir si le script renvoyait des données avoisinant celles de iMotions.

Par ailleurs, au cours des tests d'exécution du script, des données réelles de positions de l'œil sur l'écran et de temps de fixations ont été récupérées afin d'être calculées manuellement et permettre la comparaison avec les résultats obtenus.

Enfin, des tests informatisés ont été écrits. Lors des recherches théoriques menées sur les métriques intéressantes de SA, des exemples de données ont été trouvés. Dès lors, il suffisait d'écrire des tests prenant les inputs des exemples et comparant leurs outputs avec les résultats calculés par le script. A la fin des comparaisons, les résultats étaient identiques au millionième degré, ces différences étant dues à des approximations et arrondis.

Grâce à ces méthodes de validation des résultats, il a pu être affirmé que le travail réalisé au niveau de la SA était correct et que les données fournies en output pouvaient donc être fiables.

L'objectif de cette partie a bien été rempli puisqu'il permettait d'effectuer la tâche demandée, bien que le script présente des améliorations comme le passage de PyGaze à une librairie ou des méthodes « maison » permettant d'inclure les standards I-VT dans l'estimation des fixations. Cependant, ce travail a été laissé pour une équipe de développement interne au lieu de stage.

Ceci conclut la réalisation d'un premier travail qui couvre le stage réalisé dans les premiers mois. La partie qui suit expliquera plus en détail l'intégration proxémique du regard.

3.3 Intégration des interactions proxémiques

Cette seconde phase de la partie pratique du travail couvre la réalisation technique qui lie ce qui a été précédemment produit avec la SA à l'interaction proxémique. Elle consiste en la recherche et documentation sur l'interaction proxémique et l'informatique ubiquitaire, puis en l'adaptation et l'intégration du travail mentionné ci-dessus afin de fournir un produit cohérent.

Le but de cette intégration n'était pas de fournir un produit fini et prêt à l'emploi. En effet, elle consistait plutôt en la réalisation d'un prototypage rapide d'une solution accessible au grand public afin de tester rapidement un setup.

Dans cette partie, il sera expliqué comment l'outil fini a été réalisé. Tout d'abord, il y sera détaillé quelles sont les technologies possiblement envisageables, les motivations du choix et de l'utilisation de certaines. Ensuite, le travail final sera présenté afin de mettre en avant les hypothèses qui ont été posées et ce qu'il a été possible de réaliser.

3.3.1 Technologie employée

Lors de la transition entre les deux réalisations, afin de pouvoir continuer le travail effectué, le matériel a dû être changé pour diverses raisons de coûts et de fin de production de la technologie. Cependant, afin de pouvoir intégrer les notions d'interactions proxémiques à l'eye tracking, il a fallu trouver une alternative envisageable financièrement et technologiquement. Plusieurs possibilités ont été envisagées, de l'eye tracker de moins bonne qualité à la webcam, et une a été retenue. Dans cette partie, les différents matériels ayant été employés seront présentés avec leurs avantages, leurs inconvénients et les choix qui ont motivé la décision actuelle.

Eye trackers

Dans un premier temps, afin de mener à bien cette étude, la technologie la plus évidente envisagée a été l'eye tracker. Deux choix ont été étudiés et seront présentés ci-dessous.

Comme mentionné précédemment, le Tobii Pro TX300 était le premier appareil mis à disposition. Bien que fort complet et précis dans les analyses qu'il propose, la production de ce dernier a été stoppée et le modèle est considéré comme étant dépassé. De plus, un tel dispositif présentait de sérieux désavantages comme le fait qu'il soit intégré à l'ordinateur ou que, de par son haut taux d'échantillonnage et sa précision importante, il n'autorisait que peu de mouvements de la part de l'utilisateur. Tout ceci posait alors de fortes contraintes concernant le souhait d'étudier les interactions proxémiques, notamment dans le cadre d'un panneau présent dans un large environnement public où les gens vont et viennent et ne restent pas fixes à une position indiquée. En conclusion, tout suggérait que l'emploi de cet appareil était peu indiqué dans le cadre de cette étude.

Cependant, la marque Tobii possédant une renommée certaine dans le milieu des eye trackers et proposant des dispositifs moins contraignant, le choix s'est posé sur un appareil à petit budget qui semblait tout à fait indiqué dans le cadre de cette étude : le Tobii 4C. Ce petit tracker distant indépendant conçu pour le gaming demande une précision beaucoup moins importante qu'un appareil taillé pour les analyses précises. De plus, il importait surtout que l'appareil soit capable de détecter le regard et d'en renvoyer les coordonnées. Sa précision et son aspect gaming garantit que l'utilisateur peut donc détourner le regard de ce dernier, ou bouger plus fortement la tête sans que celui-ci ne perde le fil des données capturées[5].

Bien que ce dernier pouvait présenter toutes les caractéristiques plus que nécessaires pour l'exécution de ce travail, il n'était pas compatible avec le code précédemment écrit. En effet, pour travailler sur un eye tracker de la marque Tobii en Python, il faut acheter une licence professionnelle, comme mentionné plus tôt dans ce travail. Or celle-ci présente un coût assez déraisonnable. Une alternative avait été explorée, mais n'avait abouti à rien, en utilisant les SDK compatibles avec le C# de par le manque de documentation mise à la disposition des développeurs et du manque de réponse de la part du support de la marque.

Webcams

Par manques de moyens, il n'était donc pas envisageable d'acheter un eye tracker d'une autre marque qui ne garantirait toujours pas le bon fonctionnement du projet. Lors de la lecture d'articles, à la fois sur les dispositifs de tracking et à la fois sur les interactions proxémiques[33], il est apparu qu'il était possible de réaliser ce procédé via la webcam.

De plus, l'emploi d'une webcam, dans le but de réaliser le tracking d'un œil, dans ce contexte très précis, peut tout à fait se justifier de façon judicieuse. En effet, cette technologie n'est pas dépendante d'une calibration forte, et il est possible d'outrepasser cette exigence. Cela semble plutôt intéressant puisque dans le cas d'étude de l'interaction proxémique, l'utilisateur ne souhaite pas perdre quelques minutes de son exploration visuelle pour calibrer la machine. Par ailleurs, de par son manque extrême de précision et son champ de détection plus large, l'utilisateur est plus libre quant à ses déplacements et mouvements de la tête. Enfin, de nombreux projets liés à l'intégration des interactions proxémiques utilisent déjà des webcams en guise de capteur, l'intégration du regard dans les métriques du proxémique ne poserait pas de problèmes de coûts supplémentaires et d'adaptation du matériel.

Pour réaliser un tel défi, il était évidemment impossible d'envisager d'entamer le développement d'un logiciel capable de détecter l'œil et son mouvement depuis une webcam. En effet, ce procédé relève de techniques d'intelligence artificielle pour lesquelles les ressources, les compétences et le temps étaient clairement manquant.

[5]Cette capacité a été testée lors des premiers jours d'acquisition du tracker.

Comparatif des librairies

Dès lors, différentes librairies existantes ont été explorées. Ces dernières seront alors présentées en quelques mots avant d'aborder leurs points positifs et négatifs. Un résumé de ce comparatif des librairies existantes se trouvera dans le Tableau 3.1.

La première librairie a avoir été creusée était Gaze Tracking Library[6], une librairie open-source au fonctionnement similaire à un eye tracker et se basant sur les réflexions rétinienne et cornéenne afin de traquer le centre de la pupille. Cette dernière nécessitait l'emploi du C# ainsi qu'un framework .NET pour fonctionner. Cependant, le projet n'ayant plus été mis à jour depuis 2015, il a été remarqué que le projet n'était plus compatible avec les technologies actuelles.

La librairie suivante est celle de XLab[7]. Cette dernière offre un tracking de l'œil pour des applications web. Son fonctionnement basé sur du machine learning rend l'approche plus qu'intéressante puisque les développeurs mettent à disposition du grand public une technologie basée sur des algorithmes puissants. Cependant, cette dernière ne peut être employée que via un plug-in du navigateur web et ne permet pas le développement d'une application logicielle.

GazeRecorder[8] et GazePointer sont deux logiciels qui ont été envisagés. Ils sont tous les deux supportés par GazeFlow et permettent plus ou moins la même chose : le contrôle de la souris à l'aide du regard. GazeFlow, l'outil de tracking, montre des résultats très satisfaisant, avec une différence de précision comparé à un eye tracker (SMI) de $1°$[9]. Cependant, aucun des logiciels que supporte la technologie de tracking n'est munis d'un SDK ou d'une API utilisable.

Présentant le même problème que les deux logiciels précédents, EyeTwig[10] ne pouvait pas non plus présenter une solution envisageable. De plus, le développement dudit logiciel a été stoppé pour les systèmes d'opérations Windows, limitant alors la portabilité et l'accessibilité de ce dernier.

OpenCV[11] permet de réaliser un tracking de l'œil via « haarcascade ». Cette dernière est intéressante puisque existe pour Python, ce qui permettait de repartir sur la base créée pendant le stage. De plus, cette librairie est entrainée via du machine learning, pour la reconnaissance des caractéristiques faciales. Après plusieurs tests, il s'est avéré que ses capacités de tracking n'étaient pas mauvaises et sa précision plutôt bonne (90% de réussite[14]). Cependant, cette dernière présente tout de même des problèmes : puisqu'elle ne détecte pas qu'un seul œil, elle peut en trouver plus de deux sur l'image (notamment là où il n'y en a pas, ailleurs sur le visage qu'au niveau des yeux). Cela implique alors que si des personnes passent dans l'arrière-plan lors d'une utilisation, la librairie va se servir de tous les yeux détectés afin de trouver ce que l'utilisateur regarde.

[6] https://sourceforge.net/projects/gazetrackinglib/
[7] https://xlabsgaze.com
[8] http://gazerecorder.christiaanboersma.com/
[9] http://gazerecorder.christiaanboersma.com/gazeflow/
[10] http://www.eyetwig.com/
[11] https://opencv.org/ et https://docs.opencv.org/2.4/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html

Cela peut porter à confusion et n'est donc pas une solution très viable. Enfin, cette dernière est assez complexe à prendre en main, et il a donc été préféré de chercher une meilleure alternative.

Finalement, une librairie fournie par le créateur de PyGaze[12] a été trouvée (elle sera aussi nommée PyGaze puisque cette librairie initiale n'est plus employée dans cette partie du travail). Celle-ci permet la détection de l'œil en jouant avec des luminosités et contrastes et donne la possibilité à l'utilisateur d'appliquer les réglages qu'il souhaite afin d'optimiser la détection de l'œil. De plus, cette librairie a aussi pour point positif d'être compatible avec le code écrit dans la première partie du projet, et qu'il ne faut alors pas migrer totalement le code existant, risquant de perdre une fonctionnalité stable. Cependant, le code de la librairie consistait surtout en un challenge-test personnel pour le développeur, Edwin Dalmajier, et présente donc une qualité de détection assez faible : l'opération se fait à chaque fois mais il y a des risques de perte de cette détection si quelque chose de semblable (en coloris du moins) à un œil se place dans le champ de vision de la caméra (80% des essais ont déplacé l'œil pour le situer dans l'arrière-plan ou ailleurs sur le corps).

Nom de la librairie	Technologie	Points positifs	Points négatifs
Gaze Tracking Library	C#	- Détecte les réflexions rétiniennes et cornéennes - Détecte le centre de la pupille - Fonctionnement proche d'un eye tracker	- Plus à jour - Site supprimé (plus d'implication de la part des développeurs) - Communauté inexistante depuis 2013
XLab	Web	- Basé sur du machine learning	- Uniquement pour applications web - Demande l'inscription et la validation par les développeurs pour avoir accès au SDK
GazeFlow (GazeRecorder & GazePointer)	Logiciel	- Détection puissante - Précision très bonne (comparable à un eye tracker SMI) - Permet une liberté de mouvement de la tête - Interface graphique	- Pas de SDK ou d'API disponible
EyeTwig	Logiciel	- Détection puissante - Bonne précision (taper au clavier par le regard) - Interface graphique	- Pas de SDK ou d'API disponible - Uniquement sur Mac - Refusé dans l'AppStore
OpenCV	Python, C, C++	- Bonne précision (90%) - Présente sur Python (uniquement pour voir ce qui a été détecté) - Communauté de support	- Ne compte pas le nombre d'yeux présents sur l'image (plus de deux) - Pas de distinction si plusieurs personnes sont sur l'écran - Complexité élevée de mise en place
PyGaze (webcam)	Python	- Présente sur Python - Possibilité de régler la détection de l'œil - Communauté de support et développeur qui répond	- Méthode de luminosité et contraste peu fiable - Détection de l'œil difficile - Sensible aux changements d'arrière-plan

TABLE 3.1: Tableau comparatif des librairies explorées.

[12] `https://www.pygaze.org/2015/06/webcam-eye-tracker/`

C'est donc la librairie PyGaze qui a été employée pour réaliser cette partie-ci. En effet, elle s'intègre facilement dans le code précédemment écrit. Bien que sa détection de l'œil soit moins bonne que celle d'OpenCV, il est plus facile et rapide d'entrer dans le code de PyGaze et de le comprendre. Les deux librairies ayant une bonne communauté de support, elles présentent toutes deux des avantages qui s'équivalent. C'est par souci de création d'un prototypage rapide et constance d'intégration par rapport à la partie précédente que PyGaze a été préféré. Il n'est cependant pas impossible de travailler avec OpenCV. Les autres librairies, quant à elles, ne peuvent être employées tant les inconvénients qui les qualifient sont importants.

Afin de faire fonctionner le code de détection de l'œil, il est important d'installer une librairie supplémentaire : VideoCapture. Cette dernière n'ayant pas été mise à jour depuis longtemps, certaines méthodes sont dépréciées et ne s'exécutent alors pas. Il est donc primordial de remplacer à la ligne 138 :

```
Image.fromstring('RGB', (width, height), buffer, 'raw', 'BGR', 0, -1)
```

par

```
Image.frombytes('RGB', (width, height), buffer, 'raw', 'BGR', 0, -1)
```

3.3.2 Développement et produit fini

Avant d'aborder le fonctionnement du logiciel développé, il est important d'introduire les différentes hypothèses qui ont été posées sur le use case. En effet, l'idée originelle était la création d'un panneau interactif fonctionnant sous certains principes des interactions proxémiques auquel y serait ajouté un contrôle par le regard. Cependant, de par le choix de la librairie et les contraintes techniques, il n'était pas envisageable d'employer ce logiciel dans un lieu public. En effet, comme mentionné précédemment, PyGaze se base sur la détection via des coloris, la luminosité et le contraste. Dès lors, la librairie est très sensible aux couleurs et changements dans l'arrière-plan. Un exemple plus concret sera donné ci-dessous. De plus, cette technologie demande une forte mise au point par rapport à l'utilisateur et sa détection étant médiocre, le contexte d'utilisation devait alors être très spécifique, comme explicité un peu plus loin dans cette section. Cela ne permet donc pas d'intégrer les interactions proxémiques au sens des distances, mouvements, orientations, etc. qui ont été définies théoriquement. Dès lors, le projet réalisé consiste donc en un petit panneau (limité à l'écran d'un ordinateur) utilisé dans une sphère privée mais permettant tout de même la sélection sur l'écran par le regard.

Dans cette partie, le fonctionnement du script sera expliqué, notamment la partie de détection de l'œil, présentant certaines particularités.

Ce script Python, qui propose un petit use case mais n'est pas utilisable dans un environnement public, fonctionne en deux grandes parties : d'abord la phase de détection de l'œil et, ensuite, l'affichage d'une fenêtre comportant plusieurs images (ici, deux) susceptibles d'être sélectionnées et agrandies via le regard et la SA.

FIGURE 3.2: Interfaces de la détection de l'œil.

Lors du lancement du script, la détection oculaire démarre. Cette dernière se présente dans une nouvelle fenêtre qui consiste en un écran de configuration et d'adaptation de la détection, afin que l'œil de l'utilisateur soit bel et bien ciblé. Pour le bon fonctionnement de cette étape (et du programme en général), il est primordial que l'utilisateur se trouve dans un environnement neutre, face à un mur dans les tons blancs, par exemple. De plus, si celui-ci a des cheveux de couleur foncée, il est préférable qu'il porte un accessoire de tête plus clair, afin que la détection ne se fasse pas au niveau du haut de son crâne.

L'interface de détection de l'œil se présente simplement, en 3 boutons[13]. Le premier sert à régler les ombres, le second à former une petite boite autour de l'œil et le troisième à détecter la pupille. Afin d'effectuer ces réglages, il est important de d'abord cliquer sur le bouton « togl », qui passera alors l'image dans une nuance de mauves destiné à effectuer ces réglages.

Dans la première option, le but est de jouer avec les contrastes (typiquement en le baissant, à l'aide de la flèche du bas) afin d'éliminer toute zone sombre sur le visage mis à part la partie correspondant aux yeux, similaire à ce que l'on peut voir dans la Figure 3.3. Il faut faire attention à ne pas trop baisser cette mesure afin que la zone des yeux soit toujours visible. Une fois que cette zone se démarque, l'utilisateur peut passer à l'étape suivante. Celle-ci consiste en la démarcation de l'œil dans une petite boite. Pour cela, il faut alors cliquer sur la tache sombre correspondant et ajuster la taille de la boite qui l'entoure à l'aide des flèches pour qu'elle encadre bien l'œil, comme dans la Figure 3.4. Enfin, la dernière étape consiste à identifier la pupille, toujours à l'aide d'une petite boite. A ce moment, il n'est pas toujours nécessaire de jouer avec les flèches afin d'adapter les réglages, cependant, cela peut se faire. Après cela, il est possible

[13]Voir Figure 3.2

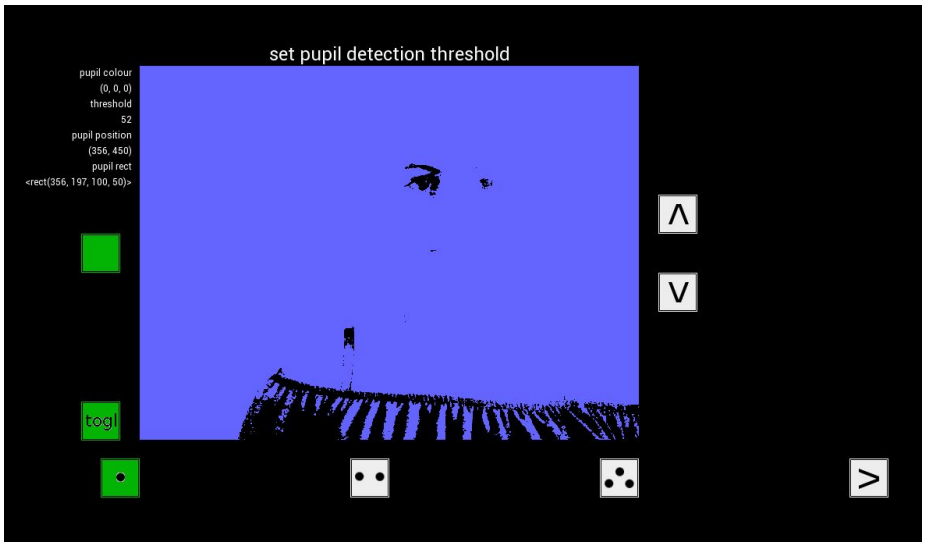

FIGURE 3.3: Diminution du contraste.

de revenir en mode couleur en cliquant à nouveau sur le bouton « togl » et de voir sa pupille détectée, comme dans la Figure 3.5.

Ce qui est intéressant avec cette méthode de détection de l'œil, c'est que, si jamais la tête bouge radicalement sur l'écran, la pupille est toujours reconnue, pour peu qu'elle soit encore visible par la caméra et que le fond soit uniforme. Cela permet une liberté de mouvement qui est la bienvenue dans le contexte d'interactions proxémiques face à un large écran. Cependant, cela présente aussi un léger désavantage : les coordonnées de la pupille sur l'écran sont assez imprécises lorsque seulement celle-ci bouge . En effet, il n'y a alors que 20 pixels d'écart entre un œil qui regarde totalement à gauche de l'écran et puis à l'extrême droite de celui-ci.

Sur base de ces mesures, le choix a été posé de n'afficher que deux images dans le use case, afin de scinder l'écran en deux parties : la droite et la gauche. Ainsi, une fois la détection terminée, en appuyant sur la petite flèche en bas à droite, une nouvelle fenêtre s'affiche, proposant deux images de choix. Si l'œil est considéré être dans la partie droite de l'écran, le choix 1 sera agrandi pour une durée de 10 secondes. Dans le cas contraire, c'est le choix 2 qui sera affiché. Pour cela, il est possible de simuler le clic sur une fenêtre grâce à la librairie OpenCV.

Afin d'exécuter ce « faux clic », la SA a été employée. Bien que plusieurs mesures aient été exploitées dans la première partie de ce travail, toutes ne sont pas nécessaires dans le cadre de cette implémentation. En effet, il n'est pas utile de mesurer l'entropie ou les distances parcourues par l'œil pour savoir ce qui a intéressé l'utilisateur. Dès lors, deux AOI ont été définies ; une pour chaque image, et la mesure employée est le dwell time. Pour une période de 3 secondes, les temps de séjour dans l'une et l'autre sont comparés et l'image ayant la valeur la plus élevée est considérée comme celle ayant le plus attiré l'utilisateur. Elle

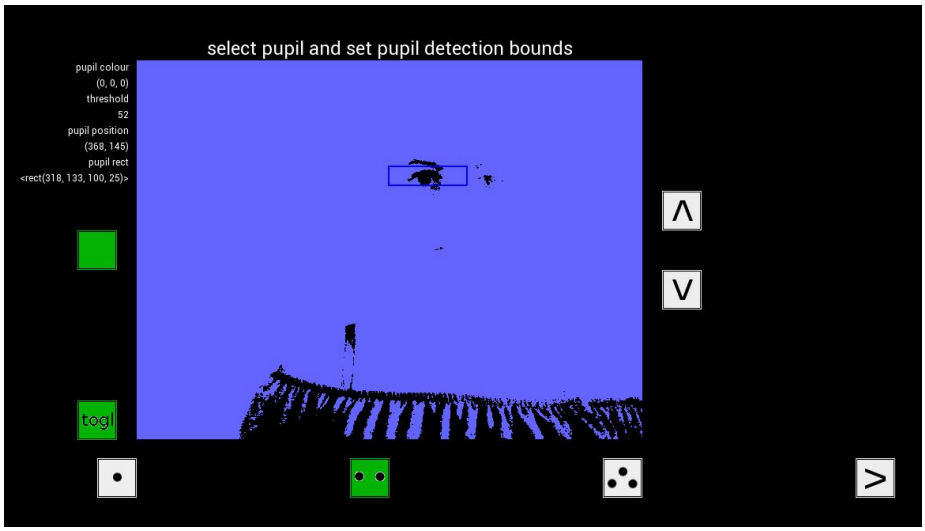

FIGURE 3.4: Identification d'un œil.

est donc agrandie pour une période de temps limitée.

Concrètement, cet usage fait office de prototype afin de tester les possibilités technologiques mais n'est pas approprié dans le cadre d'un cas réel d'affichage sur un écran dans un lieu public. En effet, de par la mauvaise qualité et précision du système, les bruits de l'image peuvent totalement fausser les réglages, ce qui donne lieu à des détections d'yeux à des endroits parfois improbables. De plus, l'utilisateur potentiel ne va pas se balader sans cesse avec un bonnet blanc au cas où il a les cheveux de couleur foncée. Enfin, la procédure de détection de l'œil est même plus longue et fastidieuse que la calibration traditionnelle (par eye tracker ou par webcam). Par ailleurs, au vu des hypothèses qui ont été mises en place pour palier à la mauvaise qualité de la technologie, un tel système était voué à être limité dans son cas d'utilisation.

Pour pallier à la détection médiocre de la librairie employée, il faudrait en utiliser une autre. Cependant, comme il l'a été expliqué en profondeur dans la section précédente, peu d'alternatives sont fournies à PyGaze. Si OpenCV n'est pas sujette aux mêmes critiques, sa détection d'un nombre d'yeux illimité ne la rend pas forcément meilleure candidate. Dès lors, pour que ce prototypage puisse devenir un cas concret d'utilisation, il est nécessaire d'employer des algorithmes de machine learning poussés, ce qui n'était pas envisageable dans le cadre de ce travail.

Dans le chapitre suivant, les techniques qui auraient pu être mises en place sous condition d'une technologie acceptable et de plus de moyens seront présentées afin de comprendre ce qui pourrait potentiellement améliorer le système et permettre la mise en place un réel système ubiquitaire afin de traquer l'œil sur un écran dans un lieu public.

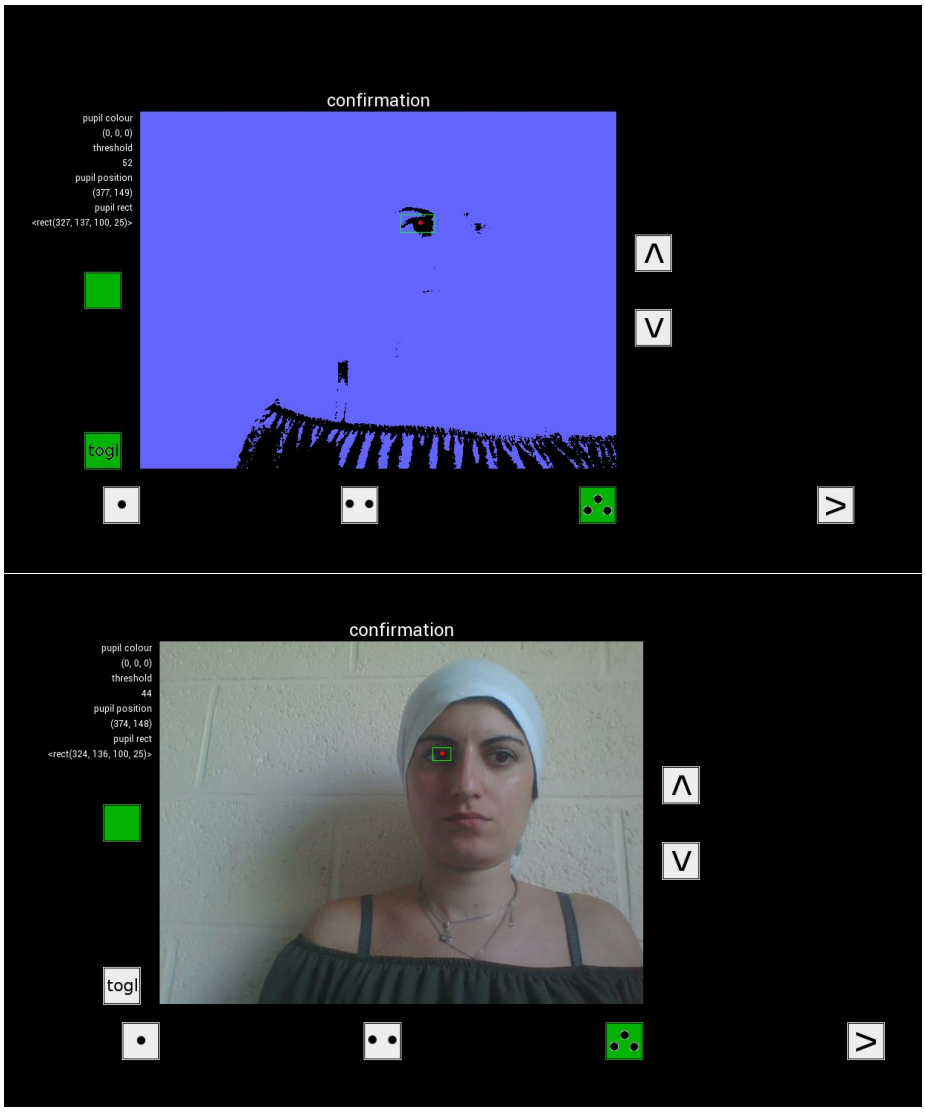

FIGURE 3.5: Identification de la pupille.

Chapitre 4

Perspectives et travaux futurs

Dans les chapitres précédents, les grands challenge de l'intégration de l'eye tracking aux interactions proxémiques ont été présentés et une tentative d'implémentation d'un cas d'utilisation a été réalisée. Lors de la réalisation de ce programme, de nombreuses difficultés ont elles aussi été rencontrées.

Dans ce chapitre, des solutions vont être présentées et explorées afin de pouvoir fournir des éléments de réponses dans le cas où de futures recherches dans le même domaine seraient entamées. Dès lors, ces solutions auront pour but de répondre aux grands challenges présentés dans l'état de l'art, et seront à la fois des résultats théoriques tirés d'articles s'étant penchés sur la question et de constations ressortant des difficultés auxquelles il a fallu faire face dans la cadre de ce travail. Enfin, quelques problématiques qui ont été mises de côté seront soulevées afin qu'elles soient toujours prises en compte lors d'extensions de ce travail.

4.1 Solutions envisageables

Comme il a été évoqué plus tôt, plusieurs grands challenges ont été identifiés au cours de l'état de l'art, plus précisément cinq (voir Section 2.3.3). Ces problématiques concernaient la calibration, la précision, la verticalité, le guidage de l'utilisateur et ce qui avait été appelé le « problème de Midas ». En outre, des difficultés, essentiellement technologiques, avaient été rencontrées au cours de l'implémentation du cas d'utilisation.

Dans cette section, il sera présenté plus en détail quelles solutions ont été apportées à chacun de ces challenges et comment elles ont été mises en place, que ce soit à l'aide d'ouvrages ou grâce au travail d'implémentation effectué.

Le premier point qu'il semble normal d'aborder est le choix de la technologie. En effet, comme il l'a été évoqué au point précédent, la technologie a posé plus qu'un problème dans la réalisation de ce travail, qui en dépend complètement. Afin de réaliser le système imaginé, il est possible d'employer soit les eye trackers

distants, soit les webcams, comme cela a pu être exploré au point précédent. Les seules technologies qu'il faille à tout prix exclure sont les eye tracker portables. En effet, bien que ceux-ci soient plus précis que le tracking par webcam, il est difficilement envisageable de fournir une paire de lunettes technologiques à tous les utilisateurs potentiels au cas où ils auraient à utiliser un panneau interactif.

Comparativement, le choix de la webcam ou de l'eye tracker est assez équivalent puisque les technologies présentent toutes les deux leurs avantages et leurs inconvénients. Plusieurs projets utilisent le premier[33] tandis que d'autres préfèrent le second[12]. Si l'eye tracker est en effet plus précis, il est également plus sensible face aux changements de positions de l'utilisateur qui ne va pas rester immobile pendant une durée plus ou moins longue. De plus, la webcam étant déjà beaucoup utilisée dans le cadre des interactions proxémiques, y ajouter la fonctionnalité de trackage de l'œil peut présenter un avantage de coût considérable.

Lors de la partie technique, un comparatif des différentes librairies permettant de traquer le regard via une webcam a été dressé (voir Tableau 3.1). Si beaucoup de possibilités ont été explorées, rares sont celles qui se sont démarquées positivement. En effet, mis à part PyGaze et OpenCV dont les forces et les faiblesses ont déjà été passées en revue, seuls les logiciels GazePointer et GazeTracker sont sortis du lot pour leur tracking impeccable mais leur manque d'API ou SDK. Dès lors, la seule solution viable pour un potentiel tracking par webcam serait de créer une librairie « maison » sur base d'algorithmes de machine learning de reconnaissance d'images. Ceci nécessiterait alors, en plus, la recherche d'un nombre de personnes suffisant pour roder le système.

De plus, lorsque le choix technologique se porte sur le tracking par webcam, il faut s'assurer de posséder un matériel d'une qualité suffisante. Ainsi, Zhang préconise une webcam de résolution 1280 par 720 pixels et une fréquence de 30Hz[33] et Edwin Dalmaijer, le créateur de PyGaze, conseille l'utilisation d'une webcam de 1280 par 1024 pixels ainsi que des leds pour illuminer son visage[1].

Par ailleurs, le travail s'est aussi, au début, basé sur des eye trackers, notamment le modèle Tobii 4C. Bien que le travail n'ait pas abouti pour des soucis d'ordre technique, cela reste intéressant d'utiliser un eye tracker présentant une meilleure précision et fiabilité. Bien que rares soient les technologies pouvant concurrencer Tobii, marque la plus répandue, certaines alternatives existent encore sur le marché. Si le souhait du potentiel futur chercheur est de reprendre le code Python déjà créé, il existe les trackers de la marque Eyelink, compatibles avec le Python 2.7 et la librairie PyGaze, qui sont fournis avec de très bons guides du programmeur[2].

Que le choix se porte sur l'une ou l'autre technologie, la seconde difficulté à affronter se trouve être la calibration. En effet, comme mentionné plus tôt dans ce travail, la calibration pose un coût à la fois en temps (durée plus ou moins longue qui empiète sur le temps de découverte que l'utilisateur est prêt à ac-

[1] `https://www.pygaze.org/2015/06/webcam-eye-tracker/`

[2] `http://human.cbtc.utoronto.ca/tutorials/eyelink/EyeLink_Programmer_Guide.pdf` ainsi que `http://download.sr-support.com/dispdoc/group__setup__eyelink.html`

corder au panneau interactif) et en performances (il faudrait alors être capable de garder en mémoire les données de calibration de chaque personne et être capable de reconnaitre un ancien utilisateur dans le but de ressortir cette donnée).

Plusieurs options sont possibles afin d'éviter la calibration. Certains auteurs se basent sur des données mathématiques pures. En effet, il existe un point commun à tous les yeux : la réflexion cornéenne, qui se traduit par un point blanc sur l'œil. Dès lors, à l'aide d'une série d'opérations mathématiques, il est possible de passer outre les opérations de calibration[25]. Une autre technique consiste à se baser sur des objets dynamiques et corréler le regard à ces objets, ce qui permet d'outrepasser la phase de calibration[28]. Il est d'ailleurs même possible d'appliquer cette technique à de la lecture de textes. Cependant, cela donne des résultats un peu moins précis[13]. Ces trois techniques précédentes ont été appliquées aux eye trackers. Dans le cas de l'emploi d'une webcam, la technique référencée dans les ouvrages est celle qui a été employée dans ce travail : la division de l'écran en deux ou trois catégories (gauche, droite, centre) afin de limiter les besoins en précision et donc de se débarrasser de l'utilité de la calibration[33].

Les imprécisions et la perte de verticalité seront toujours des problématiques qui ne pourront jamais être totalement résolues. Bien que le port d'eye trackers portables puisse apporter une solution, il a déjà été démontré plus tôt que ce n'était pas envisageable dans une situation réelle. Pourtant, l'utilisation d'eye trackers peut être une bonne idée pour réduire ces imprécisions, contrairement à une webcam. Il faut alors trouver un moyen de palier au manque de mouvement de cet eye tracker et à son champ assez limité. Pour cela, Khamis a mis en place un eye tracker mobile, monté sur des rails qui, couplés à une kinect, permettent de suivre l'utilisateur[12].

Le problème de Midas est un nouvel élément qui avait été relevé dans les ouvrages. Pour rappel, il concernait les mouvements parasites, c'est-à-dire mouvements involontaires qui viendraient cependant à être captés par la technologie et donc fausser les résultats. Cette problématique est particulièrement valable dans le cadre de l'utilisation de l'œil en tant que modalité d'interactions, puisque les mouvements des yeux sont les plus incontrôlés et traduisent justement de la réaction subconsciente de l'utilisateur. Dans les ouvrages, aucune solution ne se distingue réellement pour palier à ce problème. Cependant, grâce aux connaissances théoriques acquises au cours de ce travail, il pourrait être envisageable d'introduire la SA comme solution. En effet, cette métrique a pour but de pointer là où l'utilisateur est conscient de la situation. Cet ajout permet alors aux mouvements parasites de se fondre dans la masse et de n'avoir pas plus d'incidence et d'influence qu'ils n'en ont réellement. Ainsi, à l'inverse de Walter qui utilise ces interactions proxémiques pour comprendre l'attention visuelle[29], c'est cette attention qui va influencer le comportement proxémique du panneau dans un lieu public.

La dernière problématique qui a été identifiée est celle du guidage de l'utilisateur. En effet, ce dernier doit être capable d'approcher le panneau et d'instinctivement comprendre qu'il est interactif. De plus, il ne doit pas nécessiter les explications d'un expert afin d'être utilisable. Pour solutionner ce cas, il est nécessaire de mettre l'accent sur le design d'interaction. Ainsi, par exemple,

allumer l'écran lorsque celui-ci s'approche de la zone du panneau, un peu à l'idée du lecteur de musique dans un salon[2], peut être une des solutions mises en place pour guider l'utilisateur. De plus, un écran d'accueil attractif et expliquant brièvement les modalités d'utilisation en quelques mots est une possibilité envisageable. La meilleure façon de régler cela est de conduire une expérience utilisateur sur le système afin de voir ce qu'il faut améliorer ou pas.

Malgré la mention de toutes ces problématiques et leurs solutions, un point n'a toujours pas été abordé : la question des utilisateurs multiples. En effet, souvent les ouvrages font une hypothèse que l'utilisateur est seul face au dispositif puisqu'un eye tracker ne gère typiquement pas plusieurs paires d'yeux (et pose d'ailleurs beaucoup de problèmes de calibration lorsqu'un second visage se trouve dans son champ d'enregistrement). Pourtant, Dostal a mis en place une solution permettant de diviser l'écran en plusieurs sous-écrans afin de laisser jusqu'à 4 personnes interagir avec le panneau qu'il met en place. Pour cela, il utilise des techniques de tracking particulières à l'aide d'une caméra et d'une Kinect[7].

4.2 Challenges en suspens

Tout au long de ce travail, certains points ont été volontairement mis de côté afin de mettre l'accent sur ce qui était plus important. Cependant, ces hypothèses simplificatrices doivent être listées afin de pouvoir guider une potentielle future étude de la façon la plus correcte possible. Cependant, toutes ces hypothèses ne possèdent pas toujours une solution, de par le manque de technologie mise à disposition du grand public.

Tout d'abord, dans le cadre de l'implémentation, suite aux soucis technologiques évoqués précédemment, le côté proxémique a totalement été mis de côté afin de se focaliser sur l'interaction du regard avec l'écran. Ainsi, bien qu'il y ait une interaction, il manque un toolkit permettant de reconnaitre la proximité d'un utilisateur, de définir différentes zones d'interaction comme expliqué précédemment, etc.[15]

De plus, il ne faut pas perdre de vue la définition donnée par Weiser signifiant que l'informatique doit totalement se fondre dans le contexte d'utilisation, que l'utilisateur doit effectuer les tâches comme il le ferait naturellement. Ainsi, dans ce cas, cela signifie la présence d'un panneau interactif dans un endroit où il semble utile de placer un tel dispositif et pas dans le but de faire de l'interaction inutile. Par exemple, dans un aéroport, un panneau affichant les horaires de vol et où l'utilisateur serait capable de faire défiler les pages jusqu'à trouver le vol qu'il cherche. Ou encore, la présence d'un panneau interactif dans un centre commercial, permettant d'afficher un plan et de zoomer sur certains magasins à l'aide du regard.

Enfin, il faut faire attention à sécuriser le système[16]. En effet, les données captées par celui-ci pourraient intéresser certaines personnes mal intentionnées qui décideraient alors de s'y introduire de façon illégale. Il est donc important de prendre des précautions avant de mettre un système en état de marche réel.

Chapitre 5

Conclusion

Ce travail avait pour but de vérifier si l'eye tracking pouvait devenir une modalité dans un système ubiquitaire proxémique. En particulier, cette problématique avait été appliquée à un cas spécifique : un panneau interactif dans un lieu public peut-il être contrôlé principalement par le regard ? De plus, la question du rôle que peut jouer la SA dans un tel système a été étudiée de plus près.

Dans le but de pouvoir répondre à ces questions, il a d'abord fallu se plonger dans la théorie afin de comprendre en profondeur les différentes notions évoquées. Cela s'est fait au travers du stage réalisé à l'Université du Texas d'Austin, lors de la conception d'une première application liant eye tracking et SA.

Cette réalisation a vraiment été axée sur l'intégration des différentes notions de la SA afin de créer un outil de recherche unique pour l'université. En effet, celui-ci devait être couplé à un logiciel appelé iMotions dans le but de faire des études d'utilisabilité poussées. Ainsi, les données de dwell time, d'entropie et de distances parcourues par le regard pouvaient être affichées en temps réel, permettant alors aux chercheurs de modifier le cours de l'étude grâce aux résultats présentés.

Ensuite, les notions d'informatique ubiquitaire et d'interactions proxémiques ont été intégrées petit à petit afin de tenter de réaliser un système interactif tel que décrit ci-avant.

Suite aux difficultés techniques indépendantes de toute volonté rencontrées au cours de ce travail, certaines hypothèses ont été mises en place afin de simplifier le cas d'utilisation tout en permettant quand même de répondre aux questions posées en début de travail. L'application résultant alors des recherches menées était un petit programme permettant de sélectionner, à l'aide du regard, une image et de l'agrandir sur un écran d'ordinateur.

Si le système conçu ne permet pas de répondre catégoriquement à la question du regard en tant que modalité dans un système ubiquitaire proxémique, une réponse partielle peut être formulée. En effet, s'il n'est pas possible de sa-

voir si le regard peut constituer une telle modalité, il est réaliste d'affirmer que la réalisation et mise en place d'un tel système constituent un défi de taille et demande des investissements et des connaissances poussées. Cependant, certaines parties du script réalisé au cours de ce travail permettent la résolution de certains challenges, comme le contournement de la calibration. De plus, cet écrit permet d'ouvrir des portes concernant une future étude en présentant une analyse assez complète des technologies mises à la disposition du grand public. Enfin, dans la dernière partie de ce travail, les grands challenges que présentent une telle intégration sont démontés à l'aide de solutions afin d'ouvrir la porte à de futures recherches. Il est d'ailleurs à noter que chaque auteur répond à une des problématiques citées dans cette partie mais aucun ne semble avoir pris l'initiative de combiner les diverses solutions afin de tenter d'éliminer toutes les problématiques soulevées en une seule fois et ce, malgré l'existence de certains articles parus depuis plusieurs années. Il est donc là une piste à explorer pour un futur travail.

Par ailleurs, ce travail permet tout de même de répondre à la seconde question. En effet, le rôle de la SA dans les systèmes ubiquitaires proxémiques utilisant le regard comme une modalité pourrait être de corriger le « problème de Midas ». En effet, s'il est certain que l'œil de l'utilisateur est sujet à des mouvements incontrôlables lui permettant d'avoir une vision complète de son environnement, ici l'écran, l'utilisation des données traitées par des mesures de SA permet de garantir que ces petits mouvements se noient dans la masse des fixations sur ce qui intéresse réellement l'utilisateur.

En conclusion, si l'intégration de l'eye tracking aux systèmes ubiquitaires proxémiques n'est pas ce qu'il y a de plus simple, ce sujet présente des applications plus qu'intéressantes qui mériteraient qu'un financement et des recherches plus poussées lui soient accordés. De plus, bien que la théorie concernant l'intégration de la SA à un tel système pointe vers une élimination du problème de Midas, il pourrait être intéressant d'appliquer cette théorie à un système fonctionnel dans des études avec utilisateur.

Enfin, force est de constater que le sujet de l'eye tracking, très en vogue il y a une dizaine d'années, ne semble plus être utilisé pour autre chose que des études UX. Dès lors, une des raisons ayant mené ce travail vers un échec quant à sa question principale de l'intégration de l'eye tracking aux systèmes ubiquitaires proxémiques est le manque de mises à jour et de matériels disponibles. En effet, beaucoup de marques d'eye tracker se sont faites racheter par de plus grandes boites qui les ont ensuite laissées dépérir.

Bibliographie

[1] Sriram Karthik Badam, Fereshteh Amini, Niklas Elmqvist, and Pourang Irani. Supporting visual exploration for multiple users in large display environments. In *2016 IEEE Conference on Visual Analytics Science and Technology, VAST 2016 - Proceedings*, pages 1–10, 2016.

[2] Till Ballendat, Nicolai Marquardt, and Saul Greenberg. Proxemic Interaction : Designing for a Proximity and Orientation-Aware Environment. *ACM International Conference on Interactive Tabletops and Surfaces - ITS '10*, pages 121–130, 2010.

[3] Aga Bojko. *Eye tracking the user experience*. Rosenfeld Media, 2013.

[4] Andreas Bulling and Hans Gellersen. Toward Mobile Eye-Based Human-Computer Interaction. *IEEE Pervasive Computing*, 9(4) :8–12, 2010.

[5] Edwin S. Dalmaijer, Sebastiaan Mathôt, and Stefan Van der Stigchel. PyGaze : an open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments. *Behavior research methods*, 46(4) :913–921, 2014.

[6] Soussan Djamasbi and Adrienne Hall-Phillips. *Visual Search*. Elsevier Inc., 2014.

[7] Jakub Dostal, Uta Hinrichs, Per Ola Kristensson, and Aaron Quigley. SpiderEyes : designing attention- and proximity-aware collaborative interfaces for wall-sized displays. In *Proceedings of the 19th international conference on Intelligent User Interfaces - IUI '14*, pages 143–152, Haifa, 2014.

[8] Mica R. Endsley. Theoretical Underpinnings of Situation Awareness : A Critical Review. *Situation Awareness Analysis and Measurement*, pages 3–32, 2000.

[9] Mica R. Endsley and Debra G. Jones. *Designing for situation awareness : an approach to user-centered design*. CRC Press, 2012.

[10] Ian Everdell. Web Content. *Eye Tracking in User Experience Design*, pages 163–186, 2014.

[11] Aaron W. Johnson, Kevin R. Duda, Thomas B. Sheridan, and Charles M. Oman. A closed-loop model of operator visual attention, situation awareness, and performance across automation mode transitions. *Human Factors*, 59(2) :229–241, 2017.

[12] Mohamed Khamis, Axel Hoesl, Alexander Klimczak, Martin Reiss, Florian Alt, and Andreas Bulling. EyeScout : Active Eye Tracking for Position and Movement Independent Gaze Interaction with Large Public Displays. In *UIST '17 Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 155–166, Quebec City, 2017.

[13] Mohamed Khamis, Ozan Saltuk, Alina Hang, Katharina Stolz, Andreas Bulling, and Florian Alt. TextPursuits : Using Text for Pursuits-Based Interaction and Calibration on Public Displays. *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 274–285, 2016.

[14] Erik Learned-Miller, Gary B. Huang, Aruni RoyChowdhury, Haoxiang Li, and Gang Hua. Labeled faces in the wild : A survey. In *Advances in Face Detection and Facial Image Analysis*, pages 189–248. Springer, 2016.

[15] Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. The Proximity Toolkit : Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. *Proceedings of the 24th Annual Symposium on User Interface Software and Technology (UIST'11)*, pages 315–325, 2011.

[16] Nicolai Marquardt and Saul Greenberg. Informing the design of proxemic interactions. *IEEE Pervasive Computing*, 11(2) :14–23, 2012.

[17] Nicolai Marquardt and Saul Greenberg. Proxemic Interactions : From Theory to Practice. *Synthesis Lectures on Human-Centered Informatics*, 8(1) :1–199, 2015.

[18] Koen Van De Merwe, Henk Van Dijk, and Rolf Zon. Eye Movements as an Indicator of Situation Awareness in a Flight Simulator Experiment. *The International Journal of Aviation Psychology*, 22(1) :78–95, 2016.

[19] Kristin Moore and Leo Gugerty. Development of a novel measure of situation awareness : The case for eye movement analysis. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 54(19) :1650–1654, 2010.

[20] Erica Olmsted-Hawala, Temika Holland, and Victor Quach. Usability Testing. *Eye Tracking in User Experience Design*, pages 49–80, 2014.

[21] Anneli Olsen. Determining the Tobii I-VT Fixation Filter ' s Default Values. *Tobii Technology*, 2012.

[22] Anneli Olsen. The Tobii I-VT Fixation Filter : Algorithm description. *Tobii Technology*, page 21, 2012.

[23] Matos Ricardo Olsen Anneli. Identifying parameter values for an I-VT fixation filter suitable for handling data sampled with various sampling frequencies. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 317–320, Santa Barbara, 2012.

[24] Andrew Schall and Jennifer Romano Bergstrom. *Introduction to Eye Tracking*. Elsevier Inc., 2014.

[25] Sheng-wen Shih, Yu-te Wu, and Jin Liu. A calibration-free gaze tracking technique. In *Proceedings of the International Conference on Pattern Recognition*, volume 4, pages 201–204, 2000.

[26] Ben Steichen, Giuseppe Carenini, and Cristina Conati. User-adaptive information visualization : using eye gaze data to infer visualization tasks and user cognitive abilities. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 317–328, Santa Monica, 2013.

[27] Dereck Toker, Ben Steichen, Matthew Gingerich, Cristina Conati, and Giuseppe Carenini. Towards Facilitating User Skill Acquisition : Identifying Untrained Visualization Users through Eye Tracking. In *Proceedings of the 19th international conference on Intelligent User Interfaces - IUI '14*, pages 105–114, Haifa, 2014.

[28] Melodie Mélodie Vidal, Andreas Bulling, and Hans Gellersen. Pursuits : Spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 439–448, Zurich, 2013.

[29] Robert Walter, Andreas Bulling, David Lindlbauer, Martin Schuessler, and Jörg Müller. Analyzing visual attention during whole body interaction with public displays. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '15*, pages 1263–1267, Osaka, 2015.

[30] Miaosen Wang, Sebastian Boring, and Saul Greenberg. Proxemic Peddler : A Public Advertising Display that Captures and Preserves the Attention of a Passerby. *Proceedings of the International Symposium on Pervasive Displays (PerDis '12)*, pages 3–9, 2012.

[31] Christopher D. Wickens, Juliana Goh, John Helleberg, William J. Horrey, and Donald A. Talleur. Attentional Models of Multitask Pilot Performance Using Advanced Display Technology. *Human Factors : The Journal of the Human Factors and Ergonomics Society*, 45(3) :360–380, 2003.

[32] Bertram Wortelen, Martin Baumann, and Andreas Lüdtke. Dynamic simulation and prediction of drivers' attention distribution. *Transportation Research Part F : Traffic Psychology and Behaviour*, 21 :278–294, 2013.

[33] Yanxia Zhang, Ming Ki Chong, Jörg Müller, Andreas Bulling, and Hans Gellersen. Eye tracking for public displays in the wild. *Personal and Ubiquitous Computing*, 19(5-6) :967–981, 2015.

Annexe A

Schémas

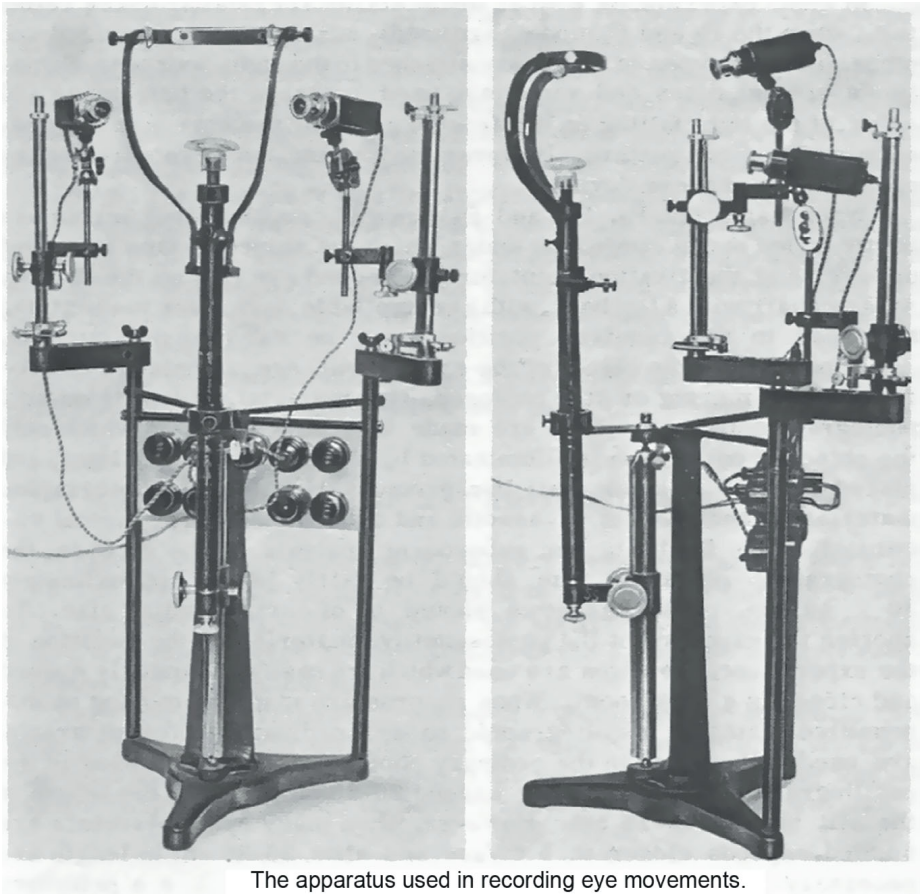

FIGURE A.1: Le premier eye tracker[24].

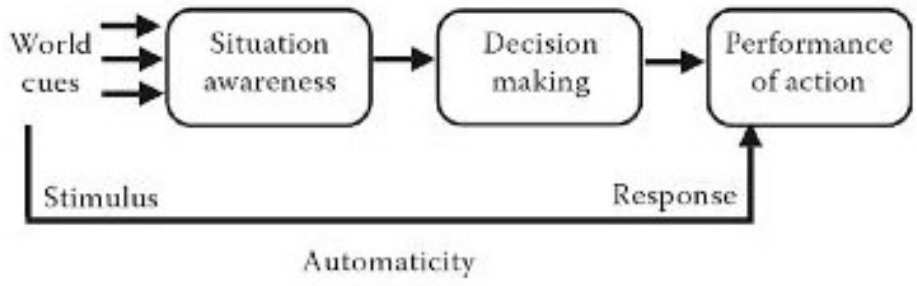

FIGURE A.2: Schema du processus de prise de décision[9].

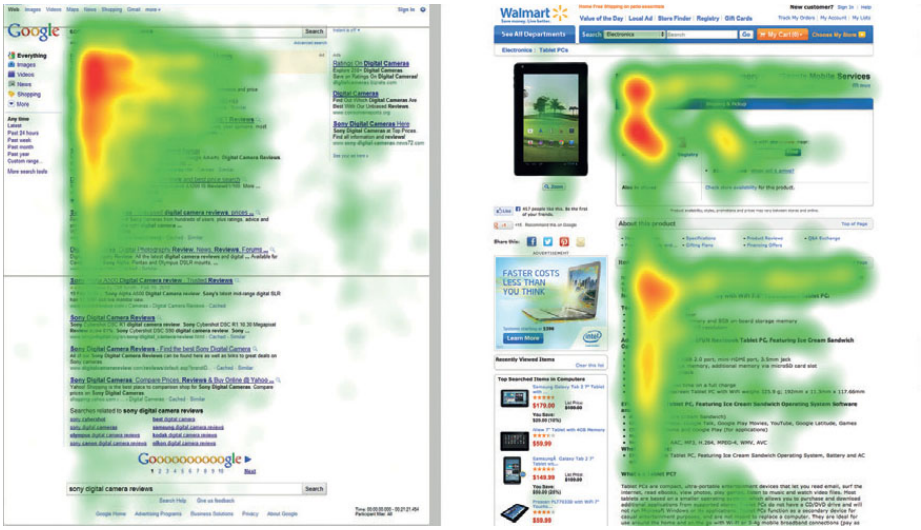

FIGURE A.3: Scan de la page selon le F Pattern (ou triangle doré)[10].

FIGURE A.4: Analyse proposée par le logiciel iMotions à la fin d'une étude du regard.[1]

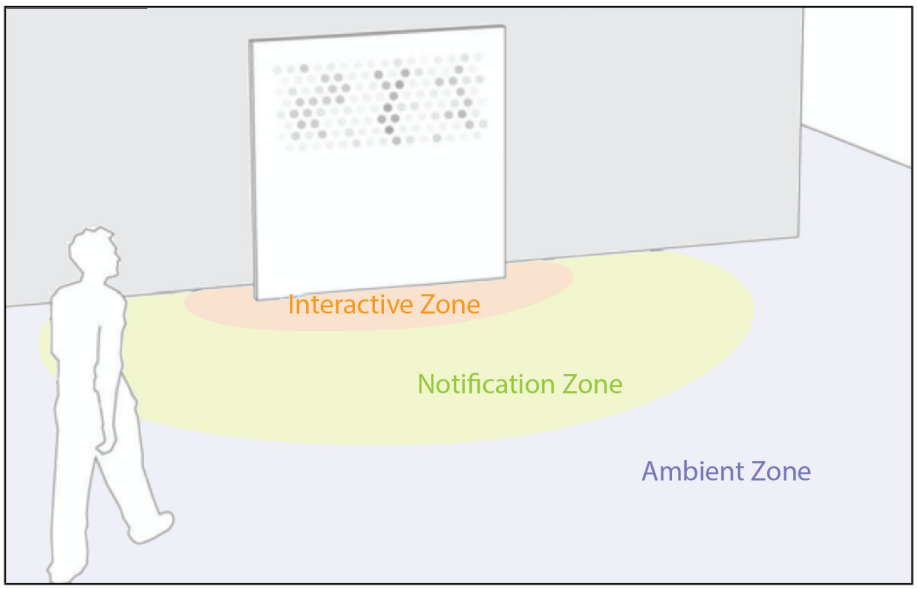

FIGURE A.5: Exemple d'un panneau interactif[17].

[1] https://imotions.com/

Annexe B

Tableaux

	Remote Eye Trackers	Wearable Eye Trackers
Setup	Placed in a fixed location in front of a participant (e.g., on a desk or a car dashboard).	Worn on a participant's head (e.g., as a pair of glasses, attached to a hat, or on a headband).
Application	Used in studies during which participants can sit or stand in one place and the stimulus is presented on a stationary surface (e.g., research using on-screen stimuli such as websites, images, and video).	Used in studies that require participants to move around and interact with physical objects or people (e.g., wayfinding, shopping, and out-of-the-box research).
Obtrusiveness	Less obtrusive than wearable systems—participants can easily forget about the eye tracker during the study.	More obtrusive because the system has to be worn on the head and is usually partially visible to the participant (even though people learn to ignore it after a while). Also, headgear may be undesirable due to hygiene or hairstyle concerns.
Freedom of Movement	Participant must be positioned in front of the eye tracker and only limited head movement is acceptable. There also has to be a clear path between the participant's eyes and the eye tracker (e.g., a participant cannot hold anything in front of his face).	Participant can move around freely and manipulate objects. However, most wearable eye trackers work best for objects that are at the same distance as the calibration, and are less accurate for objects that are closer or farther. This is called a *parallax error* and only some systems can correct for it.
Ease of Analysis	Because the recorded scene doesn't change with head movement, the same gaze location in the recorded frames indicates the same object in the world. Data analysis is typically easier and faster because more automated data mapping and aggregation are possible. However, that's only the case for static stimuli. If the content is dynamic (e.g., a video or a website with a lot of overlays), the lack of a stable coordinate system can make the analysis more manual and time-consuming.	The recorded scene keeps changing due to head movement. As a result, the same gaze location in the recorded frames can indicate something different in the world:  Because of this dissociation, data analysis is typically more manual and time-consuming. This problem can be at least partially solved by placing infrared markers in the environment, using edge/object detection, or adding a head tracker.

FIGURE B.1: Tableau comparatif des eye trackers distants versus portables[3].

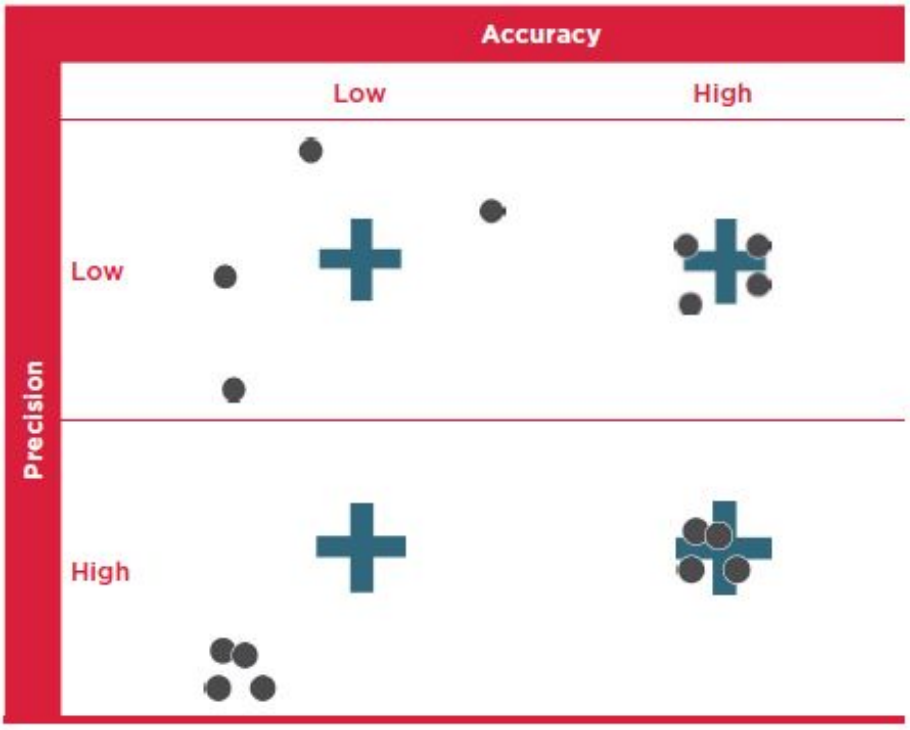

FIGURE B.2: Tableau liant précision (precision) et exactitude (accuracy)[3].

Timestamp	Number	GazePointXLeft	GazePointYLeft	CamXLeft	CamYLeft	DistanceLeft	PupilLeft	ValidityLeft	GazePointXRight	GazePointYRight	CamXRight	CamYRight	DistanceRight	PupilRight
95														
117	0	400.617	261.2175	0.7513002	0.6374704	648.1887	3.773164	0	400.9024	291.1999	0.5961173	0.6521371	636.4001	3.661853
133	1	398.393	264.365	0.7516774	0.6375181	648.2229	3.838122	0	418.6802	288.9395	0.5964873	0.6518686	636.5583	3.64993
150	2	396.2955	273.9963	0.7518954	0.6374961	647.8473	3.77234	0	393.907	301.2133	0.5968071	0.6517037	636.8112	3.675684
167	3	402.125	265.7434	0.7523934	0.6374515	648.0519	3.798369	0	399.3078	284.1953	0.5971763	0.6513929	636.6243	3.675363
183	4	414.2168	251.4495	0.7526376	0.6372757	648.2616	3.804291	0	430.813	272.5971	0.5972841	0.6511323	636.9761	3.629609
200	5	423.9691	255.5654	0.7527826	0.6372469	648.3549	3.810735	0	440.9816	264.9081	0.597554	0.6508658	636.6008	3.630603
216	6	423.9865	245.771	0.7531195	0.6373391	648.2388	3.817439	0	425.7016	274.3857	0.5979909	0.6507218	636.7742	3.65217
233	7	428.22	254.2387	0.7534257	0.6373549	648.2095	3.75736	0	426.7341	262.5856	0.5983127	0.6505361	637.0329	3.608105
250	8	427.502	235.5433	0.7537678	0.6372584	648.4797	3.747368	0	430.5081	257.4155	0.5986091	0.6503124	636.6136	3.607276
266	9	435.1046	242.4257	0.7539775	0.6371354	648.0962	3.729671	0	442.2343	254.1302	0.5988812	0.6500827	637.0001	3.587219
283	10	423.2355	261.5899	0.7541869	0.6370122	648.5149	3.7557	0	423.743	269.3259	0.5989991	0.6499335	636.6981	3.627686
300	11	433.2017	254.8062	0.7542553	0.6369436	648.4265	3.748498	0	421.653	262.8049	0.5990345	0.6497296	636.5421	3.588275
316	12	418.2851	253.153	0.7542648	0.6367035	648.2444	3.760934	0	423.6463	278.318	0.5990194	0.6495155	636.5278	3.576233
333	13	434.9374	248.5142	0.7542472	0.6363882	648.1369	3.736739	0	426.3913	273.6594	0.5989965	0.6492424	636.5372	3.573587
350	14	422.3765	230.4351	0.7540954	0.6362716	647.9678	3.742778	0	429.8147	309.0193	0.5990117	0.6495135	636.341	3.581557
366	15	418.8178	377.8113	0.753844	0.6371815	647.6741	3.709197	0	421.1444	391.3314	0.5990075	0.6500124	636.3063	3.590875
383	16	423.7231	365.6831	0.7537399	0.6368101	647.6413	3.717006	0	412.6543	375.5868	0.5986619	0.6498465	636.2288	3.577973
400	17	417.3529	375.5499	0.7536264	0.6365598	647.4288	3.708932	0	415.1166	374.7715	0.5984329	0.6496119	636.0198	3.570415
416	18	409.0231	369.9335	0.7534955	0.6363221	647.1501	3.723507	0	423.0877	385.2895	0.5982617	0.6494093	635.5084	3.535283
433	19	411.8613	366.2758	0.7534091	0.636264	647.0584	3.723622	0	417.2631	381.453	0.5981212	0.6494185	635.6729	3.55276
450	20	420.4647	372.5575	0.7533571	0.6363819	646.9561	3.710567	0	412.9641	375.7278	0.5980012	0.6496475	635.6643	3.553964
466	21	418.5887	366.1626	0.7533312	0.6367543	646.8282	3.744948	0	417.0449	370.9076	0.5978514	0.6495544	645.8091	3.641268
500	22	448.1517	426.7739	0.7534814	0.6373928	646.251	3.728294	0	425.8824	357.3699	0.5980558	0.6505944	634.8905	3.528945
516	23	425.5454	366.0304	0.7536324	0.6375493	646.0896	3.677925	0	424.6074	420.3143	0.5981955	0.6506779	634.8333	3.661961
533	24	407.2442	382.1878	0.7539406	0.6375214	646.032	3.724841	0	362.4283	368.4	0.5989246	0.6505076	634.8312	3.545217
550	25	382.0437	376.647	0.7543342	0.6372561	645.8999	3.731122	0	346.7462	378.2516	0.5996398	0.6500002	634.6049	3.559673
566	26	370.777	371.6201	0.7546184	0.637073	645.6644	3.765136	0	326.4905	386.1813	0.5997646	0.6495686	634.5554	3.610301
583	27	365.62	362.2698	0.7550178	0.6368808	645.6926	3.770904	0	345.1357	379.1794	0.6001299	0.6490359	634.3461	3.595069
600	28	371.8418	361.1357	0.7553933	0.636552	645.5532	3.784081	0	333.3999	378.8482	0.6004177	0.6485349	634.0366	3.59077
616	29	352.739	377.1207	0.7557038	0.6364886	645.5298	3.792601	0	327.5732	374.5496	0.6007668	0.6482773	634.3557	3.644621
633	30	364.7997	378.6232	0.7559824	0.6365534	645.5516	3.81725	0	331.7	389.1884	0.6010883	0.6481025	634.0649	3.648695
650	31	353.9612	370.7148	0.7562348	0.6367001	645.4138	3.871641	0	348.1435	390.4786	0.6013653	0.648073	633.7674	3.663214
666	32	358.9431	359.4175	0.7565078	0.6369652	645.5173	3.876763	0	337.5291	386.1313	0.6016595	0.6480598	634.0999	3.663415
683	33	353.6321	369.5632	0.7568166	0.6370528	645.5229	3.863897	0	343.475	391.1053	0.6019519	0.6480272	633.6867	3.694201
700	34	353.5806	366.2641	0.7569903	0.6370781	645.5316	3.89461	0	346.4277	378.5706	0.6021855	0.6477951	634.028	3.689347
716	35	357.3749	368.0299	0.7571286	0.6369992	645.3062	3.907702	0	336.7175	406.1323	0.6023462	0.6476824	633.6699	3.730237
733	36	353.7257	374.9613	0.7573475	0.6370668	645.4896	3.919349	0	330.6978	388.4258	0.6024563	0.6477064	633.4949	3.735837
750	37	352.8519	373.0769	0.7575161	0.637355	645.4871	3.929201	0	340.0563	371.7014	0.6026729	0.6479211	634.0341	3.73941
766	38	351.5531	366.0008	0.7575135	0.6377777	645.552	3.9787	0	360.0238	400.5162	0.6023919	0.6483628	633.5637	3.778778
783	39	448.1368	350.5203	0.7569316	0.6383128	645.7051	3.980263	0	490.351	388.9582	0.6014163	0.6489347	633.8975	3.786376
800	40	476.1451	336.3235	0.7566356	0.6389554	645.7972	3.981491	0	503.899	390.9497	0.6012101	0.6494787	634.3527	3.814618
816	41	482.7939	350.1748	0.7566625	0.6394882	645.7811	3.990834	0	497.8269	396.0833	0.6013622	0.6498792	634.0419	3.818946

FIGURE B.3: Tableau de données typique fourni par un eye tracker[24].

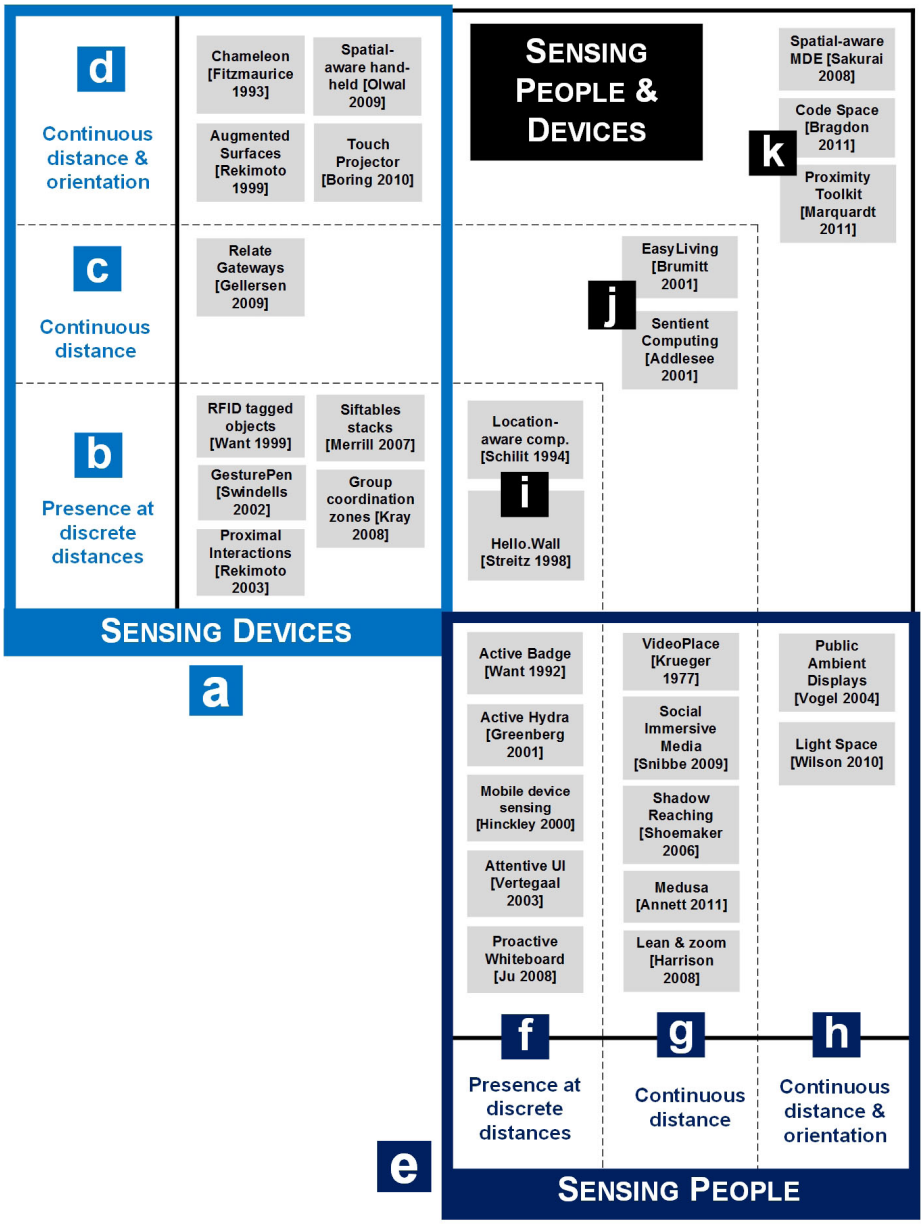

FIGURE B.4: Tableau des différentes façons de capter des données de distance[17].

Annexe C

Code source des implémentations

Eye tracking et SA

```
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#         Listener to iMotions and use of PyGaze                    #
#                     Writen by Amelie                              #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #


# # # # # # # # # # # #
# Connect to iMotions
s = socket.socket()

host = 'localhost'
port = 11000

s.connect((host, port))


# # # # # # # # # # # #
# Connect to PyGaze

# visuals
disp = Display()
scr = Screen()

# input
kb = Keyboard()
tracker = EyeTracker(disp, trackertype=TRACKERTYPE, resolution=DISPSIZE,
                eyedatafile=LOGFILENAME, logfile=LOGFILE, fgc=FGC, bgc=BGC)

# display instructions
```

```
scr.draw_text(text="Press any key to start the calibration.",
                fontsize=TEXTSIZE)
disp.fill(scr)
disp.show()

# wait for a keypress
kb.get_key(keylist=None, timeout=None, flush=True)

tracker.calibrate()

# perform a drift check
tracker.drift_correction()

# close the Display
disp.close()

# writing in a file to see better the data
f = open("testFile.txt", 'w')

recording = False

# # # # # # # # # # # #
# Collect the data
while True:
  # receiving the data from iMotions
  data = s.recv(2048)

  f.write(data)

  # splitting the data to every ";"
  separated = data.split(';')
  # the event type is the 3rd data in the line
  eventType = separated[2]

  # when a new image is displayed
  if eventType == 'SlideStart':
    # clean old data
    fixTStart = []
    fixTStop = []
    fixPos = []
    totalPos = []

    print '\n old data cleared'

    # getting the image name so we can load the new AoI
    imageName = separated[6]
    imageType = separated[7]

    aoiName = imageName + '.xml'
```

```
    # initialize a new AOI
    # AoIs xml file have to have the same name as the image file
    aoi, aoiListD = ParsingAOI.parseAoi(os.path.join(AOIDIR, aoiName),
                          DISPSIZE)
    aoiDist = SA.dist_aoiList(aoiListD)

    print 'AoI loaded'

    # start tracking
    tracker.start_recording()
    recording = True

    print 'Start recording \n'

    # initiate the timer
    initTimer = timer.get_time()

  # when the image is not displayed anymore
  elif eventType == 'SlideEnd' and recording:
    tracker.stop_recording()
    recording = False

    print 'Stop recording'

  # when we can collect eye tracker data
  elif eventType == 'EyeData' and recording:
    # waiting for fixation starting and ending time and position
    timeStart, timeStop, pos = Useful.fixation_complete_method(tracker, totalPos)

    fixTStart.append(timeStart)
    fixTStop.append(timeStop)
    fixPos.append(pos)

    # every COMPUTETIME spend, we start to compute the situation awareness
    if timer.get_time() - initTimer > COMPUTETIME:
      print '\n Start computing \n'

      dwellTime = SA.dwell_time(fixTStart, fixTStop, fixPos, aoi) # dwell time
      matrix = SA.trans_matrix(fixPos, aoi) # transition matrix
      entro = SA.entropy(SA.prob_matrix(matrix)) # entropy
      distanceF = SA.dist_fix(fixPos) # distance between 2 fixations
      distanceA = SA.totalDist(matrix, aoiDist) # distance between AoIs like asked

      # sliding window: get rid of the first data
      fixTStart.pop(0)
      fixTStop.pop(0)
      fixPos.pop(0)

      # displaying the value in the console
      print ('Dwell time = ' + str(dwellTime))
```

```
        print ('Entropy = ' + str(entro))
        print ('Distance between fixation = ' + str(distanceF))
        print ('Distance between AoIs = ' + str(distanceA))

s.close()
f.close()
```

Code C.1: Listener. Genre de main qui permet de se connecter à iMotions, d'utiliser PyGaze pour se connecter à l'eye tracker de type Tobii Pro et de lancer les calculs de SA.

```
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#            Computing the Situation Awareness                        #
#                    Writen by Amélie                                 #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #


# this method is used to compute the dwell time on each AOI
def dwell_time(start, stop, pos, aoiList):
  # checking we have the same data for the same fixation
  if len(start) != len(stop) or len(start) != len(pos) :
    print 'Data not valid!'
    return -1

  # zipping the corresponding measures together so it is easier to compute
  #  and avoid mistakes
  fixation = zip(start, stop, pos)

  # dwell time expressed as a list of dwell times for each AOI
  dt = []

  # looping on each AOI to compute each different dwell time
  for ao in aoiList:
    time = 0

    # looping on each fixation to figure out if they are part of the current AOI
    for fix in fixation:
      if ao[1].contains(fix[2]):
        # dwell time = sum of each fixation time (ending time - starting time)
        #  on the AOI
        time = time + (fix[1] - fix[0])

    dt.append(time)

  return dt


# this method is used to compute the transition matrix for the AOI
def trans_matrix(fixPos, aoiList):
  i = 1

  # transition matrix expressed as a list of the (previous AOI,
  #  current AOI) and the number of time this happened
  trans = []

  # used to know the previous AOI where there was a fixation
  aoiPrev = None
```

```
# looping on each fixation to compute the transitions
while i < len(fixPos):
  # looping on each AOI to know where the previous fixation was
  for ao in aoiList:
    if ao[1].contains(fixPos[i - 1]):
      # keeping track of the previous AOI where the previous fixation was
      aoiPrev = ao

  # looping on each AOI to know where the current fixation is
  for a in aoiList:
    # if the current fixation is in a AOI then check that it's not the
    #  same as the previous one since we don't want to have transition
    #         for one AOI to the same one
    if a[1].contains(fixPos[i]) and aoiPrev is not None and aoiPrev[0] != a[0]:
      # keep track of the two AOI (previous one to the current one)
      aoiTrans = (aoiPrev[0], a[0])

      found = False

      # if empty list of transition, add the current transition
      if not trans:
        element = (aoiTrans, 1)
        trans.append(element)

      else :
        # checking if the transition already happened in the past so
        #  we can update the counter
        for elem in trans:
          if elem[0] == aoiTrans:
            # updating the counter
            count = elem[1] + 1
            # removing the previous transition so we don't have double
            trans.remove(elem)
            trans.append((elem[0], count))

            found = True

          # if the transition never happened before then adding it and
          #  starting counting from 1
          if not found:
            element = (aoiTrans, 1)
            trans.append(element)

      # once we compute the transtion, the previous one has to be forgotten
      aoiPrev = None

  i = i + 1

return trans
```

```
# this method is used to define the matrix of probabilities
def prob_matrix(trans):
  prob = []

  total = 0

  # we need the total amount of transition between all the AoIs
  for elem in trans:
    total = total + elem[1]

  # for each transition, we compute the probability as number of transition / total
  for elem in trans:
    count = elem[1] / float(total)

    prob.append((elem[0], count))

  return prob


# this method is used to define the entropy
def entropy(prob):
  sum = 0

  for elem in prob:
    # based on the formula we compute probability * log2(probability)
    log2 = math.log(elem[1], 2)
    product = elem[1] * log2

    # we sum every result from each AoI together
    sum = sum + product

  # the entropy is -1 * the sum
  return -1 * sum


# this method is used to compute the distance between fixations
def dist_fix(fixPosList):
  distance = []

  i = 1

  # we compute the distance between the previous and the current fixations
  while i < len(fixPosList):
    # based on the eucledian distance formula
    x = fixPosList[i][0] - fixPosList[i-1][0]
    y = fixPosList[i][1] - fixPosList[i-1][1]

    dist = math.sqrt((x ** 2) + (y ** 2))
    distance.append(dist)
```

```
    i = i + 1

  return distance


# this method is used to compute the distance between two AoIs
def dist_aoi(aoi1, aoi2):

  # get the aoi coordinates (xa, ya) (xb, yb) for the 2 vertices
  xa1 = aoi1[0]
  xb1 = aoi1[2]
  ya1 = aoi1[1]
  yb1 = aoi1[3]

  xa2 = aoi2[0]
  xb2 = aoi2[2]
  ya2 = aoi2[1]
  yb2 = aoi2[3]

  # base value
  x = 0
  y = 0

  # should we compute to the left, right, top or bottom
  left = xb2 < xa1
  right = xb1 < xa2
  bottom = yb2 < ya1
  top = yb1 < ya2

  # get the x value based on left or right
  if left:
    x = xa1 - xb2
  elif right:
    x = xb1 - xa2

  # get the y value based on top or bottom
  if bottom:
    y = ya1 - yb2
  elif top:
    y = yb1 - ya2

  # eucledian distance formula
  dist = math.sqrt((x ** 2) + (y ** 2))

  return dist


# this method is used to compute the distance for all the AoIs
def dist_aoiList(aoiList):
```

```
  distance = []

  for aoi in aoiList:
    for ao in aoiList:
    # avoid computing the distance for the same AoI or something already computed
      if ao == aoi:
        break

      # making sure the label is written by a convention (smaller, bigger) numbers
      if aoi[4] < ao[4]:
        distance.append(((aoi[4], ao[4]),dist_aoi(aoi, ao)))
      else:
        distance.append(((ao[4], aoi[4]), dist_aoi(aoi, ao)))

  return distance


# this method is used to compute the total distance between the AoIs
def totalDist(transMatrix, aoiDist):
  total = 0

  for aoi in aoiDist:

    for elem in transMatrix:

      # compare the labels
      if(aoi[0] == elem[0]):

        # compute distance between AoIs * number of time doing the same path
        dist = aoi[1] * elem[1]
        total = total + dist

  return total
```

Code C.2: SituationAwareness. Tous les calculs de la SA sont définis ici.