MASTER EN SCIENCES INFORMATIQUES À FINALITÉ SPÉCIALISÉE EN SOFTWARE ENGINEERING

Interactions gestuelles en réalité augmentée appliquée au cas de l'analyse structurale

Beersaerts, Jonathan

Award date:
2018

Awarding institution:
Universite de Namur

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

Interactions gestuelles en réalité augmentée appliquée au cas de l'analyse structurale

Jonathan Beersaerts

Maître de stage : Pr Isabelle Linden

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

Co-promoteur : Pr Anne Wallemacq

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

Résumé

Durant ces dernières années, la réalité virtuelle et augmentée a gagné en popularité, tant auprès des entreprises que des particuliers, grâce à la disponibilité de périphériques facilitant l'accès à de telles technologies. Dans cette optique, le projet EFFaTA-MeM considère notamment les fonctionnalités offertes par la réalité augmentée afin de proposer des outils de visualisation et d'interaction pour l'analyse structurale de texte. Ce travail a pour objectif d'explorer les capacités du casque Microsoft HoloLens par le biais de la conception de visualisations, et leur manipulation, pour assister les analystes. Les résultats de cette exploration sont l'élaboration d'un prototype de visualisation pour lequel un ensemble de gestes de manipulation est défini et nécessite une extension des capacités de détection et reconnaissance de gestes de l'HoloLens, réalisable à l'aide de deux architectures logicielles proposées. Par ailleurs, une preuve de concept avec un dispositif de détection des mains Leap Motion est réalisé et détaillé dans ce travail. Enfin, les résultats présentés ouvrent la voie à des travaux futurs tels que l'exploration d'autres paradigmes d'interactions ou encore l'étude de méthodes d'apprentissage automatique dédiées à la définition de gestes pour en faciliter le processus de reconnaissance.

In recent years, virtual and augmented reality increased in popularity among both businesses and individuals, thanks to the availability of devices that facilitate access to this kind of technology. In this perspective, the EFFaTA-MeM research project notably considers the functionalities offered by augmented reality in order to design visualization and interaction tools for structural text analysis. This work aims to explore the Microsoft HoloLens headset capabilities through the design of visualizations, and their manipulation, to assist analysts. The results of this exploration are the development of a visualization's prototype for which a set of manipulation gestures is defined and requires an extension of gesture detection and recognition capabilities of the HoloLens, which can be implemented using two proposed software architectures. In addition, a proof of concept using a Leap Motion hand detection device is performed and detailed in this document. Finally, the presented results pave the way for future work such as the exploration of other paradigms of interactions or the study of machine learning methods dedicated to the definition of gestures to ease their recognition process.

Remerciements

Je tiens tout d'abord à remercier mon promoteur Pr Bruno Dumas, de l'Université de Namur, pour sa bienveillance et sa disponibilité durant la réalisation de ce travail. J'ai toujours pu compter sur son soutien et ses conseils pour progresser dans la rédaction de ce mémoire. Je suis fier du résultat et cette fierté lui est due.

Je voudrais également remercier ma co-promotrice Pr Anne Wallemacq, de l'Université de Namur, pour sa spontanéité et l'ouverture d'esprit qu'elle m'a apportée. Ce travail s'inscrivant dans un contexte interdisciplinaire, ses conseils et nos réunions de travail m'ont permis de m'aventurer dans les nuances, l'imaginaire et la créativité, des concepts souvent éloignés de la structure rassurante de mes compétences informatiques.

De plus, l'expression de ma gratitude va également à mon maître de stage Pr Isabelle Linden, de l'Université de Namur, pour son accueil, son accompagnement et sa supervision durant le stage de 3 mois effectué au sein du projet EFFaTA-MeM. Tout en me laissant de l'autonomie dans la réalisation de mon travail, elle a toujours été présente pour répondre à mes interrogations et faire de mon stage une expérience fructueuse et positive que je garderai en mémoire.

Je remercie par ailleurs l'ensemble des membres du projet EFFaTA-MeM (Pr Bruno Dumas, Pr Anne Wallemacq, Pr Isabelle Linden, et Pr Guy Deville) pour leur accueil bienveillant au sein du projet ainsi que Christian Collot avec qui j'ai partagé mon bureau durant mon stage.

Enfin, je remercie mes collègues étudiants et ma famille pour leur écoute et leur soutien durant cette dernière étape importante et difficile de mon parcours universitaire.

Table des matières

Chapitre 1

Introduction

De Star Wars à Iron Man, en passant par Minority Report, l'affichage d'éléments virtuels, ou hologrammes, au sein du monde réel est courante au sein des oeuvres de fiction, mais également sujette à de nombreuses recherches scientifiques dans les dernières décennies. Depuis l'épée de Damoclès d'Ivan Sutherland, considérée comme le premier système de réalité augmentée, l'évolution technologique et les recherches menées ont permis de se rapprocher de la fiction tant d'un point de vue de l'affichage que des interactions avec des hologrammes.

À l'aube de l'industrie 4.0 intégrant les nouvelles technologies au sein du monde industriel (Posada *et al.*, 2015), des entreprises comme Renault intègrent désormais la réalité augmentée au sein de leur processus métier, de leur chaine de production ou pour de la visualisation d'informations. Rapidement identifiée comme un domaine d'application de la réalité augmentée (Azuma, 1997), des opérations chirurgicales sont dès à présent menées avec l'assistance de cette technologie (Microsoft, 2017).

En parallèle, le projet EFFaTA-MeM (Evocative Framework For Text Analysis – Mediality Models) vise, par une approche transdisciplinaire, à concevoir un ensemble d'outils innovants destinés à l'analyse et l'interprétation de texte. En combinant les approches des sciences humaines et des sciences informatiques, les membres du projet cherchent à tirer le meilleur des deux disciplines en élaborant des outils de visualisation, d'automatisation et d'interaction facilitant le travail des analystes de textes à extraire les concepts, les sens cachés ou encore le paysage sémantique soutenant un texte (discours, histoire...).

Durant son processus de recherche, le projet EFFaTA-MeM a concentré son exploration sur l'analyse structurale définie dans (Piret *et al.*, 1996). Cette technique d'analyse a pour objectif de comprendre, à l'aide d'un ensemble de disjonctions entre deux termes du texte, les significations implicites se dissimulant dans ce dernier. Un des résultats de cette exploration est l'outil STAVIZ (Clarinval, 2017) qui permet notamment l'analyse d'un texte via l'encodage de relations d'associations ou d'oppositions entre des termes ainsi qu'un ensemble de visualisations en deux dimensions telles que des nuages de mots, des matrices de relations ou encore des diagrammes Node-Link.

Dès lors, le projet EFFaTA-MeM désire explorer les possibilités de visualisation et d'interaction offertes par la réalité augmentée à l'aide du casque Microsoft HoloLens. En proposant une détection de l'environnement, la visualisation d'hologrammes au sein du monde réel et des méthodes d'interactions gestuelles

vocales et avec le regard, l'HoloLens incarne potentiellement un outil pertinent pour les recherches investiguées dans le cadre de ce projet.

Par conséquent, le travail présenté dans ce document est motivé par les deux questions de recherches suivantes :

— Quelles visualisations de réalité augmentée peuvent être élaborées pour l'analyse structurale avec le casque HoloLens ?

— Quelles sont les méthodes d'interactions gestuelles, appliquées à un cas de manipulation de visualisations, pouvant être mises en oeuvre avec le casque Microsoft HoloLens ?

Pour répondre à ces deux questions de recherche, le travail de recherche est effectué en deux étapes réparties sur l'ensemble de l'année académique 2017-2018. Le premier quadrimestre de l'année académique fut consacré à un stage au sein du projet EFFaTA-MeM durant lequel une expérimentation des fonctionnalités du casque HoloLens fut réalisée à travers le développement d'un premier prototype de visualisation de cathédrales de texte. Ensuite, durant le deuxième quadrimestre, un état de l'art de la réalité augmentée fut réalisé en plus de la conception d'un second prototype de visualisation de mobiles artistiques mettant en oeuvre les différentes unités d'analyse de la technique d'analyse structurale. Ce second prototype ayant besoin d'une extension des interactions gestuelles, une architecture logicielle est élaborée et sa faisabilité est démontrée par la réalisation d'une preuve de concept étendant les gestes reconnus par le casque HoloLens.

L'organisation de ce mémoire est définie comme suit. Le chapitre 2 expose l'état de l'art mené sur la réalité augmentée et les méthodes d'interactions gestuelles en plus d'une présentation détaillée du casque de réalité augmenté Microsoft HoloLens. Le chapitre 3 détaille les deux prototypes de visualisation développés pour l'HoloLens, l'élaboration d'interactions gestuelles étendues pour ce dernier ainsi que la preuve de concept réalisée. Enfin, le chapitre 4 conclura ce mémoire en synthétisant les contributions du travail et les travaux futurs de la recherche effectuée.

Chapitre 2

Réalité augmentée

2.1 Définitions et taxonomies

Dans son état de l'art sur la réalité augmentée (Azuma, 1997), Azuma présente l'AR comme une variante de la réalité virtuelle (VR). Là où la VR plonge un utilisateur au sein d'un environnement entièrement virtuel où il n'est plus en mesure de voir le monde réel, l'AR sert un tout autre propos en permettant à un utilisateur de visualiser des éléments virtuels superposés au monde réel. Azuma propose une définition de la réalité augmentée comme une technologie se conformant aux trois spécifications suivantes :

1. Combinaison du réel et du virtuel
2. Interaction en temps réel
3. S'inscrit dans un espace en trois dimensions.

Avec cette définition, Azuma se concentre sur les fonctionnalités et caractéristiques d'un système de réalité augmenté. Ce qui lui permet de proposer une définition ne se limitant pas à une ou plusieurs technologies relatives à la mise en œuvre de tels systèmes.

Dans (Azuma, 1997), les motivations de l'AR sont détaillées comme un moyen d'enrichir la perception et l'interaction de l'utilisateur avec le monde réel. Et ce dans le but de faciliter la réalisation de tâches au sein de ce dernier. Ces motivations renforcent la pertinence de la définition proposée par Azuma. En effet, l'amélioration de la perception de l'utilisateur justifie la combinaison d'éléments réels et virtuels (1). De plus, pour faciliter la réalisation de tâche au sein du monde réel, il est nécessaire de disposer d'un système fonctionnant en temps réel (2) et s'inscrivant dans un espace en trois dimensions (3) conformément au monde réel.

Par ailleurs, les éléments de la définition, proposée par Azuma, s'appliquent au concept d'interaction augmentée décrit par Rekimoto et Nagao comme un style d'interaction homme-machine (IHM) avec le monde réel (Rekimoto et Nagao, 1995). Ce concept vise à rendre l'interface invisible entre l'utilisateur et l'ordinateur. L'utilisateur pouvant alors interagir avec le monde réel augmenté par les informations virtuelles ajoutées par l'ordinateur, comme illustré par la figure 2.1(d) où l'interaction « Homme ⇔ Monde Réel » est matérialisée par la combinaison des interactions « Homme ⇔ Machine » et « Machine ⇔ Monde Réel ». La figure 2.1(a) met en lumière la séparation existant entre l'utilisateur

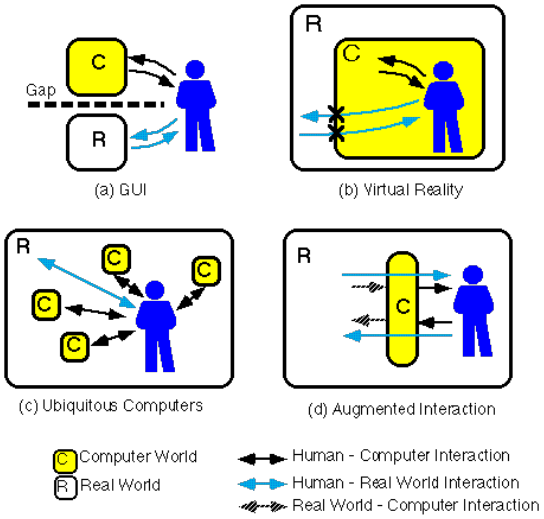

FIGURE 2.1 – Comparaison des styles d'IHM (Rekimoto et Nagao, 1995)

et l'ordinateur dans le cas d'une IHM traditionnelle (GUI), séparation que la réalité augmentée tente idéalement de supprimer. Par ailleurs, la figure 2.1 illustre les différents styles d'IHM comparés par Rekimoto et Nagao. Nous y retrouvons la VR (b) qui montre clairement une immersion complète de l'utilisateur au sein du monde virtuel. Notons que cette immersion est entre autres assurée par le blocage des interactions « Homme ⇔ Monde Réel » par le système.

D'autre part, Milgram et Kishino introduisent la notion de Réalité Mixte (MR) au sein d'un *Virtuality Continuum* illustré par la figure 2.2 (Milgram et Kishino, 1994). À son extrême gauche, nous retrouvons les *Real Environments* (RE) comportant uniquement des éléments réels tandis qu'à son extrême droite se trouvent les *Virtual Environments* (VE) totalement virtuels. Notons que Milgram et Kishino incluent l'observation d'une scène du monde réel à travers un affichage vidéo, mais également la vue directe de la scène sans aucun moyen technologique, comme faisant partie des RE à l'extrême gauche du *Virtuality Continuum*.

FIGURE 2.2 – Virtuality Continuum (Milgram et Kishino, 1994)

Milgram et Kishino définissent les environnements de MR comme la fusion d'éléments réels et virtuels dans un affichage et se situant donc entre les deux extrémités du *Virtuality Continuum*. Comme mentionné sur la figure 2.2, Milgram et Kishino disposent l'AR comme une réalité mixte tendant vers les RE et par conséquent, elle représente un environnement réel agrémenté d'éléments virtuels. Par ailleurs, Milgram et Kishino décrivent l'*Augmented Virtuality (AV)*

comme une MR tendant vers les VE et donc proposant un monde virtuel enrichi d'éléments réels. Concernant l'AR, Billinghurst *et al.* concluront que la leçon majeure de (Milgram et Kishino, 1994) est que l'AR ne se situe pas sur un point précis du *Virtuality Continuum*, mais bien n'importe où entre ses deux extrêmes (Billinghurst *et al.*, 2015).

2.2 Visualisations de données

Dans le domaine de la visualisation de données, l'utilisation de visualisation en trois dimensions est sujette à de nombreuses recherches et évaluation au sein du monde scientifique. En effet, St. John *et al.* concluront dans (St. John *et al.*, 2001) que les vues en trois dimensions facilitent la compréhension des formes au détriment de l'évaluation des positions relatives dues aux distorsions inhérentes des affichages 3D. Ce type de distorsions est nommé l'ambiguïté de ligne de vue. St. John *et al.* justifieront cette facilité de compréhension des formes par la mise en œuvre des trois dimensions dans une seule vue, l'apport de repères de profondeurs additionnels et la représentation de caractéristiques qui seraient invisibles dans une vue 2D. Ces conclusions seront notamment reprises par Munzner dans son ouvrage au sujet de l'analyse et la conception de visualisations (Munzner, 2014). Par ailleurs, Wang et Mueller constateront que des tâches d'exploration de données à hautes dimensions avec des projections 2D augmentent l'effort cognitif nécessaire contrairement aux affichages en 3D sur les mêmes ensembles de données (Wang et Mueller, 2015).

Dans (Brath, 2014), Brath effectue une révision de différentes visualisations d'informations 3D tout en détaillant les attributs intrinsèques de la 3D. Il conclura que des visualisations 3D spécialisées ont été mises en œuvre avec succès dans des domaines tels que l'utilisation de cubes spatio-temporels par les forces de l'ordre, ou encore la finance. Par ailleurs, il détaillera plusieurs problèmes et défis devant encore être résolus concernant la visualisation 3D (liste non exhaustive) :

Navigation L'exécution de différentes opérations telles que l'agrandissement de la scène, la sélection d'objets ou encore la rotation de la scène nécessitera, selon Brath, la création de modes ou contextes pour différencier l'interaction voulue par l'utilisateur, ce qui complexifie l'utilisation du système par rapport à une scène 2D.

Occlusion Dépendant des dimensions, de la densité ou de la concentration des objets représentés, certains sont susceptibles d'être masqués par d'autres, ce qui nuit à la bonne compréhension de la visualisation. Bien que l'occlusion d'éléments soit une problématique qui se retrouve dans les visualisations 3D, Brath rappellera que les visualisations en 2D peuvent être concernées également.

Sélection et manipulation Brath présente ce défi comme une conséquence des deux précédents. Si l'action de « cliquer-déplacer » est utilisée pour la navigation au sein de la visualisation, la création d'un rectangle de sélection, par exemple, ne sera pas disponible avec cette action. De plus, la sélection d'éléments occlus par d'autres se voit également complexifiée. Il en est de même pour la manipulation d'objets 3D étant donné les degrés de liberté plus importants apportés par la 3D.

Points sans ancrage En cas d'absence d'interactions ou de mouvements, les nuages de points 3D compliquent la localisation de points quelconques dans l'espace si aucun repère supplémentaire n'est mis en œuvre (ombres, tracés de références...). Brath mentionne néanmoins que certains types de données ne sont pas concernés par cette problématique, leurs structures pouvant fournir les indications nécessaires à la bonne compréhension des relations spatiales en présence.

Perspective La perception de la perspective peut également être biaisée au sein d'une visualisation 3D. En effet, en considérant deux objets de la même forme, l'un situé au premier plan et l'autre en arrière-plan, selon le champ de vision, ils peuvent apparaître de taille identiques alors que l'objet en arrière-plan est d'une taille supérieure.

Bien que les travaux de St. John *et al.*, Wang et Mueller et Brath ne concernent pas la réalité augmentée, selon la définition d'Azuma et les différentes conclusions présentées dans (St. John *et al.*, 2001; Wang et Mueller, 2015; Brath, 2014), l'AR incarne un outil prometteur pour la visualisation d'informations en trois dimensions. Combinée avec des méthodes d'interactions adaptées, elle est susceptible de proposer des solutions aux problèmes énoncés par Brath. Dans la section suivante, nous détaillerons différentes contributions pouvant apporter des éléments de réponses quant à l'utilisation de visualisations 3D avec l'AR.

2.2.1 Contributions de l'AR

Dans (Slay *et al.*, 2001), Slay *et al.* décrivent la mise en œuvre de la réalité augmentée au sein du système de visualisation InVision, un environnement de visualisation développé pour étudier les problèmes relatifs à l'assemblage et au déploiement de systèmes de visualisation adaptatifs (Pattison *et al.*, 2001). À l'aide des technologies de Computer Vision [1], Slay *et al.* utilisent des marqueurs de référence qui permettent de visualiser l'espace de travail InVision en réalité augmentée comme illustré par la figure 2.3.

Dès lors, l'utilisateur est en mesure d'inspecter le modèle, de tous les points de vue, en pivotant ou en changeant l'orientation du marqueur. Cette technique a également été utilisée pour la sélection d'éléments de la visualisation. En attachant un marqueur prédéfini sur un dispositif de pointage, le système est capable de suivre sa position et un rayon virtuel est affiché lorsque le bouton du périphérique est actionné. Par conséquent, la contribution présentée dans (Slay *et al.*, 2001) dispose d'une interface AR tangible proposant une solution aux problèmes de navigation et de sélection énoncés par Brath.

Par ailleurs, l'utilisation de Computer Vision et de marqueurs de référence a également été utilisée dans (Belcher *et al.*, 2003) pour explorer les effets de l'utilisation de la réalité augmentée dans l'analyse de liens d'un graphe à trois dimensions. Nous y retrouvons une interface AR tangible avec un disque, muni d'un marqueur de référence, que l'utilisateur manipule pour pivoter ou incliner le graphe visualisé, comme le montre la figure 2.4(3). Cette interface AR tangible est comparée avec la visualisation en 2D et 3D sur un écran (cf. figure 2.4(1) et (2)) et montre que les participants ont un taux d'erreur assez similaire avec les visualisations 3D (sur écran et AR), mais nettement inférieur comparé

1. Vision par ordinateur

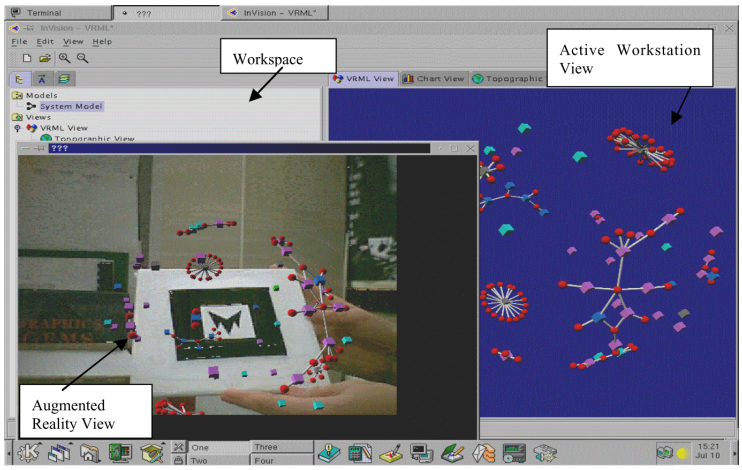

FIGURE 2.3 – Vue augmentée au sein d'InVision (Slay *et al.*, 2001)

à la visualisation 2D. L'expérience montre cependant que le temps de réponse moyen est supérieur dans l'utilisation de l'AR. Belcher *et al.* concluront qu'un des avantages majeurs des interfaces AR est son support de la métaphore d'interaction tangible permettant un mappage direct entre des éléments réels et virtuels.

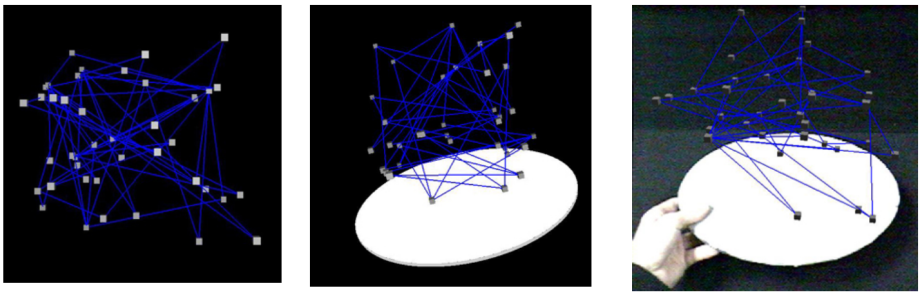

FIGURE 2.4 – Conditions de visualisation et manipulation d'un graphe à trois dimensions (Belcher *et al.*, 2003)

En reprenant les différents problèmes des visualisations 3D énoncés par Brath, nous pouvons remarquer que l'utilisation de l'AR, en combinaison avec la métaphore d'interaction tangible, peut apporter des solutions intéressantes. En effet, (Slay *et al.*, 2001) et (Belcher *et al.*, 2003) nous montrent qu'une interface AR tangible, en permettant de manipuler intuitivement la visualisation, donne à l'utilisateur la possibilité de l'observer selon différents points de vue et perspectives. Par conséquent, les désagréments tels que l'occlusion, la perspective et les points sans ancrages se voient fortement réduits grâce au contrôle dont l'utilisateur dispose sur les points et angles de vue de la visualisation.

En outre, dans (Meiguins *et al.*, 2006), un prototype de visualisation d'informations multidimensionnelles en réalité augmentée est présenté (cf. figure 2.5). Toujours par l'intermédiaire de Computer Vision et de marqueurs de référence,

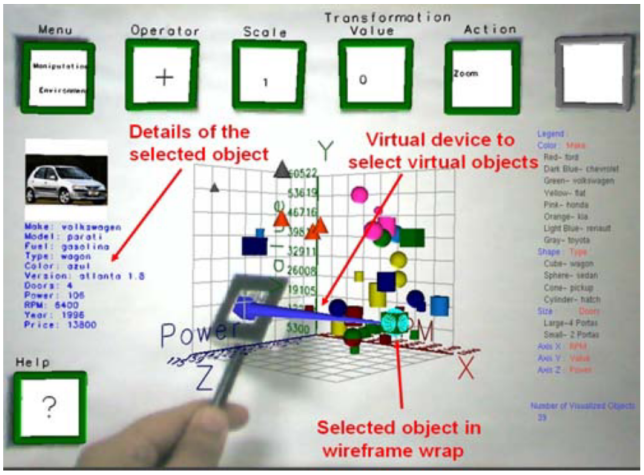

FIGURE 2.5 – Prototype de visualisation (Meiguins *et al.*, 2006)

l'utilisateur est en mesure d'effectuer des tâches comme le filtrage d'éléments, le zoom sémantique, les détails à la demande ou encore la sélection d'attributs. Ces tâches étant réalisables à l'aide d'interactions AR tangibles. En plus d'utiliser un dispositif de pointage comme nous avons pu l'observer dans (Slay *et al.*, 2001), Meiguins *et al.* ont implémenté un menu AR reposant sur plusieurs marqueurs et permettant à l'utilisateur de naviguer dans le menu en cachant le marqueur approprié. Le système de menu AR constituant une possible solution supplémentaire aux problèmes évoqués par Brath.

Enfin, une étude comparant trois environnements de visualisation de nuages de points en 3D montre que, de manière générale, l'environnement de bureau classique (écran, clavier et souris) reste le plus rapide et le plus précis, dans presque tous les cas, bien que chaque environnement soit plus efficace pour certaines tâches (Bach *et al.*, 2018). Les deux autres environnements sont des environnements de réalité augmentée soit avec une tablette portative, soit avec un casque AR (Microsoft HoloLens). Les trois environnements étudiés sont illustrés par la figure 2.6.

Pour évaluer ces trois environnements, Bach *et al.* définissent quatre tâches à effectuer sur un nuage de points et mesurent le temps nécessaire à la réalisation ainsi que le nombre d'erreurs de l'utilisateur pour chacune de ces tâches. Les tâches à effectuer sont : (1) identifier, entre deux paires de points, laquelle comporte les points les plus proches, (2) estimer le nombre de groupements de points (clusters) identifiables dans un nuage de points donné, (3) sélectionner les points mis en évidence, le plus rapidement possible et (4) ajuster un plan de découpe de façon à ce qu'il touche les 3 groupes de points indiqués. C'est cette dernière qui est mise en œuvre dans la figure 2.6.

En plus des résultats généraux de l'évaluation énoncés précédemment, Bach *et al.* concluent que l'environnement AR immersif représente la meilleure solution pour la réalisation de tâches qui requièrent des manipulations précises et que l'entrainement des utilisateurs peut encore améliorer leur performance. Par ailleurs, l'étude présentée dans (Bach *et al.*, 2018) montre également que l'utilisation de la réalité augmentée avec une tablette portative comporte les plus

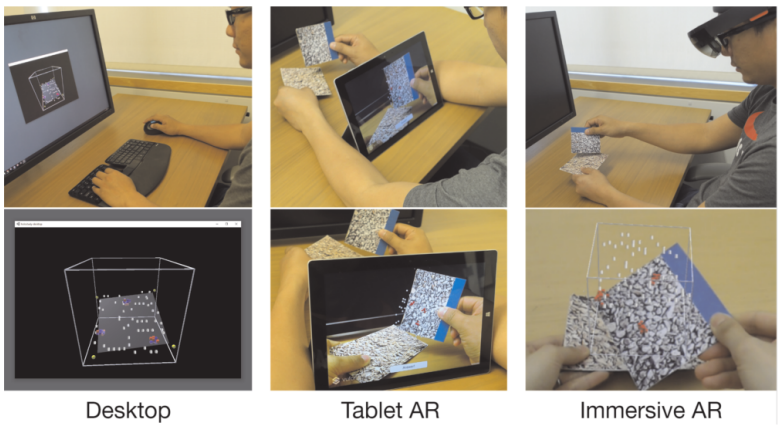

FIGURE 2.6 – Environnements à l'étude (ligne supérieure) et perspectives de l'utilisateur (ligne inférieure)(Bach *et al.*, 2018)

mauvaises performances dans la quasi-totalité des tâches. Bach *et al.* présument que le problème est causé par la faible proximité entre les espaces d'interactions et de perceptions (3D pour l'interaction et 2D pour la perception).

Pour conclure, les différents travaux présentés dans cette section démontrent l'intérêt de l'AR pour la visualisation d'informations en mettant en œuvre non seulement des visualisations 3D, mais également des interactions s'éloignant du clavier et de la souris : les interactions tangibles. Remarquons d'ailleurs l'importance des méthodes d'interactions facilitant la manipulation, la navigation, et autres opérations, sans lesquelles le simple affichage d'une visualisation 3D ne permettrait pas aux utilisateurs d'en comprendre et d'en évaluer pleinement le sens. En effet, les méthodes d'interactions élaborées et/ou utilisées dans (Slay *et al.*, 2001; Belcher *et al.*, 2003; Meiguins *et al.*, 2006; Bach *et al.*, 2018) représentent des éléments clés pour la mise en œuvre de solutions aux problèmes des visualisations 3D énoncés dans (Brath, 2014). Bien que Brath ait concentré son étude sur la visualisation 3D au moyen d'IHM traditionnelle comme définie dans le schéma de Rekimoto et Nagao (cf. figure 2.1(a)), les différentes contributions discutées dans cette section nous montrent le potentiel des technologies relatives à la réalité augmentée.

2.3 Méthodes d'interactions

Dans la section précédente, nous avons pu constater l'importance des méthodes d'interactions au sein de la réalité augmentée confirmant que l'interaction en temps réel fait partie intégrante d'un système AR (cf. (Azuma, 1997)). Dès lors, de nombreuses recherches ont été conduites à ce sujet : en 2008, sur un total de 313 documents sur l'AR publié entre 1998 et 2007, Zhou *et al.* en identifient 46 (14,7%) concernant les interactions en AR, faisant de ce domaine de recherche, le deuxième le plus abordé dans les articles recensés. Par conséquent, de nombreuses méthodes ont été élaborées pour permettre une interaction intuitive avec le contenu virtuel (Zhou *et al.*, 2008). Dans (Billinghurst *et al.*, 2015), différentes interfaces et méthodes d'interactions sont présentées et organisées

selon cinq catégories : *Information Browsers*, *3D User Interfaces*, *Tangible User Interfaces*, *Natural User Interfaces* et *Multimodal Interfaces*.

En premier lieu, la catégorie *Information Browsers* considère l'affichage AR comme une fenêtre vers un espace d'informations que l'utilisateur parcourt en la manipulant (Billinghurst *et al.*, 2015). Le système NaviCam met en oeuvre ce type d'interaction à l'aide d'un ordinateur portatif muni d'une caméra vidéo permettant à l'utilisateur de voir le monde réel agrémenté d'informations contextuelles générées par l'ordinateur (Rekimoto et Nagao, 1995). En deuxième lieu, la catégorie *3D User Interfaces* a recours à des dispositifs d'entrée tels que des souris 3D, périphériques tactiles ou encore des pointeurs de type baguette. En troisième lieu, les interfaces tangibles (*Tangible User Interfaces*) créent un mappage entre des objets réels et des éléments virtuels comme nous avons pu le voir dans la section précédente. En quatrième lieu, nous retrouvons les *Natural User Interfaces* qui repose sur la posture ou le mouvement du corps de l'utilisateur (ou certaines parties de son corps) pour interagir avec les éléments virtuels. Enfin, les interfaces multimodales (*Multimodal Interfaces*) sont présentées par Billinghurst *et al.* comme la combinaison de la parole et des gestes de l'utilisateur.

Dans cette section, sachant que le casque HoloLens propose principalement des interactions gestuelles, nous nous concentrons sur ce type d'interactions, les autres méthodes étant en dehors du domaine de recherche présenté dans ce document.

Bien que les interactions gestuelles représentent une des méthodes d'interaction de l'AR, elles ne sont pas spécifiques à cette dernière. En effet, nous retrouvons l'utilisation d'interactions gestuelles dans divers domaines tels que les écrans tactiles et les interfaces tabletop. Pour présenter une vue globale des interactions gestuelles, nous aborderons différentes définitions et taxonomies concernant les gestes et les interactions qu'ils permettent, pour ensuite détailler les méthodes d'interactions gestuelles et techniques de reconnaissances de gestes en réalité augmentée.

2.3.1 Définitions et taxonomies de gestes

Dans ses études sur le discours humain et les gestes qui l'accompagnent, McNeill définit quatre catégories de gestes : *deictic*, *metaphoric*, *iconic* et *beat* correspondant respectivement aux gestes de pointage, de représentation d'idées abstraites, d'illustration d'un objet et enfin des gestes sans forme particulière accompagnant l'énonciation d'un discours (McNeill, 1992). Bien que cette classification facilite la catégorisation des gestes, McNeill se concentre sur les gestes réalisés pendant un discours. Dès lors, ils sont indissociables de la parole. Cette relation forte sera nuancée par la définition d'un geste donnée par Kendon dans (Kendon, 2004) :

> A gesture is a form of non-verbal communication in which visible bodily actions communicate particular messages, either in place of speech or together and in parallel with words. Gestures include movement of the hands, face, or other parts of the body. Gestures differ from physical non-verbal communication that does not communicate specific messages, such as purely expressive displays, proxemics, or displays of joint attention.

Par ailleurs, concernant les interactions homme-machine, Karam et Schraefel propose une taxonomie des interactions gestuelles illustrée par la figure 2.7 (Karam et Schraefel, 2005). Nous y retrouvons différentes classifications des interactions gestuelles selon les domaines d'applications, les technologies utilisées, les réponses du système et les différents styles de gestes. Notons que la VR et l'AR sont présentes en tant que domaines d'applications des interactions gestuelles.

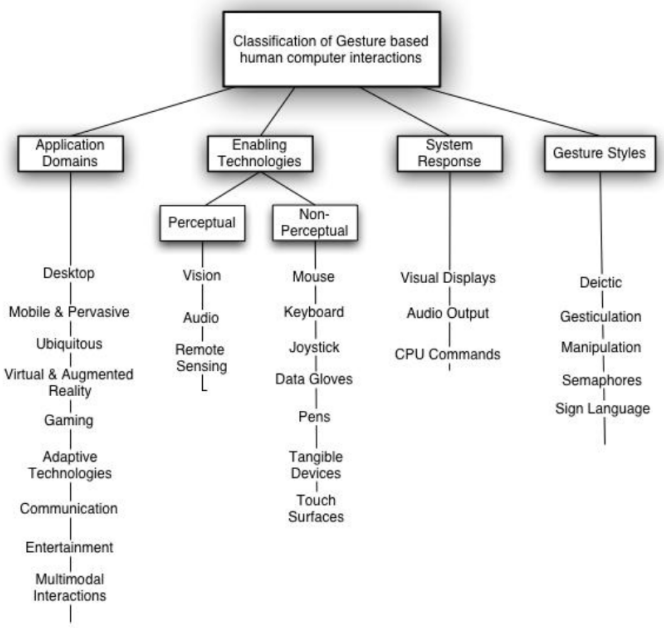

FIGURE 2.7 – Classification des interactions homme-machine gestuelles (Karam et Schraefel, 2005)

Dans cette taxonomie, Karam et Schraefel définissent une classification des styles de gestes reposant notamment sur les contributions présentées dans (McNeill, 1992; Wexelblat, 1998; Quek *et al.*, 2002) et ne reposant pas exclusivement sur l'utilisation de geste dans un discours contrairement à la classification énoncée par McNeill. Nous y retrouvons les catégories de gestes suivantes :

Déictiques Ce type de gestes implique une action de pointage pour désigner un élément ou sa position. Dès lors, l'utilisation de tels gestes permet à un utilisateur de désigner des éléments virtuels ou une position dans l'espace. Les gestes déictiques sont souvent utilisés conjointement avec la parole. L'exemple le plus réputé étant « Put that there » présenté dans (Bolt, 1980) où l'utilisateur utilise des gestes déictiques combinés à la parole pour interagir (créer, déplacer, modifier...) des éléments virtuels dans un espace à deux dimensions.

Manipulatifs Gestes effectués pour manipuler une entité. Par conséquent, il existe un lien fort entre les mouvements effectués et l'élément manipulé. À titre d'exemple, attraper un objet pour le déplacer d'une position à une autre représente un geste manipulatif. La définition de ce type de

geste est directement reprise de (Quek *et al.*, 2002). Karam et Schraefel distinguent différents types de gestes manipulatifs : (1) les gestes à deux degrés de liberté pour les interactions à deux dimensions à l'aide, par exemple, d'un stylet ou d'une souris. (2) Les gestes à multiples degrés de liberté pour les interactions à deux dimensions. (3) L'utilisation d'objets tangible pour interagir dans un espace en trois dimensions. (4) Les gestes destinés à la manipulation d'objets physique du monde réel, Karam et Schraefel donnent comme exemples la manipulation d'un bras de robot ou d'une chaise roulante.

Sémaphoriques Pour définir ce style, Karam et Schraefel s'appuient une nouvelle fois sur la définition donnée par Quek *et al.* dans (Quek *et al.*, 2002) :

> Semaphores are systems of signaling using flags, lights or arms [Brittanica.com]. By extension, we define semaphoric gestures to be any gesturing system that employs a stylized dictionary of static or dynamic hand or arm gestures...Semaphoric approaches may be referred to as "communicative" in that gestures serve as a universe of symbols to be communicated to the machine.

Dans cette définition, un geste statique représente une certaine pose du corps (le pouce en l'air par exemple) tandis qu'un geste dynamique implique des mouvements. Les gestes sémaphoriques comprennent également des gestes décrivant des traits ou marques exécutant une action prédéfinie. Nous retrouvons ce type de geste dans les interfaces tactiles avec, par exemple, le balaiement de l'écran avec un doigt pour faire défiler le contenu à l'écran ou encore le pincement/écartement de l'index et du pouce pour zoomer/dézoomer.

Gesticulation Les gestes décrits par ce style correspondent aux mouvements du corps effectués naturellement par une personne pendant une conversation. Ce style de gestes est décrit comme un des domaines de recherche les plus complexes concernant les interactions gestuelles. En effet, les gestes ne pouvant pas être définis au préalable et reposent donc sur l'analyse continue des mouvements du corps de l'interlocuteur.

Langages Cette catégorie concerne les gestes correspondant aux langages tels que la langue des signes bien que l'épellation avec les doigts peut être considérée comme un ensemble de gestes sémaphoriques. Il n'est pas question ici de gestes destinés à effectuer une action, mais à communiquer une ou plusieurs informations.

Gestes multiples Les différents systèmes étudiés dans (Karam et Schraefel, 2005) proposant différents styles de gestes combinés les uns avec les autres, cette catégorie reprend les différentes combinaisons de gestes appartenant aux catégories précédentes.

La classification présentée par Karam et Schraefel nous permet de mettre en évidence la quantité et la diversité des gestes pouvant être mis en oeuvre pour les interactions homme-machine.

Concernant les interactions gestuelles en réalité augmentée, Piumsomboon *et al.* présentent un ensemble de gestes de la main définis par des utilisateurs pour les systèmes de réalité augmentée. Sur base des résultats obtenus, ils proposent notamment une taxonomie de gestes AR illustrée par la table 2.1 (Piumsomboon *et al.*, 2013). Cette dernière étend la taxonomie de Wobbrock *et al.*

concernant les gestes de surface (*surface gestures*) présentée dans (Wobbrock *et al.*, 2009). Ces deux classifications caractérisent un geste selon différentes dimensions, chacune comportant des catégories relatives à celle à laquelle elle appartient. Dès lors, un geste est défini selon son appartenance à une catégorie de chaque dimension de la taxonomie.

Taxonomy of Gestures in AR		
Form	static pose	Hand pose is held in one location.
	dynamic pose	Hand pose changes in one location.
	static pose and path	Hand pose is held as hand relocates.
	dynamic pose and path	Hand pose changes as hand relocates.
Nature	Symbolic	Gesture visually depicts a symbol.
	physical	Gesture acts physically on objects.
	metaphorical	Gesture is metaphorical.
	abstract	Gesture mapping is arbitrary.
Binding	object-centric	Gesturing space is relative to the object.
	world-dependent	Gesturing space is relative to the physical world.
	world-independent	Gesture anywhere regardless of position in the world.
	mixed dependencies	Gesture involves multiple spaces.
Flow	Discrete	Response occurs after the gesture completion.
	continuous	Response occurs during the gesture.
Symmetry	dominant unimanual	Gesture performed by dominant hand.
	nondominant unimanual	Gesture performed by nondominant hand.
	symmetric bimanual	Gesture using both hands with the same form.
	asymmetric bimanual	Gesture using both hands with different form.
Locale	on-the-surface	Gesture involves a contact with real physical surface.
	in-the-air	Gesture occurs in the air with no physical contact.
	mixed locales	Gesture involves both locales.

TABLE 2.1 – Taxonomies des gestes en AR (Piumsomboon *et al.*, 2013)

Avec la taxonomie présentée dans (Piumsomboon *et al.*, 2013), un geste est défini par six dimensions : *form*, *nature*, *binding*, *flow*, *symmetry* et *locale*. Les quatre premières sont directement reprises de la taxonomie de Wobbrock *et al.* et seules deux catégories de la dimension *form* (*one-point touch* et *one-point-path*) ont été abandonnées étant donné l'aspect tridimensionnel des gestes en AR.

L'aspect d'un geste est caractérisé par la dimension *form*, dans laquelle Wobbrock *et al.* et Piumsomboon *et al.* identifient les gestes statiques et dynamiques, évoqués précédemment, en tenant également compte d'un éventuel déplacement dans l'espace pour chacun des deux types. Cette dimension considère chaque main séparément dans le cas d'un geste à deux mains.

La dimension *nature* s'attarde quant à elle sur la signification cognitive du geste, la sémantique qui l'accompagne. Par conséquent, un geste peut représenter un symbole, une métaphore, une manipulation physique ou une signification arbitraire. Pour chaque catégorie, Piumsomboon *et al.* donnent les exemples suivants : (*symbolic*) Pouce vers le bas pour signifier un refus ; (*physical*) Attraper un objet virtuel pour le déplacer ; (*metaphoric*) Tourner l'index dans le sens des aiguilles d'une montre pour indiquer la commande « Lecture » ; (*abstract*) Double-tap sur une surface pour désélectionner tous les objets.

Concernant la dimension *binding*, cette dernière distingue la position relative par rapport à laquelle un geste est effectué, l'espace dans lequel le geste se situe. Par conséquent, un geste peut être relatif à un objet ou à une position du monde physique, mais aussi à aucune position particulière. La catégorie *mixed dependencies* regroupant les gestes exécutés dans différents espaces.

L'objectif de la dimension *flow* est d'indiquer comment le système répond au geste, à savoir, l'action correspondante au geste doit-elle être effectuée une fois le geste terminé (*discrete*) ou être appliquée pendant l'exécution du geste (*continuous*). Dessiner un cercle avec l'index pour exécuter la commande « Lecture » est un geste appartenant à la catégorie *discrete* tandis qu'un geste manipulatif destiné à déplacer un objet est un geste *continuous*.

La dimension *symmetry* définit si le geste est effectué à une ou deux mains. D'une part, pour les gestes à une main, Piumsomboon *et al.* distinguent laquelle des deux mains est utilisée : la main dominante (*dominant unimanual*) ou l'autre (*nondominant unimanual*). D'autre part, les gestes à deux mains sont différenciés par la symétrie ou l'asymétrie du geste. Un geste à deux mains est dit *symmetric bimanual* lorsque les deux mains effectuent le même geste (cf. dimension *form*). Inversement, si elles réalisent chacune un geste différent, le geste est catégorisé comme *asymmetric bimanual*. Rappelons que cette dimension représente un des ajouts à la taxonomie de Wobbrock *et al.* avec la dimension *locale* détaillée ci-après.

Enfin, la dimension *locale* s'attarde sur la description de la position, non plus relative (cf. dimension *binding*), mais absolue, du geste dans l'espace. Dès lors, un geste est effectué soit dans les airs (*in-the-air*) soit sur une surface (*on-the-surface*) ou une combinaison des deux (*mixed locale*).

Après avoir détaillé différentes définitions et taxonomies de geste, la section suivante détaille différentes contributions ayant mis en oeuvre des systèmes de reconnaissance de geste et les techniques élaborées pour ces derniers.

2.3.2 Reconnaissance de gestes

Pour permettre à l'utilisateur d'interagir avec un système en utilisant des gestes, ce dernier doit être capable de détecter ses mains, mais également de reconnaitre le geste effectué. Dès lors, de nombreuses recherches ont été entreprises en ce sens. Dans cette section, nous détaillons les techniques de reconnaissance de gestes concernant les interactions gestuelles, avec une attention particulière pour les systèmes de réalité augmentée.

Parmi les technologies utilisées pour la reconnaissance de gestes, nous discernons deux catégories principales étant parfois utilisées ensemble : les périphériques d'interaction et la Computer Vision (CV). Pour la première catégorie, nous retrouvons l'utilisation de gants intelligents (Weissmann et Salomon, 1999; Lee *et al.*, 2010; Kumar *et al.*, 2012) ou encore de dispositifs avec accéléromètres (Mäntyjärvi *et al.*, 2004). Avec la CV, deux sous-catégories peuvent être identifiées : l'utilisation de marqueurs de référence (Buchmann *et al.*, 2004; Wang et Popović, 2009; Lee *et al.*, 2010) et la détection des mains nues (Lee et Höllerer, 2007; Van den Bergh et Van Gool, 2011; Wang *et al.*, 2011; Hilliges *et al.*, 2012; Corbett-Davies *et al.*, 2013; Piumsomboon *et al.*, 2014).

Par ailleurs, les deux catégories précédemment citées permettent de récupérer des informations brutes, qu'il est nécessaire de traiter au préalable pour détecter les gestes de l'utilisateur. Dès lors, quelle que soit la technologie uti-

lisée pour récupérer les données brutes, nous retrouvons principalement des techniques de pattern matching pour reconnaitre le geste en lui-même.

2.3.2.1 Périphériques d'interaction

Dans (Mäntyjärvi *et al.*, 2004), Mäntyjärvi *et al.* présentent une méthode d'interaction gestuelle pour commander un lecteur DVD. À l'aide d'un boitier muni d'un accéléromètre, l'utilisateur est capable d'effectuer des commandes classiques comme *Play*, *Pause* ou encore *Fast Rewind*. Les gestes supportés par le prototype sont illustrés par la figure 2.8. Ce prototype est élaboré pour appuyer la procédure d'entrainement et de reconnaissance de gestes personnalisés à l'aide d'un accéléromètre. Cette dernière repose sur l'utilisation de *Discrete Hidden Markov Model*, un modèle statistique incarnant un bon outil de modélisation de séquences à variabilités spatiales et temporelles (Mäntyjärvi *et al.*, 2004).

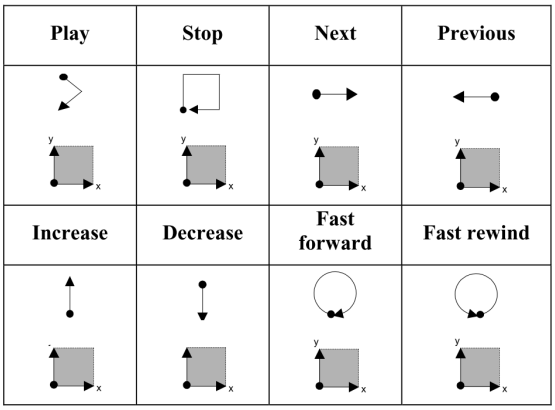

FIGURE 2.8 – Mouvements détectés à l'aide de l'accéléromètre (Mäntyjärvi *et al.*, 2004)

Par ailleurs, la détection de gestes peut être réalisée à l'aide de gants dits « intelligents », c'est-à-dire, des gants munis d'un ensemble de capteurs permettant de récupérer des données brutes sur la position ou le touché entre les doigts de la main. Ils sont soit pourvus de différents capteurs détectant la flexion des articulations des doigts (Weissmann et Salomon, 1999; Kumar *et al.*, 2012), soit constitués d'un tissu conducteur permettant de détecter le contact avec un autre (Lee *et al.*, 2010).

En utilisant le CyberGlove [2], Weissmann et Salomon sont en mesure de détecter un ensemble de 20 gestes statiques sur base des 18 mesures angulaires des articulations fournies par le CyberGlove. La détection est rendue possible par l'entrainement de plusieurs types de réseaux de neurones dont la performance de reconnaissance est évaluée dans (Weissmann et Salomon, 1999). Dans (Kumar *et al.*, 2012), un gant DG5 VHand 2.0 [3], composé d'un capteur de flexion par doigt et d'un accéléromètre, est utilisé pour permettre les actions tel que cliquer, pivoter, déplacer et pointer. De plus, grâce à l'accéléromètre, Kumar *et al.* utilisent le suivi de trajectoire pour proposer une fonctionnalité d'*air-writing*.

2. http://www.cyberglovesystems.com/
3. http://www.dg-tech.it/vhand3/index.html

Sur base des données obtenues grâce au VHand, les différents gestes supportés sont reconnus à l'aide de la méthode des *k-Nearest Neighbour (kNN)*, une méthode de classification non paramétrique. Notons que les travaux présentés dans (Weissmann et Salomon, 1999; Kumar *et al.*, 2012) ne concernent pas la réalité augmentée bien que les contributions apportées peuvent raisonnablement s'appliquer à ce contexte.

Enfin, Lee *et al.* proposent un processus alternatif à l'égard de la reconnaissance de geste avec une paire de gants (Lee *et al.*, 2010). En utilisant un tissu conductif disposé à des emplacements stratégiques des gants, la détection d'un geste n'est plus réalisée en fonction de la flexion des articulations, mais par les points de contact qui se forment entre les doigts et la paume de la main, comme le montre la figure 2.9. Avec ces différents points de contact, Lee *et al.* définissent les gestes en fonction des points actifs et inactifs. Par exemple, le geste *OK*, représenté par le pouce en l'air et le poing fermé, est caractérisé par l'activation des points de contact B, C et L, tandis que le geste *Selecting* (poing fermé et index vers l'avant) est identifié par l'activation du point K et l'absence d'activation de B.

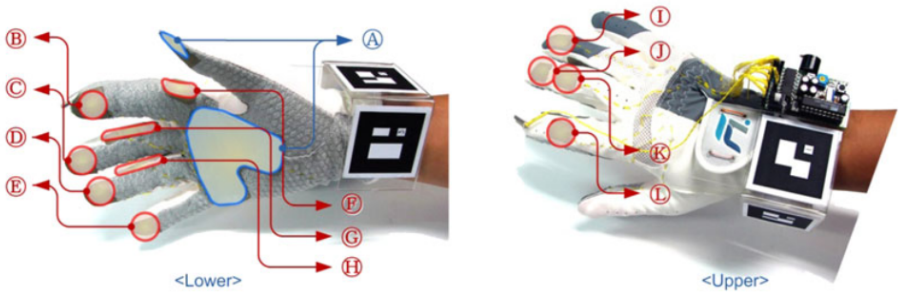

FIGURE 2.9 – Points de contact du gant droit (Lee *et al.*, 2010)

Les travaux présentés dans (Lee *et al.*, 2010) mettent en oeuvre l'utilisation des gants dans le cadre d'interactions tangibles permettant la manipulation d'objet virtuel dans un environnement de réalité mixte (cf. *Virtuality continuum* (Milgram et Kishino, 1994)). Par conséquent, les gestes supportés permettent la sélection, la copie, le déplacement et le coupage d'objets virtuels par l'intermédiaire des gants détaillés précédemment. Notons cependant que la détection de la position et l'orientation de la main de l'utilisateur n'est pas détectée par le tissu conductif des gants, mais à l'aide de Computer Vision et de marqueurs de référence situés au niveau du poignet de la paire de gants.

Pour conclure, parmi les différentes contributions présentées, nous remarquons que des capteurs de détection de flexion ou les tissus conductifs sont plus adaptés à la détection de gestes statiques (Weissmann et Salomon, 1999; Lee *et al.*, 2010), la détection de mouvement étant assurée tantôt par un accéléromètre (Mäntyjärvi *et al.*, 2004; Kumar *et al.*, 2012), tantôt par la CV (Lee *et al.*, 2010). Par ailleurs, un accéléromètre permet de détecter les mouvements, mais pas la position relative au monde physique contrairement à la CV.

2.3.2.2 Computer Vision

En ce qui concerne les techniques de Computer Vision, de nombreuses contributions concernant la réalité augmentée ont été présentées. Ces travaux se concentrent principalement sur la détection des mains de l'utilisateur soit à l'aide de marqueurs de référence soit par la détection des mains nues. Avec les marqueurs, la détection est facilitée par l'utilisation d'un gant ou d'une paire de gants, sur lesquels ils sont disposés, tandis que la détection des mains nues a recours à des processus de segmentation de la peau ou encore l'utilisation d'une caméra de profondeur voir de plusieurs caméras vidéo.

Concernant l'utilisation de marqueur de référence, les algorithmes de CV, à partir d'un flux vidéo, sont en mesure de détecter leurs positions et leurs orientations. Cette technique est utilisée par la paire de gants à tissu conductif dans (Lee *et al.*, 2010) que nous avons détaillée précédemment. Dans ce cas, la CV est uniquement utilisée pour identifier la position et l'orientation des gants dans l'espace. Cette technique fut employée différemment par Buchmann *et al.* qui implémentent le système FingARTips (Buchmann *et al.*, 2004) reposant sur la détection de geste à l'aide de trois marqueurs identifiant l'index, le pouce et la base de l'articulation entre ces derniers (cf. figure 2.10(a)). En détectant l'index et le pouce de l'utilisateur, Buchmann *et al.* mettent en oeuvre quatre types d'interaction gestuelle : la saisie d'objets virtuels, le pointage, la navigation et l'exécution de commandes. Ces différents gestes sont utilisés dans une application AR de planification urbaine (cf. figure 2.10(b)) dans laquelle l'utilisateur peut positionner un ensemble de composants urbains (bâtiments et routes) et en modifier les dimensions ou les positions.

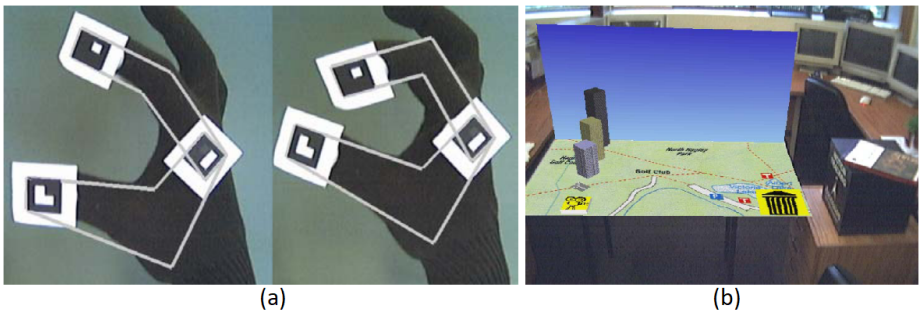

FIGURE 2.10 – FingARTips : (a) Marqueurs de référence positionnés sur un gant. (b) Application AR de planification urbaine (Buchmann *et al.*, 2004)

D'autre part, Wang et Popović présentent un système de suivi des mains en temps réel par la détection d'une paire de gants colorés dans un flux vidéo (Wang et Popović, 2009). La conception des gants a été spécifiquement étudiée pour réduire l'ambiguïté de détection pouvant survenir avec des mains nues (cf. figure 2.11). Dès lors, la conception des gants permet une estimation de la pose des mains sur base d'une seule image. Le suivi en temps réel est rendu possible par la construction d'une base de données de configurations des mains indexées par une image matricielle de la configuration. La pose de la main est ensuite identifiée ou estimée à l'aide de la méthode des plus proches voisins (*Nearest Neighbour*).

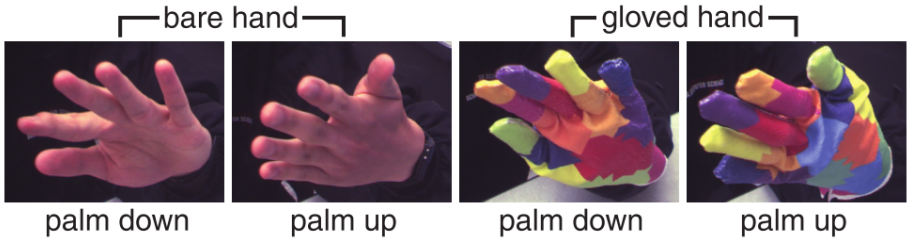

FIGURE 2.11 – Désambigüisation de détection à l'aide de gants colorés (Wang et Popović, 2009)

Par ailleurs, la détection des mains nues à l'aide de Computer Vision a été investiguée avec différentes configurations matérielles. Nous retrouvons l'utilisation d'une caméra grand public (Lee et Höllerer, 2007), de deux caméras RGB (Wang *et al.*, 2011) ou encore la combinaison d'une ou plusieurs caméras RGB avec une caméra de profondeur (Van den Bergh et Van Gool, 2011; Hilliges *et al.*, 2012; Corbett-Davies *et al.*, 2013; Piumsomboon *et al.*, 2014).

En 2007, Lee et Höllerer présentent Handy AR, un algorithme de suivi des mains reconnaissant le bout des doigts à l'aide d'une seule étape de calibrage (Lee et Höllerer, 2007). Bien que l'algorithme présenté permet d'évaluer la position et l'orientation d'une main sans marqueur de référence, la détection des bouts des doigts n'est possible qu'en maintenant la main ouverte. Par conséquent, les interactions gestuelles se limite à l'inspection d'objet virtuel en déplaçant ou pivotant la main, toujours en position ouverte. Une fois calibré, étant donné une image, l'algorithme va exécuter une segmentation de la couleur de peau pour ensuite appliquer une transformation de distance afin d'extraire un composant connexe unique de la main observée. Ce processus est illustré par la figure 2.12 et c'est à la fin de ce dernier que la position des bouts des doigts sera déterminée.

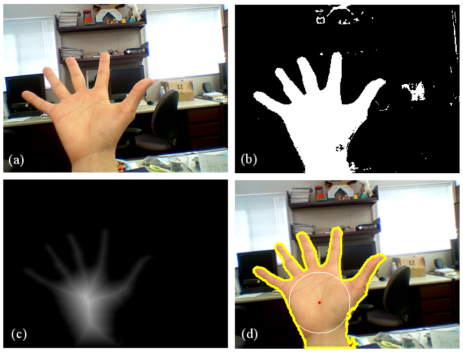

FIGURE 2.12 – Handy AR : processus de segmentation de la main (Lee et Höllerer, 2007)

Dans (Wang *et al.*, 2011), les travaux effectués avec les gants colorés (Wang et Popović, 2009) sont étendus pour permettre la détection des deux mains nues de l'utilisateur grâce au système 6D Hands. Ce dernier s'appuie sur l'utilisation

de deux webcams grand public fournissant deux points de vue différents sur les gestes que l'utilisateur exécute. Dès lors, comme pour les gants colorés, une base de données de configurations des mains est interrogée selon la méthode des plus proches voisins. Elle est cependant indexée selon deux images matricielles, correspondant aux deux points de vue des webcams, contrairement à une unique image matricielle utilisée pour les gants colorés. La figure 2.13 compare la détection d'une main avec un ou deux points de vue et illustre le gain en précision obtenu dans les travaux de Wang *et al.*.

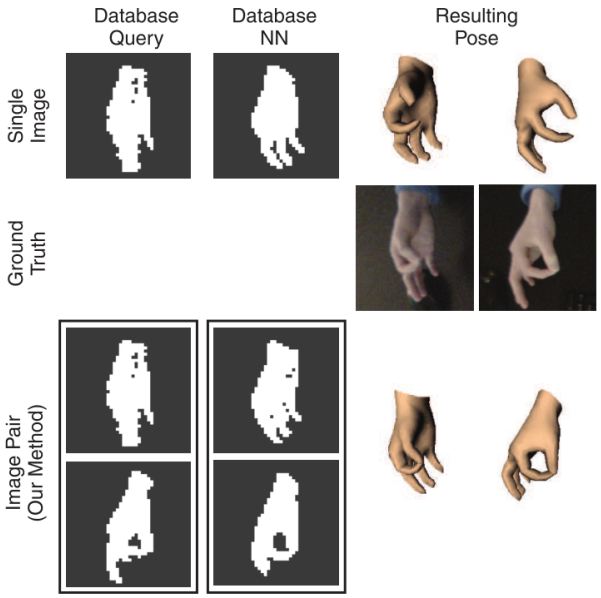

FIGURE 2.13 – Comparaison de l'estimation de la pose d'une main avec une ou deux webcams (Wang *et al.*, 2011)

Enfin, différentes contributions ont eu recours à des caméras de profondeurs, parfois combinées avec une ou deux caméras RGB ou encore des caméras stéréoscopiques, pour faciliter le processus de détection des mains de l'utilisateur. En effet, avec des dispositifs tels que Microsoft Kinect [4] ou Intel RealSense [5], les chercheurs sont en mesure d'obtenir une représentation de la profondeur en plus d'un flux vidéo.

Van den Bergh et Van Gool évaluent la détection et la reconnaissance de gestes par flux vidéo et par données de profondeur dans (Van den Bergh et Van Gool, 2011). Pour ce faire, ils ont implémenté un système de détection et de reconnaissance fonctionnant avec une caméra RGB combinée à une caméra de profondeur. Deux méthodes de détection des mains ont été évaluées : la segmentation de couleur de peau (*skin color-based*) ou à l'aide d'informations de profondeur (*depth-based*). L'évaluation est effectuée selon trois scénarii : une situation normale avec la main proche du visage, main partiellement devant le visage et enfin avec une personne en arrière-plan tentant de perturber la détection. Van den Bergh et Van Gool présentent les taux de détection correcte

4. https://developer.microsoft.com/en-us/windows/kinect
5. https://realsense.intel.com/

suivants :

Scénarii	Skin color-based	Depth-based
Proche du visage	92.0%	99.2%
Devant le visage	71.8%	100%
Perturbation d'un tiers	19.8%	98.8%

Bien que la segmentation de couleur de peau produise un taux de détection correcte de 92.0% dans une situation normale, il est fortement impacté lorsque la main recouvre partiellement le visage ou que la peau d'un tiers se situe en arrière-plan de la main. Cet impact se justifie par le fonctionnement de la technique de segmentation utilisée : pour pallier aux soucis de variation de couleur de peau (luminosité, exposition, pigmentation...), la gamme de couleurs de cette dernière est déterminée par la détection du visage de l'utilisateur. À l'inverse, la détection *depth-based* produit de bons taux de détection correcte dans les trois scénarii. Par ailleurs, pour la reconnaissance de gestes appliquée une fois la main détectée, Van den Bergh et Van Gool ont recours à la méthode *Average Neighborhood Margin Maximization* (ANMM) pour réduire la main détectée à une image de plus petite dimension pour ensuite trouver son plus proche voisin dans une base de données de gestes. Cette technique est évaluée avec trois données d'entrée différentes : image RGB, image de profondeur et la combinaison des deux. Les résultats obtenus montrent un taux de reconnaissance correcte assez similaire pour les trois types avec respectivement 99.54%, 99.07% et 99.54%. Par conséquent, nous remarquons que l'utilisation de données de profondeur facilite grandement la détection des mains par rapport à la segmentation de couleur de peau, mais n'a pas d'impact significatif concernant la reconnaissance de geste.

En plus de faciliter la détection des mains et des gestes, la disponibilité des données de profondeur permet de mettre en oeuvre des interactions gestuelles répondant aux lois de la physique. Ce type d'interaction a été mis en oeuvre dans l'Holodesk (Hilliges *et al.*, 2012) : un système AR proposant un ensemble d'interactions gestuelles *physics-based* avec des éléments virtuels. Par exemple, Hilliges *et al.* illustre un utilisateur employant ses mains pour jongler avec deux balles virtuelles comme le montre la figure 2.14. Ce type d'interaction fut également présenté dans un système de traitement d'exposition AR pour l'arachnophobie (Corbett-Davies *et al.*, 2013) autorisant l'utilisateur à manipuler une araignée virtuelle à mains nues (cf. figure 2.15). L'Holodesk et ce dernier ont tous les deux recours à un Microsoft Kinect pour l'acquisition du flux vidéo et des données de profondeur.

FIGURE 2.14 – Holodesk : interaction gestuelle *physics-based* (Hilliges *et al.*, 2012)

Piumsomboon *et al.* utiliseront également une caméra de profondeur pour le module de reconnaissance de gestes mis en oeuvre dans le système G-SIAR

FIGURE 2.15 – Manipulation d'araignées virtuelles en AR (Corbett-Davies *et al.*, 2013)

(Piumsomboon *et al.*, 2014) offrant les six degrés de liberté de manipulations bimanuelles à l'aide d'une seule caméra de profondeur. Les six degrés de liberté (6 *degree of freedom* ou 6DOF) représentent un ensemble de paramètres décrivant la translation et la rotation d'un objet dans un espace tridimensionnel à savoir selon trois axes pour la translation et trois pour la rotation. La reconnaissance de gestes au sein de G-SIAR est assurée par la librairie 3Gear Nimble SDK [6] se basant sur les travaux de 6D Hands (Wang *et al.*, 2011). Par conséquent, G-SIAR permet de manipuler des objets virtuels à l'aide d'interactions gestuelles à mains nues. Les manipulations disponibles sont le déplacement d'un objet (de la même façon qu'un objet physique : attraper-déplacer-déposer), l'agrandissement/rétrécissement d'un objet (attraper l'objet à deux mains et les éloigner/rapprocher), pousser un objet ou encore jeter un objet.

6. http://nimblevr.com/latest/doc/

2.4 Microsoft HoloLens

Microsoft HoloLens (Microsoft, d) est un casque de réalité augmentée permettant de visualiser et d'interagir avec des hologrammes au sein du monde réel dans lequel l'utilisateur se trouve. La cohérence des hologrammes au sein de l'environnement est assurée par un système de cartographie de l'environnement intégré au le casque. Contrairement à un casque de réalité virtuelle tel que Oculus Rift [7], l'HoloLens n'est pas un périphérique se connectant à une machine. Éxécutant une version dite « holographique » du système d'exploitation Windows 10, le casque est un système embarqué autonome. Annoncée fin 2015 par Microsoft, l'édition de développement est mise en vente depuis mars 2016. C'est cette dernière qui sera détaillée dans la suite de ce document.

Le casque prend la forme d'un HMD (voir figure 2.16) reposant sur un affichage « Optical see-through », les hologrammes étant superposés au monde réel à travers les verres du casque. Le casque repose sur la technologie Microsoft Kinect [8] (Hempel, 2015b), introduite en 2010 pour la console de jeux Xbox, tout en étendant et améliorant ses fonctionnalités.

FIGURE 2.16 – Casque HoloLens (Microsoft, d)

Dans cette section, nous détaillerons les différentes fonctionnalités offertes par le casque HoloLens ainsi que les spécifications techniques permettant de mettre en oeuvre ces dernières. Enfin, une brève description des technologies de développement d'applications holographiques pour l'HoloLens clôturera la section. Étant destiné à la vente, il existe très peu de références scientifiques sur l'HoloLens. Les informations concernant ce dernier étant principalement issues d'articles de presse et de communications officielles de Microsoft.

2.4.1 Fonctionnalités

Cette section vise à détailler l'ensemble des fonctionnalités du casque HoloLens. Tout d'abord, nous détaillerons la fonctionnalité de visualisation d'hologrammes. Cette dernière représente une des innovations majeures des HoloLens. Nous aborderons ensuite la capacité de cartographie de l'environnement du casque pour enfin décrire les différentes méthodes d'interactions proposées à l'utilisateur. Le casque n'étant pas connecté à un ordinateur, nous ne disposons pas de périphériques d'interactions tels qu'un clavier ou une souris. Par conséquent, le casque dispose de différentes méthodes d'interactions telles que le suivi du regard, la reconnaissance des gestes ou encore les commandes vocales.

7. https://www.oculus.com/rift/
8. https://developer.microsoft.com/en-us/windows/kinect

Par ailleurs, l'HoloLens supporte également la spatialisation du son. Dès lors, lorsqu'un hologramme émet un signal sonore, ce signal est traité pour simuler la spatialisation du son dans l'espace. Le casque étant en mesure de détecter l'environnement de l'utilisateur et d'y introduire des éléments virtuels, la cartographie de l'environnement est utilisée pour donner une présence physique au son et sa source d'émission. À titre d'exemple, l'intensité d'un son émis par un hologramme sera progressivement réduite lorsque l'utilisateur s'en éloigne et augmentée lorsqu'il s'en rapproche.

Enfin, le casque dispose d'une fonctionnalité de partage d'hologramme. Cette dernière permet de partager le même monde virtuel entre plusieurs casques HoloLens. Dès lors, les différents utilisateurs visualisent et interagissent avec les mêmes éléments virtuels. La spatialisation du son et le partage d'hologrammes ne sont pas détaillés ci-dessous, leur présentation se suffisant à elle-même.

2.4.1.1 Visualisation

Une fois le casque correctement installé sur la tête de l'utilisateur, ce dernier est en mesure de visualiser des hologrammes à travers le casque. Un hologramme représente tout élément virtuel affiché par l'HoloLens. Dans (Microsoft, 2018a), Microsoft définit deux types de vues : les vues 2D et les vues immersives.

La vue 2D est destinée aux applications traditionnelles disponibles sur le Windows Store. Ces dernières seront représentées comme une projection 2D dans l'environnement de l'utilisateur comme illustré par la figure 2.17. L'utilisateur est ensuite capable de positionner cette projection à l'emplacement de son choix, mais également de l'agrandir ou la rétrécir. Ce type de vue permet d'ouvrir différentes applications en même temps, l'utilisateur étant en mesure de les visualiser et d'interagir avec chacune d'elles.

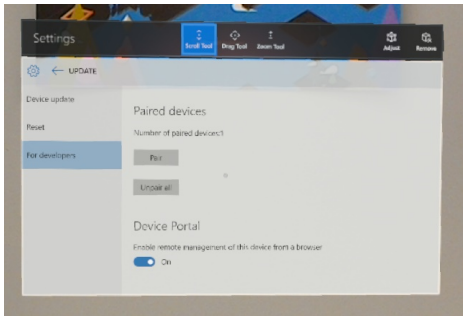

FIGURE 2.17 – Vue 2D (Microsoft, 2016)

Les vues immersives, quant à elles, permettent de visualiser des hologrammes dans l'environnement de l'utilisateur ou de l'immerger dans un monde virtuel. Cependant, lorsqu'une application s'exécute dans ce mode, aucune autre application ne peut être visualisée au même moment. La figure 2.18 illustre un utilisateur au sein d'une vue immersive. Notons également qu'une application proposant une vue immersive peut également proposer une ou plusieurs vues 2D et basculer de l'une à l'autre en fonction des besoins auxquels elle répond.

FIGURE 2.18 – Vue immersive (Microsoft, 2018a)

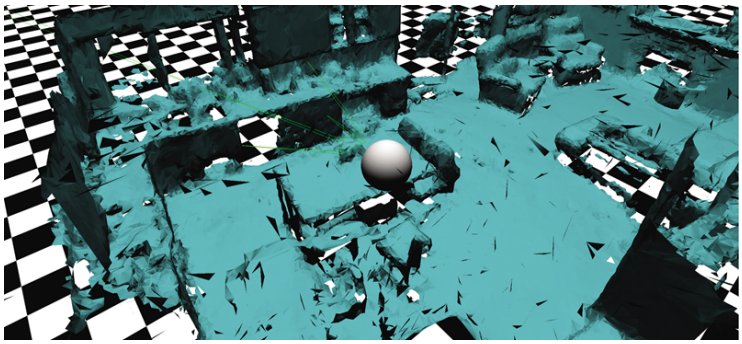

FIGURE 2.19 – Cartographie de l'environnement (Tuliper, 2017)

2.4.1.2 Cartographie de l'environnement

Comme annoncé précédemment, l'HoloLens dispose d'un système de cartographie (spatial mapping) lui permettant d'analyser l'environnement dans lequel se trouve l'utilisateur. Cette fonctionnalité joue un rôle crucial dans le bon fonctionnement du casque. D'une part, elle permet de renforcer la cohérence des hologrammes en les positionnant de façon adéquate aux yeux de l'utilisateur et d'une autre part, elle contribue au bon fonctionnement du système de visualisation du casque. La figure 2.19 illustre la représentation de l'environnement que le casque HoloLens est en mesure de construire.

En disposant d'une cartographie de l'environnement, les éléments virtuels peuvent être positionnés de façon cohérente. En effet, un hologramme flottant dans les airs, ou étant visible à travers un mur, tendra à renforcer la barrière existante entre les hologrammes et le monde réel. Par conséquent, en veillant à ce qu'un hologramme soit disposé ou mis à l'échelle adéquatement en fonction de l'espace disponible, la cohérence de cette réalité mixte sera plus grande et légitime pour l'utilisateur.

Par ailleurs, le spatial mapping assure un rôle clé dans la visualisation des hologrammes. Avec une représentation de l'environnement, le casque fournit les 6 degrés de liberté à l'utilisateur. L'HoloLens a connaissance de la position et de

l'orientation de l'utilisateur dans son environnement et peut donc en déduire son champ de vision et actualiser la visualisation des hologrammes en conséquence. Soit un utilisateur u et un hologramme h. Nous définissons p_i et $\vec{o_i}$, la position et l'orientation d'un élément i dans l'environnement. La position étant définie comme un triplet (x, y, z) et l'orientation comme un vecteur $< x, y, z >$ dans l'espace. Avec,

$$p_u = (0, 0, 0),$$
$$\vec{o_u} =< 1, 0, 0 >,$$
$$p_h = (-1, 0, 0).$$

L'utilisateur u se trouve au centre de la pièce et dirige son regard dans le sens opposé à la position de l'hologramme h. Par conséquent, h ne doit pas apparaitre dans le champ de vision de u.

2.4.1.3 Méthodes d'interactions

Comme détaillé précédemment, le casque est autonome et ne nécessite aucune connexion avec un ordinateur pour être utilisé. Dès lors, le couple clavier/-souris n'est pas disponible pour interagir avec le casque. Par conséquent, les méthodes d'interactions qu'il intègre sont diverses et reposent sur de multiples canaux d'entrées : les gestes, le regard et la reconnaissance vocale. Par ailleurs, ses canaux sont rarement utilisés indépendamment les uns des autres, ils sont fusionnés pour fournir une interaction plus riche à l'utilisateur.

Regard

L'HoloLens utilise le regard de l'utilisateur comme un dispositif de pointage permettant à l'utilisateur de signaler l'hologramme avec lequel il désire interagir. Il n'est pas question d'eye tracking pour ce canal : l'approche choisie par Microsoft est de considérer le regard de l'utilisateur comme le centre de son champ de vision. Ce dernier est matérialisé par un curseur permettant à l'utilisateur d'en connaitre la position. La figure 2.20 illustre l'affichage d'un curseur ainsi que la mise en surbrillance d'un hologramme lorsqu'il est pointé par ce dernier.

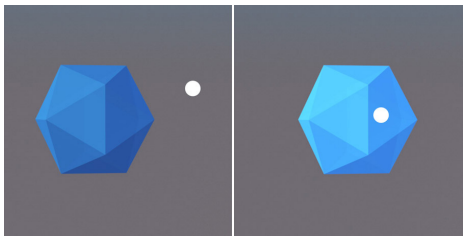

FIGURE 2.20 – Exemple de curseur HoloLens (Microsoft, b)

En matérialisant le regard par un curseur, ce canal d'interaction devient bidirectionnel. D'une part, le regard est utilisé par l'HoloLens pour déterminer ce sur quoi l'utilisateur porte son attention. D'autre part, en utilisant un curseur, le casque communique des informations à l'utilisateur : avec une rétroaction

adéquate du curseur, l'utilisateur pourra plus facilement comprendre avec quels éléments il peut interagir ou non. La figure 2.20 illustre une telle rétroaction.

Cette bidirectionnalité du canal d'interactions a mené Microsoft à définir des principes de conception concernant l'apparence, le comportement et les effets du curseur (Microsoft, a). Les développeurs d'applications holographiques sont invités à suivre ces concepts pour assurer une expérience utilisateur homogène et cohérente sur l'ensemble des applications de la plateforme.

Gestes

Le regard étant utilisé comme dispositif de pointage, les gestes permettent d'interagir avec les hologrammes visés. La combinaison du regard et de l'interaction gestuelle est définie par Microsoft comme l'interaction « gaze-and-commit » (Microsoft, c).

L'HoloLens détecte les mains de l'utilisateur et reconnait deux *core gestures* (Microsoft, c) : *Bloom* (figure 2.21) et *Air tap* (figure 2.22). Le premier est un geste réservé du système permettant uniquement de revenir au menu principal de l'HoloLens. Le second est utilisé comme un substitut du clic de souris. Comme illustré dans la figure 2.22, il s'effectue en deux temps : *ready* (n°1 sur la figure), lorsque l'index est levé, et *tap* (n°2 sur la figure) lorsque l'index rejoint le pouce puis retourne à la position *ready*.

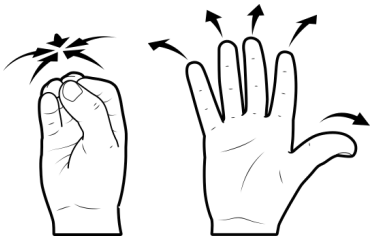

FIGURE 2.21 – Geste *Bloom* (Hempel, 2015a)

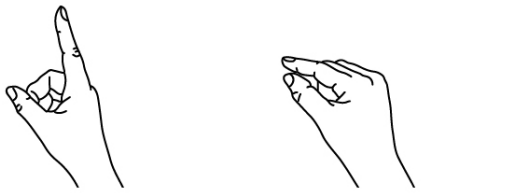

FIGURE 2.22 – Geste *Air tap* (Microsoft, c)

De plus, l'*Air tap* s'effectuant en deux temps, différents gestes composites en sont dérivés (Microsoft, c). Ces derniers sont listés ci-dessous :

Tap and hold consiste à maintenir l'index contre le pouce pendant l'exécution d'un *Air tap*

Navigation et Manipulation détectent la translation de la main de l'utilisateur lorsque ce dernier effectue un *Tap and hold*.

D'un point de vue physique, *Navigation* et *Manipulation* sont identiques dans leur exécution, il s'agit ici d'une distinction sémantique par rapport au contexte dans lequel ils sont réalisés. D'une part, le *Manipulation* s'applique aux manipulations d'hologrammes telles que le déplacement ou le redimensionnement. D'autre part, le *Navigation* s'applique typiquement au défilement horizontal ou vertical au sein d'un menu ou d'une page. Ce dernier s'applique donc majoritairement sur les hologrammes 2D tandis que le *Manipulation* s'applique à l'ensemble des hologrammes.

Reconnaissance vocale

Pour fournir une méthode d'interaction supplémentaire à l'utilisateur, le casque propose de la reconnaissance vocale. Comme pour les interactions gestuelles, la reconnaissance vocale est fusionnée avec le regard afin de mettre en oeuvre une seconde fois l'interaction « gaze-and-commit », détaillée précédemment. Pour ce faire, l'HoloLens est en mesure de reconnaitre des ordres vocaux qui seront, une fois prononcés, appliqués aux éléments visés par l'utilisateur, à l'aide de son regard.

La reconnaissance vocale fournit une redondance d'interactions à l'utilisateur. En effet, au sein du système Windows, la navigation peut être effectuée aussi bien avec les gestes qu'avec la reconnaissance vocale. Tout d'abord, un geste *Air tap* peut être simulé à tout moment avec la commande vocale *Select* (Microsoft, 2018p). Ensuite, afin d'éviter à l'utilisateur de devoir apprendre l'ensemble des commandes vocales prises en charge, Microsoft a mis en oeuvre le concept « See It, Say It » qui affiche l'ordre vocal à prononcer au sein de l'interface du système HoloLens (Microsoft, 2018o). Dès lors, l'utilisateur n'a pas à se souvenir de toutes les commandes vocales disponibles, car elles sont constamment mises en évidence pendant son utilisation. Enfin, des commandes de manipulation d'hologrammes sont reconnues par le casque pour effectuer des tâches rapides telles qu'agrandir ou rétrécir une application 2D ou des hologrammes placés dans l'environnement (Microsoft, 2018q).

En plus des commandes vocales, l'assistant personnel Cortana [9] est intégré dans le casque. Ce dernier est agrémenté de commandes vocales spécifiques à l'HoloLens permettant entre autres d'éteindre le casque, prendre une capture d'écran, diminuer le volume, lancer une application ou encore de retourner au menu principal (Microsoft, 2018m). Notons que cette dernière commande est équivalente à l'exécution du geste *Bloom*.

Par ailleurs, la saisie de texte par l'utilisateur est facilitée par un système de dictée. Bien qu'un clavier virtuel soit disponible et utilisable avec le regard et des *Air taps*, la saisie de texte en mode dictée est bien plus rapide et facile pour l'utilisateur (Microsoft, 2018l). Pour ce faire, le clavier virtuel propose un bouton permettant d'activer le mode dictée comme l'illustre la figure 2.23.

9. https://www.microsoft.com/en-us/windows/cortana

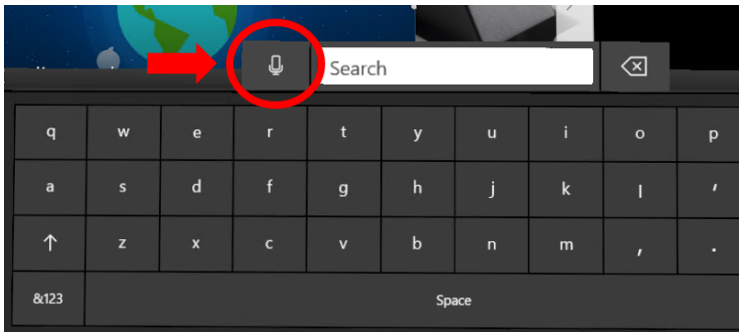

FIGURE 2.23 – Mode dictée sur le clavier virtuel (Microsoft, 2018l)

2.4.2 Spécifications

L'HoloLens étant un système autonome géré par le système d'exploitation Windows 10 holographic, ce dernier est muni des composants constituant n'importe quel système informatique autonome. Ainsi, le casque dispose d'un processeur Intel Cherry Trail en architecture 32 bit, de 2Go de mémoire vive LPDDR3 [10] et 64Go de mémoire flash dédiée au stockage (Microsoft, 2018g). Le tout étant alimenté par une batterie spécialement conçue par Microsoft et pouvant fournir l'énergie nécessaire au casque pendant une durée de deux à trois heures. Concernant la connectique du casque, nous retrouvons un port micro-USB destiné au chargement de la batterie ainsi qu'au déploiement d'applications en cours de développement, une sortie audio jack 3.5mm ainsi que la connectivité Bluetooth et Wi-Fi.

Par ailleurs, un système de refroidissement passif (sans ventilateur) maintient les composants à une température de fonctionnement adéquate. Le choix d'un tel système n'est pas sans conséquence : en cas de dépassement de ses capacités thermiques, l'HoloLens terminera immédiatement l'exécution de l'application en cours afin de retrouver une température de fonctionnement appropriée pour éviter une dégradation matérielle du casque. Pour prévenir un tel scénario, Microsoft fournit une liste de recommandations de développement destinées à augmenter la performance des applications (Microsoft, 2018h).

2.4.2.1 Composants et architecture matérielle

L'ensemble des fonctionnalités du casque sont rendues possibles grâce aux nombreux capteurs et composants électroniques (Microsoft, 2018g) intégrés dans ce dernier. L'ensemble des fonctionnalités présentées reposent sur des données environnementales qui sont récupérées par les capteurs listés ci-dessous :

- 1 capteur de luminosité ambiante ;
- 4 microphones ;
- 1 caméra de profondeur (*depth camera*) ;
- 1 caméra photo 1.2MP / vidéo HD ;
- 4 caméras de compréhension de l'environnement ;
- 1 *Inertial Measurement Unit* (IMU).

10. Low Power DDR3 : mémoire DDR3 à faible consommation électrique

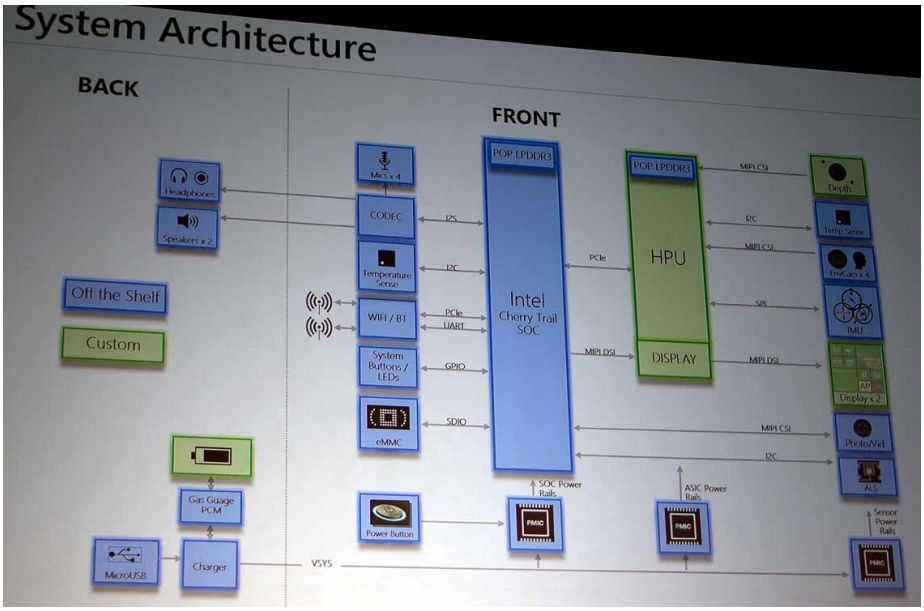

FIGURE 2.24 – Architecture système de l'HoloLens (Colaner, 2016)

Les rôles respectifs de ces capteurs ne sont pas communiqués en détails par Microsoft pour des raisons commerciales évidentes. Cependant, une présentation de Microsoft, rapportée dans (Colaner, 2016), nous apprend néanmoins que certains éléments ont été conçus spécifiquement pour l'HoloLens tandis que d'autres sont des composants off-the-shelf intégrés dans l'architecture. La caméra de profondeur a été créée pour le casque HoloLens alors que les caméras de compréhension de l'environnement sont des éléments off-the-shelf. Ces dernières fournissent des informations de bases pour le suivi du regard en récupérant une position et une orientation de l'utilisateur dans l'espace. D'autre part, la caméra de profondeur facilite le suivi des mains de l'utilisateur et permet la reconstitution de son environnement.

De plus, pour fournir un suivi efficace du mouvement de tête de l'utilisateur, le casque est équipé d'un IMU (Microsoft, 2018g). Il s'agit d'un composant électronique généralement constitué d'un gyroscope, un accéléromètre et un magnétomètre. Il permet de récupérer les mouvements d'un équipement, à savoir, son accélération et sa vitesse angulaire. Associé avec les caméras de compréhension de l'environnement, il permet au casque d'assurer un suivi précis du mouvement de l'utilisateur au sein de l'environnement et par conséquent, de pouvoir ancrer les hologrammes à une position fixe dans ce dernier (Colaner, 2016). La combinaison des données, obtenues par les différents capteurs, permet au casque de déterminer les valeurs de position p_i et d'orientation $\vec{o}_i$ définies dans la section Cartographie de l'environnement (2.4.1.2).

L'organisation de tous ces éléments est illustrée par la figure 2.24 détaillant l'architecture matérielle du casque. Nous y retrouvons les composants conçus spécifiquement pour le casque en vert, et les éléments off-the-shelf en bleu. Ce schéma permet également de comprendre la communication existante entre les différents éléments constituant l'HoloLens. Nous pouvons constater la pré-

sence d'un intermédiaire entre le processeur Intel et la majorité des capteurs : l'*Holographic Processing Unit* (HPU). Ce dernier est responsable de l'agrégation des données obtenues depuis les différents capteurs, tout en effectuant l'ensemble des traitements nécessaires à la détection de l'environnement et des gestes de l'utilisateur. Les résultats de son traitement étant ensuite transmis au processeur Intel. L'HPU est également responsable du système de visualisation de l'HoloLens. La mémoire vive du casque est donc répartie entre ses deux unités de calculs : 1Go pour l'Intel Cherry Trail et 1Go pour l'HPU (Colaner, 2016).

L'utilisation d'un composant matériel dédié à l'ensemble de ces tâches permet l'accélération des algorithmes de traitements jusqu'à 200 fois par rapport à une implémentation logicielle, tout en assurant une consommation énergétique réduite. Dès lors, seuls la caméra photo/vidéo et le capteur de luminosité ambiante sont directement accessibles au processeur Intel. (Colaner, 2016).

2.4.3 Développement

Cette section vise à détailler les différentes technologies relatives au développement d'applications HoloLens. Avec le regard, les gestes, la parole, la cartographie de l'environnement et l'ancrage d'hologrammes au sein du monde réel, de nombreux éléments nouveaux entrent en ligne de compte pour le développement d'applications de réalité augmentée contrairement aux applications graphiques traditionnelles (simulation 3D, jeux vidéo...). En effet, les principaux canaux d'entrée de l'utilisateur restent le couple clavier/souris ou encore la manette de jeu dans la majorité de ces applications. Dépendant du type d'application à réaliser, nous disposons de trois technologies de développement : l'API *Universal Windows Platform* [11] (UWP), le moteur de jeux Unity [12] et l'API graphique DirectX [13] (Microsoft, 2018b).

2.4.3.1 Universal Windows Platform

L'UWP est une API de Microsoft facilitant le développement d'applications destinées aux systèmes Windows. Dès lors, une application développée avec UWP sera exécutable, sans modification, aussi bien sur un ordinateur que sur un smartphone, une Xbox One [14] ou l'HoloLens. Sur ce dernier, les applications UWP seront matérialisées par une projection en deux dimensions au sein de l'environnement (Microsoft, 2018b). Ce type de projection fut présenté dans la section 2.4.1.1 Visualisation.

Cette API ne permet cependant pas de profiter des nouvelles méthodes d'interactions du casque HoloLens. En effet, l'aspect multi-plateforme de cette API introduit une couche d'abstraction supplémentaire : les canaux d'entrée sont représentés sous la forme d'évènements indépendants des entrées réelles. Par conséquent, l'API UWP s'occupe de la complexité de gestion des différentes entrées pouvant exister selon la plateforme sur laquelle l'application s'exécute. Pour l'HoloLens, le regard sera converti en évènements de survol (*hover event*) et un geste *air tap* ou la commande vocale *Select* deviendra un évènement de pression e.g. l'appui d'un bouton (Microsoft, 2018k).

11. https://docs.microsoft.com/en-us/windows/uwp/index
12. https://unity3d.com
13. https://msdn.microsoft.com/en-us/library/windows/desktop/ee663274(v=vs.85).aspx
14. https://www.xbox.com

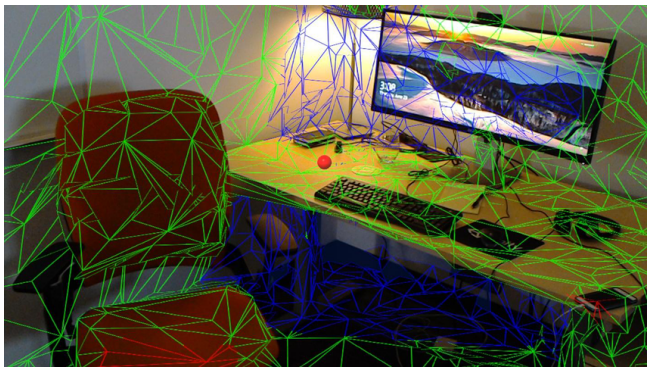

FIGURE 2.25 – Affichage de la cartographie spatiale avec Unity (Unity, 2017c)

2.4.3.2 Unity

Unity est un moteur de développement de jeux vidéo proposant une collection d'outils et de fonctionnalités facilitant le développement d'applications graphiques. Parmi ces fonctionnalités, nous pouvons citer la gestion et l'importation d'assets (images, modèles 3D, textures, sons...), un moteur physique, un système de gestion personnalisable des entrées de l'utilisateur ou encore un module de création d'UI [15]. Par conséquent, Unity apporte un niveau d'abstraction pouvant grandement faciliter le développement d'application et Microsoft recommande son utilisation pour le développement d'applications HoloLens (Microsoft, 2018b). Nativement supporté par Unity, l'ensemble des méthodes d'interaction du casque sont accessibles à l'aide de la Scripting API d'Unity (Microsoft, 2018j).

Tout d'abord, le regard est automatiquement lié à la caméra principale de l'application. Par conséquent, les hologrammes visés par le regard de l'utilisateur sont aisément identifiables (Microsoft, 2018e) et le mouvement de l'utilisateur au sein de son environnement est automatiquement traduit en déplacement de caméra au sein de l'application (Microsoft, 2018j). Concernant la détection des gestes, les développeurs sont en mesure de définir des actions qui seront exécutées lors de la détection des différents gestes reconnaissables par le casque (Microsoft, 2018f). Enfin, il est également possible de profiter du mode dictée, mais aussi de définir des commandes vocales qui seront reconnues par l'application et qui enclencheront une ou plusieurs actions, définies par les développeurs (Microsoft, 2018n).

Par ailleurs, les développeurs peuvent également contrôler le système de cartographie de l'environnement et ainsi accéder à une collection de surfaces (Microsoft, 2018i) représentant l'environnement de l'utilisateur selon un degré de précision paramétrable. Unity permet nativement l'affichage (cf. figure 2.25) et la création de boîtes de collision des données récoltées (Unity, 2017c).

Finalement, en complément de la *Scripting API* d'Unity, Microsoft propose la librairie *MixedRealityToolkit-Unity* (MRTK), une collection de fonctionnalités reposant sur l'API d'Unity pour en faciliter son utilisation. MRTK dispose d'un ensemble de scripts, modèles 3D, sons, shaders, etc. que les développeurs sont libres d'utiliser pour développer des applications destinées à l'HoloLens. Avec

15. User Interface

MRTK, les concepteurs d'applications disposent entre autres de curseurs prêts à être utilisés, d'une intégration des gestes au sein du système de gestion des entrées d'Unity ou encore d'une couche d'abstraction de la cartographie spatiale : *Spatial Understanding*. Cette couche permet de traiter les données environnementales récupérées pour en déduire des zones d'intérêt telles que les murs, le sol, des surfaces sur lesquelles l'utilisateur peut s'assoir ou encore les tables. Les développeurs pouvant ensuite effectuer des requêtes de placement (e.g. trouver une zone d'un m^2 sur un mur)

2.4.3.3 DirectX

Pour laisser plus de flexibilité aux développeurs, l'API graphique DirectX permet de concevoir des applications destinées au casque HoloLens. Cette solution permet un contrôle total sur le comportement de l'application au détriment du niveau d'abstraction offert par Unity. En effet, le développement d'applications holographiques avec DirectX revient à développer un moteur graphique à part entière (Microsoft, 2018b,d).

Cependant, DirectX fournit une API accédant à l'ensemble des fonctionnalités offertes par le casque. Bien que l'interface de programmation soit semblable aux composants fournis par la *Scripting API* d'Unity, c'est l'intégration de ces éléments avec le reste de l'application qui est entièrement sous le contrôle des développeurs (Microsoft, 2018c). À titre d'exemple, la liaison entre le regard de l'utilisateur et la caméra de l'application est ici inexistante, le concept même de caméra devant être défini.

En résumé, l'utilisation d'une telle technologie de développement est destinée à des applications très spécifiques qui requièrent une flexibilité importante dans leur implémentation. Unity embarque une grande quantité de fonctionnalités pouvant être inadaptées voire superflues. Par conséquent, certains développements vont nécessiter une application plus légère proposant uniquement les éléments utiles à son fonctionnement. Les développeurs sont alors responsables de l'ensemble de l'implémentation avec DirectX.

Chapitre 3

Extensions des interactions gestuelles HoloLens

3.1 Contexte du travail

Cette section a pour objectif de présenter le contexte dans lequel s'inscrit la recherche présentée dans ce document. Comme mentionné précédemment, la recherche est effectuée au sein du projet EFFaTA-MeM désirant concevoir un ensemble d'outils destinés à l'analyse et l'interprétation de texte par une approche transdisciplinaire, réunissant les sciences humaines et les sciences informatiques. Les prototypes AR résultant de la recherche présentée dans ce document sont destinés à la visualisation d'informations intervenant dans l'analyse structurale, une technique d'analyse de texte sur laquelle EFFaTA-MeM a réalisé de nombreuses recherches. Dès lors, la section ci-dessous présente une vue d'ensemble de cette technique d'analyse définie dans (Piret *et al.*, 1996).

3.1.1 Analyse structurale

Piret *et al.* définissent l'analyse structurale comme une méthode d'analyse de contenu sémantique et structurale. D'une part, le caractère sémantique de cette technique est validé par l'intérêt porté sur le sens du discours pour y comprendre ce que le locuteur veut effectivement exprimer. D'autre part, cette méthode est qualifiée de structurale, car elle a pour objectif d'identifier les relations (associations ou oppositions) liant les différents axes sémantiques incarnés dans le discours. Il n'est pas question ici d'ordre ou de fréquence d'apparition, ce sont bien des relations entre les éléments du texte qui constituent l'unité d'analyse utilisée pour découvrir la signification du discours.

3.1.1.1 Relation de disjonction

L'unité d'analyse étant les relations entre les éléments du texte, l'analyse structurale repose sur un postulat de binarité modélisant une relation dite de disjonction entre deux termes. Piret *et al.* précisent de cette relation représente « deux mots (ou deux groupes de mots) qui ont quelque chose en commun tout en étant différents ». L'élément commun entre les deux termes portant le

nom d'axe sémantique. Plus formellement, nous pouvons définir une relation de disjonction d comme un couple $(a_d, b_d)^{x_d}$ où a_d, b_d sont les deux termes de d et x_d est l'axe sémantique de d. La figure 3.1 illustre une relation de disjonction entre le terme « superflu » et le terme « nécessaire », leur dénominateur commun, l'axe sémantique, étant « les achats ».

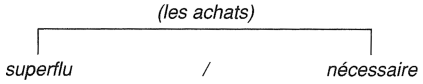

FIGURE 3.1 – Exemple de relation de disjonction (Piret *et al.*, 1996)

Par ailleurs, Piret *et al.* décrivent le concept de valorisation : lorsqu'un locuteur introduit une relation de disjonction au sein de son discours, il aura souvent tendance à connoter positivement un terme, et négativement son inverse. Cette connotation sera représentée par un indice de valorisation « + » ou « - » en dessous des termes connotés.

Sur base de l'unité d'analyse que représente la relation de disjonction, Piret *et al.* décrivent des structures dont l'objectif est de schématiser la manière dont les relations de disjonction se combinent entre elles. Nous y retrouvons la structure parallèle, la structure hiérarchique et enfin, la structure croisée.

3.1.1.2 Structure parallèle

Lorsqu'une relation de double implication existe entre les termes de deux ou plusieurs relations de disjonction, nous nous retrouvons face à une structure parallèle. La figure 3.2 illustre une telle structure. Dès lors soit les relations de disjonction d et e, il existe une structure parallèle entre d et e si $(a_d \Leftrightarrow a_e) \wedge (b_d \Leftrightarrow b_e)$.

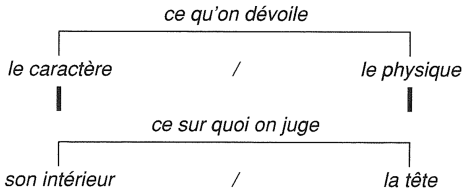

FIGURE 3.2 – Structure parallèle entre deux disjonctions (Piret *et al.*, 1996)

3.1.1.3 Structure hiérarchique

La structure hiérarchique exprime le double statut d'un terme qui se retrouve en tant qu'inverse dans une relation de disjonction et axe sémantique d'une autre. Cette dualité forme une hiérarchie plus ou moins profonde selon les disjonctions en présence. Ainsi, comme l'illustre la figure 3.3, une structure hiérarchique existe entre deux disjonctions d et e lorsque $(a_d = x_e) \oplus (b_d = x_e)$, $\oplus$ représentant l'opérateur de disjonction exclusive (XOR). Notons également que cette structure peut être combinée avec des structures parallèles si nécessaire.

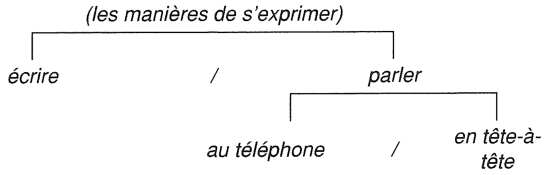

FIGURE 3.3 – Structure hiérarchique entre deux disjonctions (Piret *et al.*, 1996)

3.1.1.4 Structure croisée

Selon les disjonctions introduites par le locuteur au sein du discours, les structures parallèles et hiérarchiques ne permettent pas d'analyser ou décrire entièrement les combinaisons de termes mises en oeuvre par le locuteur. Par conséquent, Piret *et al.* définissent la structure croisée. Cette dernière associe deux relations de disjonction dont chacun des termes représente une réalité appelée réalité mère. Cette combinaison, illustrée par la figure 3.4, donne naissance à quatre quadrants symbolisant chacun une réalité théorique, dite fécondée, caractérisée par les deux réalités mères de son quadrant. Les deux disjonctions étant le siège d'une structure croisée sont nommées disjonctions mères ou axes mères.

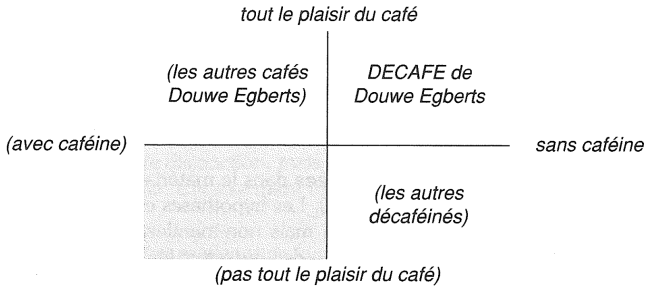

FIGURE 3.4 – Structure croisée entre deux disjonctions (Piret *et al.*, 1996)

3.2 Prototypes de visualisations en AR

L'élaboration de visualisations en réalité augmentée avec le casque HoloLens a mené à la création de deux prototypes qui sont présentés dans cette section. Ces visualisations ont pour objectif de représenter les différentes relations de disjonctions identifiées durant le processus d'analyse structurale d'un texte. Dès lors, les membres du projet EFFaTA-MeM désirent proposer une visualisation reposant sur la métaphore d'une cathédrale de texte que l'analyste construit au fil de son analyse. L'idée étant d'immerger l'analyste au sein des relations qu'il identifie et lui permettre de se mouvoir au sein d'une structure architecturale modélisant les relations présentes dans le texte. L'analogie entre l'écartement produit par une voûte sur les deux piliers qu'elle lie et une relation de disjonction représentant le point central de la métaphore de cathédrale de texte. Durant l'exploration des fonctionnalités offertes par l'HoloLens, un premier prototype de visualisation à été développé selon cette métaphore. Cependant, la struc-

ture illustrée ne présentait pas une plus-value significative par rapport à une visualisation en deux dimensions et son lien fort avec l'architecture avait pour effet de rendre la visualisation peu interactive. Ce constat a donc conduit à la conception d'un second prototype représentant les relations de disjonction, et les structures qu'elles sont susceptibles de former, comme un mobile artistique pouvant mettre à profit la 3D.

3.2.1 Cathédrales de texte

La visualisation de relations de disjonction sous la forme d'une cathédrale de texte est matérialisée par le prototype *CathedralBuilder* proposant la construction de structures constituées de piliers et de voûtes de cathédrales. Cette visualisation fait l'analogie entre une relation de disjonction et deux piliers de cathédrale liés par une voûte comme le montre la figure 3.5(a). Dès lors, les deux termes constituant la relation sont incarnés par les piliers et la voûte liant ces derniers représente l'axe sémantique sur lequel la relation de disjonction est établie.

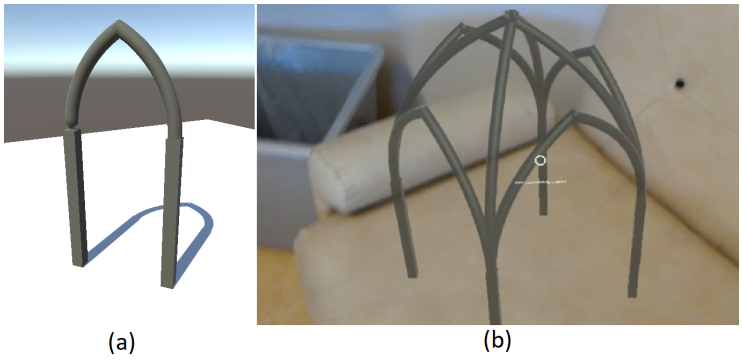

FIGURE 3.5 – (a) Représentation d'une relation de disjonction (b) Visualisation à échelle réduite au sein de l'environnement

CathedralBuilder permet à un utilisateur de construire des piliers et des voûtes au sein de son environnement. En effet, à son lancement, l'application va en effectuer la cartographie pour en détecter les surfaces horizontales. Elle permettra ensuite la création de piliers et de voûtes soit à taille réelle, avec des piliers d'approximativement $2,5m$ de haut, soit à la dimension d'une maquette avec une hauteur de $30cm$. La création de piliers est effectuée à l'aide d'un *air tap* sur une surface horizontale et la taille du pilier sera déterminée par le type de surface sur laquelle le pilier doit être créé : un pilier créé sur le sol sera de taille réelle tandis qu'il sera de taille restreinte sur toute autre surface horizontale (table, assise d'une chaise...). La figure 3.5(b) illustre la visualisation de piliers et de voûtes créés sur l'assise d'un fauteuil, identifiée comme une surface horizontale n'étant pas le sol.

De plus, lorsque *CathedralBuilder* clôture le processus de cartographie de l'environnement, une barre d'outils est accessible à l'utilisateur (cf. figure 3.6) et suivra ses déplacements. Cette dernière permet à l'utilisateur de sélectionner l'interaction qu'il souhaite réaliser au sein de l'application. La liste des fonctionnalités de *CathedralBuilder* est détaillée ci-dessous.

— Création de piliers
— Création de voûtes
— Suppression de piliers
— Déplacement de piliers
— Réinitialisation de la scène
— Affichage de la cartographie spatiale

FIGURE 3.6 – Barre d'outils de *CathedralBuilder*

Le mode de création de voûtes permet à l'utilisateur de sélectionner deux piliers avec un *air tap* pour générer une voûte entre ces derniers. Il n'est évidemment pas permis d'en créer en sélectionnant deux fois le même pilier ou entre deux piliers de taille différente. En sélectionnant le mode de suppression, il est possible de supprimer les piliers sur lesquels un *air tap* est effectué. Dès lors, les éventuelles voûtes y étant liées seront effacées étant donné la binarité d'une relation de disjonction. Le mode de déplacement permet quant à lui d'effectuer un geste *manipulation* sur un pilier pour appliquer la translation, effectuée par la main de l'utilisateur, au pilier visé. En déplaçant un pilier, les voûtes qui y sont liées sont automatiquement redimensionnées pour s'adapter aux nouvelles coordonnées du pilier. Enfin, l'utilisateur peut également réinitialiser la scène et visualiser la cartographie de l'environnement comme illustré par la figure 2.25 page 31.

Bien que *CathedralBuilder* ait permis d'explorer les fonctionnalités proposées par le casque HoloLens, la métaphore incarnée dans *CathedralBuilder* ne permet pas d'exprimer les concepts plus avancés de l'analyse structurale. En effet, les différentes structures pouvant être formées par les relations de disjonction (parallèles, hiérarchiques et croisées) ne sont pas aisément applicables à cette visualisation. De plus, cette représentation n'exploite pas la dimension supplémentaire apportée par la réalité augmentée, les piliers et les voûtes peuvent être directement comparés aux noeuds et liens des diagrammes Node-Link. Par ailleurs, l'utilisation d'une barre d'outils pour contrôler les modes d'interactions a pour effet d'introduire de la friction dans l'exécution des tâches bien qu'elle permette d'utiliser une même gestuelle pour des actions différentes.

L'ensemble de ces inconvénients ont motivé la conception d'une visualisation de cathédrales de texte plus abstraites, plus éloignées du sens architectural, pouvant mieux exploiter les trois dimensions, proposant une représentation des différentes structures de l'analyse structurale et offrant un degré d'interaction plus important avec les éléments présents tout en permettant d'intégrer de futurs concepts d'analyse de texte. Dès lors, un second prototype est réalisé en présentant une visualisation de mobiles artistiques détaillée dans la section suivante.

3.2.2 Mobile

Comme mentionné précédemment, la métaphore ayant guidée la réalisation de ce deuxième prototype est le mobile artistique. Il s'agit d'une sculture consti-

tuée d'éléments légers dont la configuration est notamment influencée par l'équilibre des masses. Sa disposition est changeante selon les forces pouvant être appliquées sur ses éléments (vent, moteur, touché...). La figure 3.7 illustre un mobile artistique réalisé par Alexander Calder. En plus des motivations précédemment évoquées, EFFaTA-MeM projette d'intégrer des concepts d'analyse de texte supplémentaires pour fournir des outils additionnels aux analystes. Dès lors, la visualisation de mobile artistique est mise en oeuvre de façon à proposer une configuration changeante des éléments la constituant. En effet, en plus de proposer une modélisation des relations de disjonction, les liens de synonymie et de distance sémantique entre les mots représentent un paradigme d'analyse sur lequel le projet EFFaTA-MeM porte son attention. Un tel paradigme pourrait incarner une force influençant la configuration ou la disposition changeante des éléments du mobile.

FIGURE 3.7 – Mobile artistique réalisé par Alexander Calder (Tate, 2008)

3.2.2.1 Représentation des relations de disjonction

Avec cette métaphore, une relation de disjonction est représentée par deux éléments d'un mobile liés par un fil de fer, symbolisant respectivement les deux termes et l'axe sémantique de la relation, ce qui autorise la représentation des différentes structures définies par la technique d'analyse structurale en plus d'exploiter les trois dimensions disponibles. La figure 3.8 illustre la visualisation d'une relation de disjonction entre les termes « pure » et « impure » sous la forme d'un mobile artistique. Les deux termes sont incarnés par les deux sphères tandis que l'axe sémantique matérialisant la relation de disjonction est modélisé par la courbe les reliant.

FIGURE 3.8 – Relation de disjonction sous la forme d'un mobile

3.2.2.2 Structures de l'analyse structurale

Pour les structures parallèles, les relations de disjonction concernées sont représentées à la même hauteur, avec un point de connexion entre les courbes de relations. La figure 3.9 représente la structure parallèle donnée en exemple à la figure 3.2.

FIGURE 3.9 – Mobile d'une structure parallèle

Concernant les structures hiérarchiques, il existe une analogie forte entre la hiérarchie des relations de disjonction et la structure même d'un mobile artistique. Dès lors, le terme portant le rôle d'axe sémantique d'une seconde relation devient le point d'ancrage de cette dernière comme le montre la figure 3.10 reprenant l'exemple de la figure 3.3.

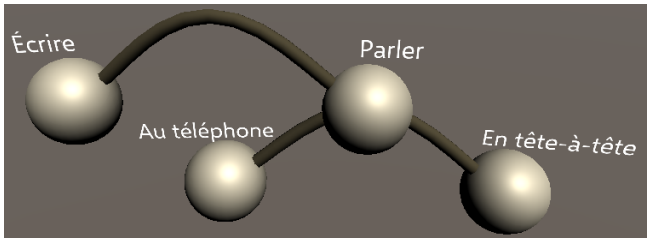

FIGURE 3.10 – Mobile d'une structure hiérarchique

Enfin, la figure 3.11 illustre la représentation d'une structure croisée sous la forme d'un mobile artistique. Pour cette structure, les termes sont positionnés sur le même plan horizontal de façon similaire à la structure parallèle. Le croisement des deux relations de disjonction est directement représenté par l'intersection perpendiculaire des deux courbes du mobile.

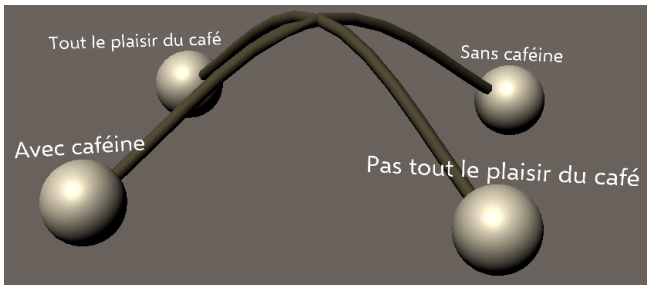

FIGURE 3.11 – Mobile d'une structure croisée

3.2.2.3 Configuration d'un mobile

Pour faciliter la représentation des différentes structures de l'analyse structurale présentées précédemment, la visualisation de mobiles est organisée comme un arbre. Cette organisation en arbre est facilitée par la hiérarchisation des éléments au sein du moteur Unity. En effet, au sein d'une scène, chaque entité est en mesure de contenir une hiérarchie de sous éléments. Par conséquent, de nombreuses méthodes de la Scripting API d'Unity permettent de gérer les sous-entités, mais également les entités parentes. Deux types d'entités sont définies : *MobileSolid* et *MobileLine*. Le premier représente les objets solides d'un mobile tandis que le second caractérise un fil et les deux objets qu'il lie. Par conséquent, un *MobileLine* représente une relation de disjonction comme illustrée précédemment par la figure 3.8.

L'organisation des différents éléments d'une relation de disjonction est mise en oeuvre par un ensemble de caractéristiques paramétrables telles que la distance entre chaque terme, la translation verticale des termes par rapport au sommet de la courbe ou encore l'orientation de la structure. Plus formellement, un *MobileLine* ml est défini comme un 7-uplet $(p, l, r, \vec{h}, d_l, d_r, \vec{o})$ où

— p est la position de ml dans l'espace ;
— l (resp. r) est le *MobileSolid* de gauche (resp. de droite) ;
— $\vec{h}$ représente le vecteur de translation vertical de l et r ;
— d_l (resp. d_r) correspond à la distance d'écartement de l (resp. de r) ;
— $\vec{o}$ est le vecteur d'orientation de ml.

Ces différents paramètres définissent l'apparence et la structure d'un *MobileLine* comme le montre la figure 3.12. Le vecteur $\vec{h}$ configure l'abaissement de l et r par rapport à la position p de ml tandis que l'écartement des l et r est défini par d_l et d_r.

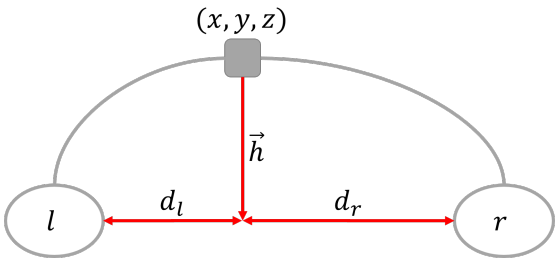

FIGURE 3.12 – Configuration d'un *MobileLine* (vue 2D)

Par ailleurs, le vecteur $\vec{o}$ définit l'orientation du mobile (non représenté dans la figure 3.12) et influence donc le positionnement de l et r comme illustré par la ligne verte dans les figures 3.13(a) et (b). De plus, $\vec{h}$ étant un vecteur dans l'espace, en plus de définir l'abaissement de l et r, il peut être utilisé pour définir une translation sur les autres axes comme le montre la ligne cyan dans la figure 3.13(b). Par conséquent, la position de l est obtenue avec :

$$(p + \vec{h}) + (\widehat{(\vec{o} \times \vec{h})} * d_l)$$

et la position de r est calculable avec :

$$(p + \vec{h}) - (\widehat{(\vec{o} \times \vec{h})} * d_r)$$

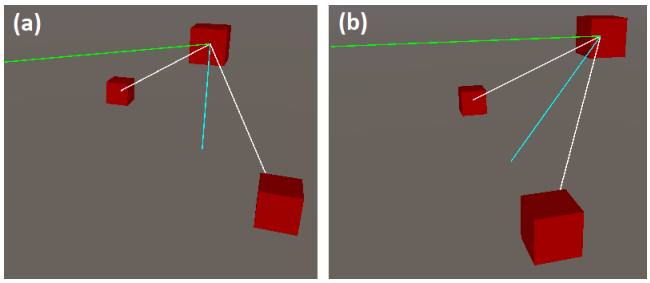

FIGURE 3.13 – (a) Configuration schématique d'un *MobileLine*. (b) Translation sur l'axe des x produite par $\vec{h}$. Les cubes rouges identifient les trois entités du mobile : un *MobileLine* et ses deux *MobileSolid*. Les tracés verts et cyans correspondent respectivement aux vecteurs $\vec{o}$ et $\vec{h}$.

3.2.3 Contributions techniques

3.2.3.1 Système de fonctionnalités

Étant donné la disponibilité d'une architecture Entity-Component au sein d'Unity, un système d'encapsulation de fonctionnalités a été implémenté pour faciliter le développement et l'évolution des prototypes. Pour ce faire, deux types de fonctionnalités sont définis :

Fonctionnalité à actions immédiates Une fois activées, un ensemble d'actions sont exécutées au sein de l'application, sans intervention de l'utilisateur. Une fois exécutée, la fonctionnalité se désactive d'elle-même. Sauvegarder l'état de l'application ou remettre à zéro l'espace de travail sont des exemples de fonctionnalité à actions immédiates.

Fonctionnalité complexe Lorsqu'une fonctionnalité nécessite une ou plusieurs interactions avec l'utilisateur, elle est décrite comme complexe. Un tel type de fonctionnalité est comparable à l'activation d'un « mode » offrant une fonctionnalité clairement définie (mode sélection, mode déformation, mode suppression...). Par exemple, avec l'activation d'un mode de suppression, l'utilisateur devient responsable de la sélection des éléments qu'il désire supprimer. Il y a donc une série d'interactions avec l'utilisateur qui doit être gérée par la fonctionnalité active.

La mise en oeuvre de ces deux types de fonctionnalités est facilitée par l'introduction du concept d'éléments activables, modélisant des éléments définis par leur capacité à s'initialiser, s'activer et se désactiver. Par conséquent, la figure 3.14 illustre la machine à états d'un élément activable.

Dès lors, les fonctionnalités à actions immédiates et les fonctionnalités complexes peuvent être définies comme des éléments activables. D'une part, une fonctionnalité à action immédiate peut être modélisée comme un élément activable dont l'activation implique l'exécution d'un processus prédéfini, dont l'étape finale aura pour effet de la désactiver. D'autre part, une fonctionnalité complexe est un élément activable dont l'état d'activation comporte un ensemble d'états internes contrôlant les interactions requises à son bon fonctionnement. L'orchestration des éléments activables peut ensuite être gérée à l'aide d'un gestionnaire central spécifique à l'application et responsable de l'activation/désactivation des fonctionnalités.

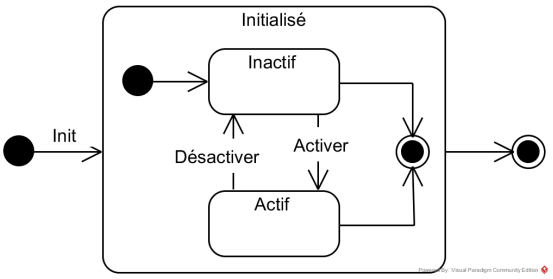

FIGURE 3.14 – Machine à états d'un élément activable

Concernant les fonctionnalités complexes, le concept d'élément activable est étendu pour faciliter leur gestion. Les interactions avec l'utilisateur sont gérées par un ou plusieurs composants d'interaction dont un gestionnaire de la fonctionnalité complexe sera responsable. Par conséquent, une fonctionnalité complexe est matérialisée par un gestionnaire de fonctionnalité (un élément activable) responsable de la bonne exécution de la machine à état qu'il modélise et gérant les composants d'interactions relatifs à la fonctionnalité. L'ensemble des concepts définis sont modélisés par le diagramme de classe illustré à la figure 3.15.

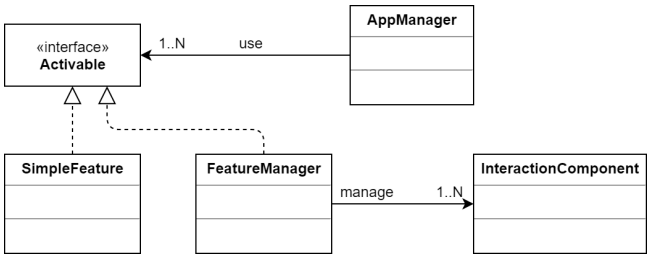

FIGURE 3.15 – Diagramme de classe du système de fonctionnalité (attributs et méthodes omis par souci de clarté)

La définition d'une telle structuration des fonctionnalités permet d'encapsuler la logique métier des fonctionnalités au sein d'éléments activables faiblement couplés. Par exemple, dans le prototype de cathédrale, les différentes fonctionnalités sont réalisées en respectant les concepts précédemment définis. Les fonctionnalités de réinitialisation de la scène et d'affichage de la cartographie spatiale sont des fonctionnalités à actions immédiates tandis que les autres sont des fonctionnalités complexes, le tout étant géré par un gestionnaire central responsable de la barre d'outils. Lorsque l'utilisateur actionne un bouton dans cette dernière, le gestionnaire central désactive la fonctionnalité en cours, s'il y en a une, et active celle que l'utilisateur a sélectionnée.

3.2.3.2 Système de trajectoire

Afin de générer dynamiquement des structures architecturales ou abstraites, un module de trajectoires dans l'espace a été conçu. Cette fonctionnalité est nommée *tracing* et permet de créer des trajectoires définies par une formule

mathématique. La principale motivation à la réalisation de ce module est la création dynamique et automatique de structures plus ou moins complexes pouvant être définies à l'aide de concepts mathématiques. Dans les deux prototypes de visualisation présentés, *tracing* est utilisé pour générer les voûtes de cathédrales et les liens du mobile.

Au sein de ce module, nous retrouvons le concept de trajectoire, mais également un système de décoration permettant d'obtenir un rendu visuel de ces dernières. Ce système est responsable de la création dynamique d'un modèle 3D respectant les caractéristiques de la trajectoire qu'il décore. Chaque élément du module *tracing* dispose d'un ensemble de paramètres permettant d'adapter son comportement ou son apparence. *Tracing* est construit sur une architecture extensible facilitant la création de nouvelles trajectoires. Cette extensibilité est concrétisée par l'utilisation du paradigme orienté-objet et le système Entity-Component du moteur Unity. Dès lors, toute trajectoire est modélisée comme un tuple $t = (origin, f)$ où $origin$ est le point de départ de t sous la forme (x, y, z) et f est la fonction décrivant la trajectoire dans l'espace. La fonction f est définie comme

$$f : [0, 1] \longrightarrow \mathbb{R}^3$$
$$f(a) = (x, y, z) \in \mathbb{R}^3$$
$$f(0) = origin$$

$f(a)$ représente le point dans l'espace de t au pourcentage d'avancement a exprimé avec une valeur réelle comprise entre 0 et 1. Avec cette définition, il est possible de récupérer l'infinité de points constituant la trajectoire, grâce à la fonction f, ainsi que le vecteur tangent de chacun des points, avec $\frac{f(x)}{dx}$. Ce concept fut ensuite utilisé pour définir des segments de droite dans l'espace, des courbes de Bézier et des arcs de cercle comme le montre la figure 3.16.

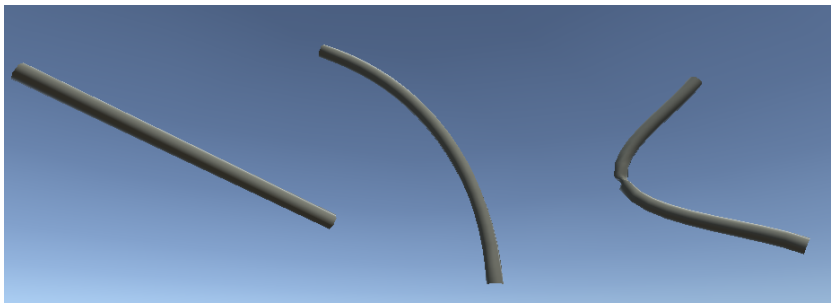

FIGURE 3.16 – Trajectoires crées à l'aide du système *tracing*

3.3 Interactions gestuelles sur les mobiles

Le casque HoloLens étant en mesure de reconnaitre les gestes, ces derniers sont utilisés pour interagir avec la visualisation de mobiles. Ayant pour objectif d'être un outil facilitant le travail d'analyse des analystes de texte, elle se doit de proposer un nombre important d'interactions. Dès lors, les trois gestes reconnus par l'HoloLens (*air tap*, *tap and hold* et *manipulation*) ont été utilisés pour permettre à l'utilisateur d'interagir avec le ou les mobiles.

3.3.1 Création de relations

Lors de l'analyse d'un texte selon la technique d'analyse structurale, l'analyste identifie les relations de disjonction au fur et à mesure de son analyse du texte. Par conséquent, la visualisation doit permettre la création de relations de disjonction sous la forme d'un mobile. Dès lors, au lancement de l'application, un bouton permet à l'utilisateur de créer un *MobileLine* symbolisant une relation de disjonction. Il peut ensuite créer une sous-relation à partir des *MobileSolids*, élaborant ainsi une structure hiérarchique. Le geste *air tap* est utilisé à cet effet : en effectuant ce geste, tout en pointant un *MobileSolid* à l'aide du regard, un nouveau *MobileLine* sera créé à partir de l'élément sélectionné. La figure 3.17 illustre la création d'une relation de disjonction tandis que la figure 3.18 montre la création d'une structure hiérarchique.

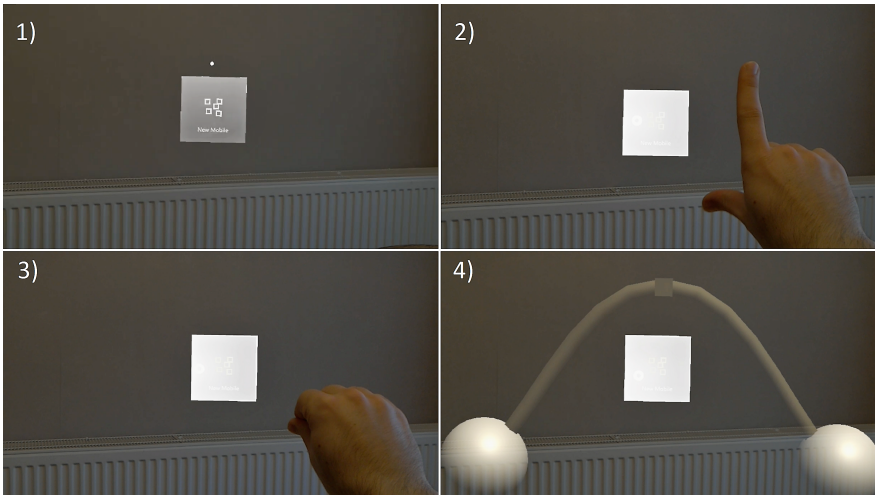

FIGURE 3.17 – Création d'une relation de disjonction

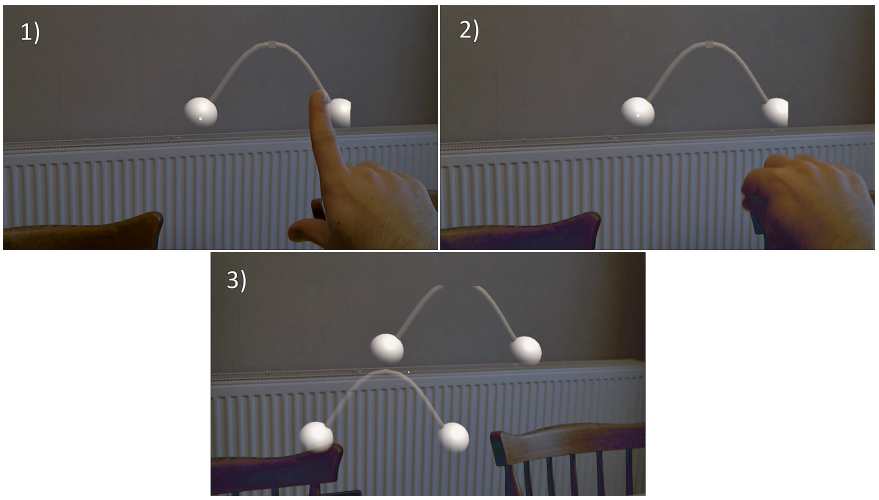

FIGURE 3.18 – Création d'une structure hiérarchique

3.3.2 Nommage des termes

Une fois des relations créées, l'utilisateur est en mesure de stipuler les termes symbolisés par les *MobileSolids*. Pour ce faire, il doit effectuer un *tap and hold* sur ce dernier pour faire apparaitre une boite de dialogue et un clavier virtuel permettant la saisie de texte comme le montre la figure 3.19. Une fois un terme défini, il peut être modifié en réalisant la même action.

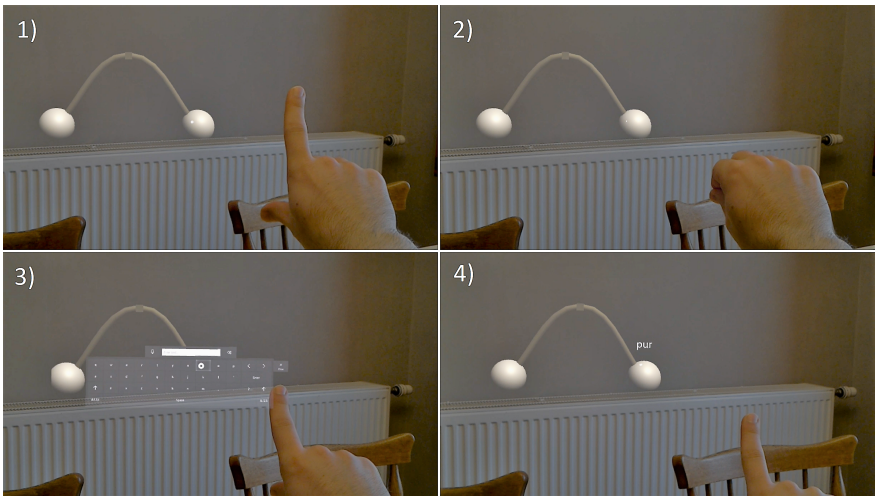

FIGURE 3.19 – Nommage d'un *MobileSolid*

3.3.3 Manipulation de mobiles

Enfin, à l'aide du geste *manipulation*, deux interactions sont rendues possibles : l'utilisateur est en mesure de déplacer un mobile au sein de son environnement, mais également de modifier l'écartement des termes d'une relation de disjonction (d_l ou d_r).

Pour déplacer un mobile, un cube de petite taille est disposé sur la courbe de la relation de disjonction de plus haut niveau (en cas de hiérarchie). En effectuant un geste *manipulation* sur ce dernier, l'utilisateur est en mesure de déplacer l'entièreté du mobile. Cette interaction est illustrée par la figure 3.20.

Enfin, le geste *manipulation* réalisé en pointant un terme, représenté par une sphère, aura pour effet d'ajuster son écartement sur la relation de disjonction à laquelle il appartient comme le montre la figure 3.21.

3.3.4 Interactions manquantes

L'ensemble des gestes détectés par le casque est donc utilisé pour permettre à l'utilisateur d'interagir avec la visualisation. Cependant, les différents paramètres de la visualisation induisent des actions supplémentaires, mais également nécessaires à la flexibilité qu'ils autorisent. En effet, comme présenté dans les sections précédentes, les gestes de l'HoloLens permettent de créer des relations de disjonction, de nommer les termes, déplacer les mobiles et créer des structures

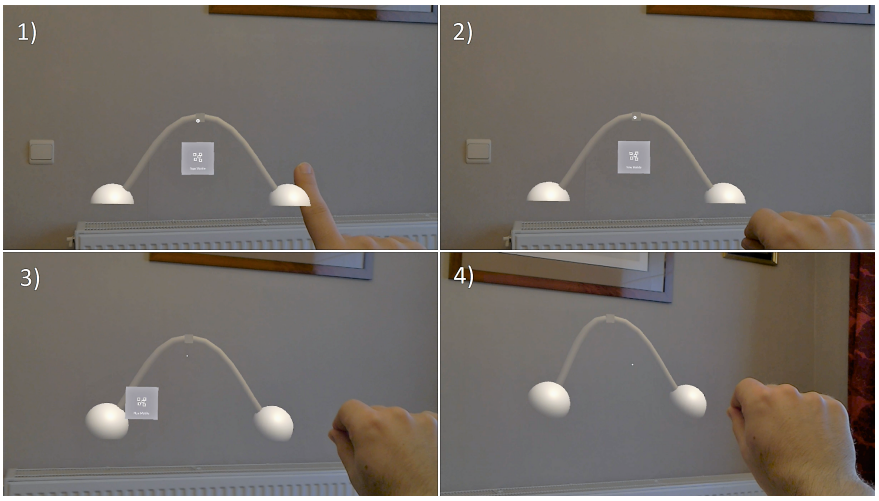

FIGURE 3.20 – Déplacement d'un mobile

hiérarchiques. Néamoins, nous identifions, ci-dessous, des actions pertinentes pour lesquelles des interactions de la visualisation sont manquantes.

Structures parallèles et croisées Avec les interactions mises en oeuvre, l'utilisateur n'est pas en mesure d'ajuster le vecteur d'orientation $\vec{o}$ ou le vecteur de translation verticale $\vec{h}$ sur lesquels reposent les visualisations des structures parallèles et hiérarchiques. Ces dernières pourraient bénéficier d'une interaction dédiée à leur création ou en laissant l'utilisateur agir sur les vecteurs pour les incarner.

Suppression de relations Comme pour la plupart des visualisations d'information, la suppression d'une ou plusieurs relations doit être possible, et ce, dans le cas d'une structure hiérarchique, sans supprimer les sous-relations qui y sont liées.

Dissociation de structures Concernant les structures de l'analyse structurale, l'utilisateur doit être en mesure de dissocier les relations qui les constituent. L'analyse étant effectuée progressivement sur un texte, l'analyste est susceptible de créer une structure de relation avant de se rendre compte que le sens des relations qu'il a créées ne correspond pas à celui véhiculé par le texte et son auteur.

Mise à l'échelle Durant la construction et l'interaction avec la visualisation, l'analyste est susceptible de vouloir mettre en évidence les éléments sur lesquels il concentre son attention. Cette mise en évidence peut être réalisée en réduisant/augmentant la taille des différents mobiles présents.

Sélection Pour faciliter la réalisation des tâches précédemment citées sur plusieurs mobiles, l'utilisateur doit pouvoir sélectionner plusieurs structures pour ensuite indiquer l'action qu'il leur désire appliquée. Cette fonctionnalité à l'avantage de réduire la friction nécessaire à l'accomplissement de tâches sur un nombre important d'éléments.

Par conséquent, de nombreuses fonctionnalités, parfois fondamentales, ne sont pas concrétisables avec les gestes reconnus par l'HoloLens. Cette limitation

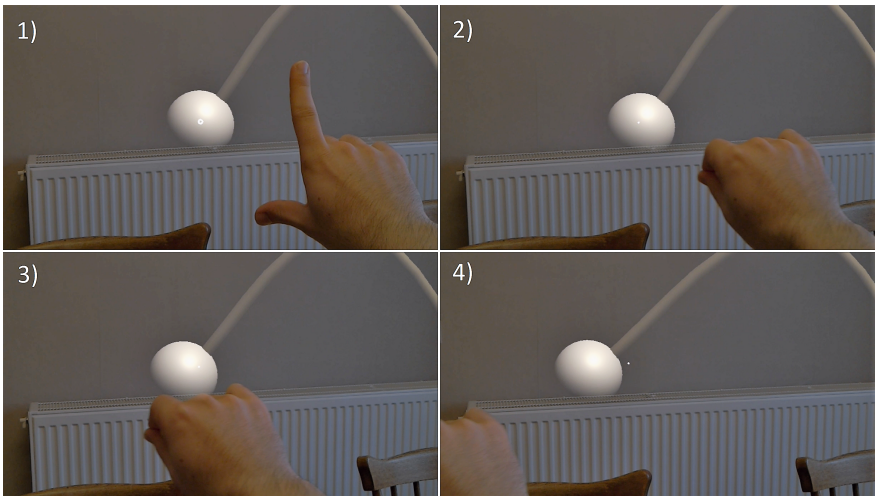

FIGURE 3.21 – Modification de l'écartement d'un *MobileSolid*

peut être évitée par l'introduction d'une barre d'outils au prix de l'ajout de frictions et de complexités dans les interactions, comme mentionné dans (Brath, 2014) et concernant *CathedralBuilder*. Dès lors, la reconnaissance de geste du casque HoloLens doit être étendue pour proposer des interactions simplifiées sur les visualisations de mobiles.

3.4 Extension des gestes

3.4.1 Gestes supplémentaires

En se référant aux gestes AR définis par les utilisateurs, présentés dans (Piumsomboon *et al.*, 2013), un ou plusieurs gestes pouvant correspondre aux interactions manquantes sont proposés. En effet, en plus d'exposer la taxonomie de gestes AR détaillée précédemment, Piumsomboon *et al.* dressent un ensemble de 44 gestes destinés à l'accomplissement de 40 tâches telles que la transformation d'éléments 3D, la sélection, la navigation et l'édition en réalité augmentée.

Pour la création de structures parallèles et croisées, la modification des vecteurs $\vec{h}$ et $\vec{o}$ est requise. Dès lors, plusieurs gestes de pronation sont proposés (cf. figure 3.22(a)) pour la modification de la translation verticale $\vec{h}$ des *MobileSolids* d'une relation de disjonction. Ces gestes permettent d'agripper un *MobileSolid* pour ensuite mapper la translation de la main de l'utilisateur au vecteur $\vec{h}$, tant que le geste de pronation est réalisé. Une telle interaction autorise la modification de $\vec{h}$ dans les trois dimensions de l'espace. D'autre part, l'ajustement de l'orientation d'un mobile (vecteur $\vec{o}$) ne nécessite pas une telle liberté de mouvement, car seuls les axes des x et des z de $\vec{o}$ sont pertinents pour le calcul de l'orientation. Par conséquent, deux gestes sont sélectionnés : le tournoiement de l'index (cf. figure 3.22(b)) et le pincement à deux mains (cf. figure 3.22(c)). Ce dernier est inspiré du système de rotation bimanuelle présenté dans 6D Hands reposant sur la métaphore de la manipulation d'une feuille de papier (Wang *et al.*, 2011). Dans les deux cas, la relation est pointée du regard par l'utilisa-

teur et son orientation est modifiée en fonction de la rotation exprimée par un des deux gestes que l'analyste réalise.

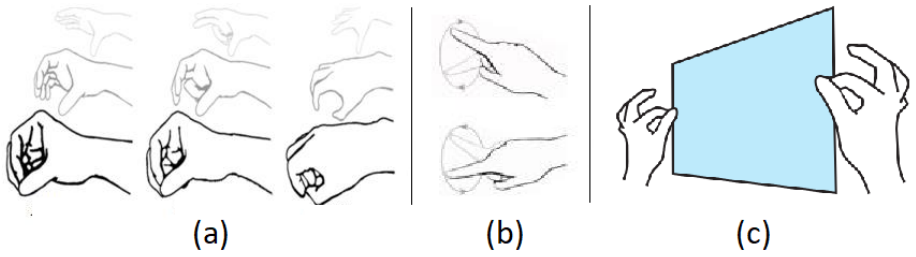

FIGURE 3.22 – (a) Gestes de pronation, (b) tournoiement de l'index, (c) pincement à deux mains. Sources : (a), (b) : (Piumsomboon *et al.*, 2013) et (c) (Wang *et al.*, 2011)

De plus, le poing, illustré par la figure 3.23, est associé par Piumsomboon *et al.* à la tâche de suppression d'éléments virtuels. De ce fait, ce geste est proposé pour la suppression d'une relation, visée avec le regard, au sein de la visualisation. Piumsomboon *et al.* donnent cependant une description plus précise de ce geste : l'action se réalise en agrippant l'élément (geste de pronation) pour ensuite l'écraser en serrant les doigts. Cette description implique que l'élément doit être à une distance relativement réduite de l'utilisateur pour lui permettre de l'agripper. Par ailleurs, l'action de pronation de l'élément représente un moyen de sélection de l'objet à faire disparaitre. Avec le suivi du regard du casque HoloLens, la pronation et la contrainte qu'elle implique sont évitées étant donné que l'objet à supprimer est distingué à l'aide du regard.

FIGURE 3.23 – Geste de suppression d'une relation. Source : (Piumsomboon *et al.*, 2013)

Par ailleurs, pour la fonctionnalité de dissociation de structures, la métaphore de découpe a été choisie. En effet, la séparation de relations de disjonction peut être considérée comme une coupe de la structure pour la dissocier. La tâche *Cut* et son geste associé, défini dans (Piumsomboon *et al.*, 2013), incarne cette métaphore comme montré par la figure 3.24. Dès lors, en désignant un *MobileLine* (ou un de ses *MobileSolids*) avec le regard, l'analyste effectue le geste symbolisant une coupe avec des ciseaux pour séparer la relation de la structure dans laquelle elle se trouve.

Ensuite, un geste de pronation à deux mains, illustré par la figure 3.25, est sélectionné pour incarner la mise à l'échelle d'un mobile. Toujours en pointant un mobile avec le regard, l'agrandissement de ce dernier est réalisé en écartant les mains lors d'un geste de pronation à deux mains. À l'inverse, le rapprochement

FIGURE 3.24 – Geste de la métaphore de découpe, détaillée dans (Piumsomboon *et al.*, 2013)

des mains diminuera la taille du mobile.

FIGURE 3.25 – Pronation à deux mains pour modifier, augmenter ou réduire, l'échelle d'un mobile. Source : (Piumsomboon *et al.*, 2013)

Enfin, l'utilisation des gestes précédemment cités peut être appliquée à un groupe d'éléments, préalablement sélectionnés par l'utilisateur. Le geste de sélection proposé repose sur la tâche de sélection multiple définie dans (Piumsomboon *et al.*, 2013) pour laquelle le geste correspondant consiste à toucher successivement les objets à sélectionner avec l'index, éventuellement l'index et le majeur, comme le montre la figure 3.26. Comme pour la fonctionnalité de suppression de relation, le contact physique entre les doigts de l'utilisateur et l'élément n'est pas conservé dans le cadre de la visualisation de mobile.

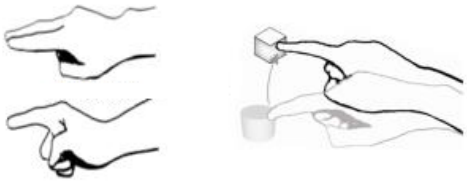

FIGURE 3.26 – Geste de pointage permettant la sélection d'éléments. Source : (Piumsomboon *et al.*, 2013)

3.4.2 Limitations du casque

En se référant aux techniques de détection et reconnaissance de gestes décrites dans la section 2.3.2 Reconnaissance de gestes et les multiples capteurs présents sur le casque HoloLens, la Computer Vision représente une solution pour étendre les interactions gestuelles du casque. En effet, l'HoloLens dispose d'une caméra de profondeur et est capable de cartographier l'environnement. Dès lors, l'ensemble des données qu'il récupère peuvent être mises à profit.

Cependant, les interfaces de programmation du casque n'autorisent pas l'accès aux données brutes qu'il obtient. L'architecture matérielle (présentée par la figure 2.24 page 29) semble être à l'origine de cette limitation. Comme présenté précédemment, l'Holographic Processing Unit est le composant matériel responsable de la reconnaissance de gestes. Il est le seul à posséder un accès direct à la plupart des capteurs de l'HoloLens. Ainsi, les seules données brutes intéressantes disponibles, pour la reconnaissance de gestes, sont celles obtenues par l'unique caméra vidéo directement connectée au CPU Intel Cherry Trail. Par conséquent, l'utilisation de Computer Vision reposant sur les données recueillies par le casque est compromise par l'indisponibilité des données de profondeur. En effet, comme détaillé dans la section 2.3.2 Reconnaissance de gestes, les contributions reposant sur un flux vidéo unique ne proposent qu'un nombre restreint de gestes peu intuitifs, définis pour faciliter leur détection, ou nécessitent l'utilisation de gants ou de marqueurs de référence.

Concernant les données de cartographie spatiale, les données accessibles sont définies comme un ensemble de surfaces géométriques correspondant aux surfaces réelles de l'environnement. Néanmoins, la précision de ces données n'est pas destinée à effectuer de la modélisation 3D précise, ce qui proscrit toute utilisation dans le cadre de la reconnaissance et détection des mains. Ce manque de précision est notamment présenté dans les travaux de Garon *et al.* comme le montre la figure 3.27 où les données environnementales obtenues sont comparées à celles d'une caméra de profondeur Intel RealSense (Garon *et al.*, 2016).

(a) HoloLens 3D (b) RealSense 3D (c) Scene

FIGURE 3.27 – Figurine (c) dont les données environnementales sont récupérées avec (a) l'HoloLens et (b) une caméra Intel RealSense (Garon *et al.*, 2016)

Par conséquent, sans un accès aux données brutes recueillies par le casque HoloLens, une extension de ses capacités de détection et reconnaissance de gestes n'est pas réalisable, sans avoir recours à des éléments externes fournissant des données exploitables. Dès lors, il est nécessaire d'introduire un système de reconnaissance et détection des gestes additionnels pour pouvoir mettre en oeuvre les interactions gestuelles nécessaires à la visualisation de mobiles. L'utilisation d'un système additionnel est discutée dans la section suivante.

3.4.3 Solution proposée

Comme mentionné dans la section précédente, l'HoloLens ne permet pas d'accéder aux données brutes obtenues par les capteurs dont il dispose. Dès lors, pour détecter et reconnaitre des gestes supplémentaires, il est nécessaire d'introduire un ou plusieurs capteurs, connectés au casque, pour avoir accès à des données utilisables pour les techniques de Computer Vision. Cependant, contrairement à un ordinateur, le casque n'autorise pas la connexion de périphé-

riques pouvant récupérer de telles données. En effet, un seul port Micro-USB est présent sur le casque et ce dernier est destiné au chargement de la batterie et au déploiement d'applications lors de leur processus de développement. Ainsi, nous ne sommes pas en mesure de connecter un quelconque périphérique supplémentaire sur le casque.

Par ailleurs, comme montré dans (Bai *et al.*, 2015) et (Garon *et al.*, 2016), cette problématique peut être résolue par l'introduction d'une communication réseau entre le dispositif de réalité augmentée et un ordinateur muni du ou des périphériques devant être utilisés. En effet, Bai *et al.* présentent l'intégration d'un système client-serveur avec d'un côté, une caméra de profondeur connectée à un serveur de reconnaissance de gestes et de l'autre, une application AR sur smartphone. Les utilisateurs profitant alors d'une interaction gestuelle sur l'application AR installée sur leur smartphone et communiquant avec le serveur de détection et reconnaissance de gestes. D'autre part, Garon *et al.* proposent un système client-serveur similaire liant une caméra Intel RealSense et un casque HoloLens, dans l'objectif de fournir une cartographie spatiale plus détaillée (cf. figure 3.27(b)).

Par conséquent, l'intégration d'un système de reconnaissance de geste avec le casque HoloLens peut être réalisée d'une façon similaire à celles présentées dans (Bai *et al.*, 2015) et (Garon *et al.*, 2016). Dès lors, deux architectures d'applications holographiques à gestuelles étendues pour l'HoloLens sont conçues et détaillées ci-dessous.

Pour définir les deux architectures d'applications HoloLens à gestuelles étendues, deux types d'interactions gestuelles sont identifiées : directes (naturelles) ou indirectes. Le premier nécessite un lien étroit entre le geste et les éléments virtuels de l'application, à savoir manipuler des éléments virtuels comme s'ils étaient réels. Par exemple, attraper un objet pour ensuite le déplacer, pousser ou encore jeter un élément virtuel. En effet, une telle interaction impose une présence physique (position, orientation, vitesse de déplacement...) des mains de l'utilisateur au sein du monde virtuel. À l'inverse, les interactions indirectes concernent les gestes n'ayant aucun lien direct avec les éléments virtuels. À titre d'exemple, le tournoiement de l'index (cf. figure 3.22(b)) est un geste dont la présence physique dans le monde virtuel n'est pas pertinente pour la bonne exécution de la tâche à laquelle il correspond. En s'inspirant des travaux présentés par Bai *et al.* et Garon *et al.*, deux architectures sont proposées, la première permet la concrétisation d'interactions indirectes à l'aide d'une architecture client-serveur tandis que la seconde, plus complexe au profit d'une flexibilité accrue, s'adresse aux deux types d'interactions en tirant parti de l'API multijoueur d'Unity.

3.4.3.1 Architecture client-serveur

Lorsque l'application nécessite la détection de gestes indirects, une architecture client-serveur, illustrée par la figure 3.28, est proposée. D'une part, un serveur est responsable de la détection et la reconnaissance des nouveaux gestes à l'aide d'un ou plusieurs périphériques dédiés. Par exemple, des caméras RGB et de profondeur. D'autre part, l'HoloLens, exécutant l'application holographique, incarne le client communiquant avec le serveur à l'aide d'une interface réseau (Wi-Fi) pour récupérer les gestes effectués par l'utilisateur.

Côté serveur, sachant que l'architecture est destinée aux interactions indi-

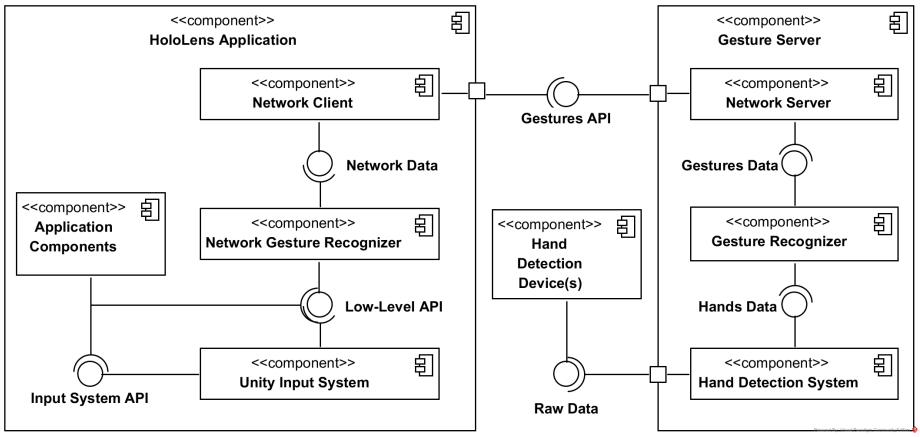

FIGURE 3.28 – Architecture client-serveur destinée aux interactions indirectes

rectes, ce dernier n'a pas besoin de connaître l'état de l'application holographique. Dès lors, il est entièrement responsable du processus de détection et de reconnaissance de gestes. Par ailleurs, cette architecture à l'avantage d'être indépendantes de la technologie utilisée pour détecter les mains. Qu'il s'agisse d'une caméra de profondeur, d'une paire de caméras RGB ou encore d'un périphiques plus spécifique comme un Leap Motion [1], un composant *Hand Detection System* sera toujours présent pour traiter les données brutes récupérées par le ou les *Hand Detection Devices*. Il en est de même pour le composant *Gesture Recognizer* responsable de la reconnaissance de gestes : la technique de reconnaissance de geste est laissée à la discrétion des développeurs, qu'il s'agisse d'une base de données de représentations matricielles de gestes interrogée selon la méthode des plus proches voisins, de l'utilisation de modèles de Markov cachés ou autre. D'autre part, la communication réseau est contrôlée par le composant *Network Server*. C'est avec ce dernier que le casque HoloLens, le client, va interagir.

De plus, côté client, différents composants sont mis en oeuvre et détaillés ci-dessous :

Network Client Composant responsable de la communication réseau avec le serveur. Il supervise la connexion ainsi que l'envoi et la réception des données de détection de gestes.

Network Gesture Recognizer Composant principal de l'architecture, il permet à l'application de gérer les différents gestes supportés. Pour ce faire, il permet aux composants de l'application de définir un ensemble d'action devant être exécutée lorsqu'un geste donné est reconnu. Dès lors, ce composant propose, pour chaque geste, un groupe d'évènements auquel les composants de l'application s'inscrivent pour être notifiés de leur concrétisation.

Unity Input System Le système d'entrée d'Unity, brièvement mentionné dans la section 2.4.3.2, est responsable de la gestion des entrées de l'application en déterminant les évènements qui y sont liés pour les transmettre aux éléments concernés (Unity, 2017a). À titre d'exemple, si un élément

1. https://www.leapmotion.com/

doit réagir au clic de la souris, l'*Input System* est responsable de la détection du clic, de l'identification de l'élément visé et de l'envoi de l'évènement correspondant à ce dernier. Dès lors, l'élément concerné, s'étant inscrit à l'évènement, sera notifié de la manifestation dudit évènement. Ce système est extensible et supporte l'intégration d'entrées personnalisées. Par conséquent, l'ajout des gestes détectés par le serveur représente un avantage non négligeable au regard des fonctionnalités qu'il apporte, les gestes n'étant que des entrées au même titre que l'appui d'une touche de clavier ou d'un clic de souris.

Application Components Cet ensemble de composants représente tous les éléments propres à l'application qui sont susceptibles d'interagir avec soit le composant *Network Gesture Recognizer*, soit l'*Unity Input System*. Bien que la plupart des cas d'utilisation auront recours au système d'entrées d'Unity, lorsqu'un geste doit être détecté indépendamment de l'élément visé, il est nécessaire de pouvoir interagir directement avec le *Network Gesture Recognizer*.

Par ailleurs, le processus de communication entre le serveur et le ou les clients est défini par les diagrammes de séquences illustrés par les figures 3.29 et 3.30. Nous distinguons deux scénarii de communication de reconnaissance de geste : la notification d'un geste statique (cf. figure 3.29) et celle d'un geste dynamique (cf. figure 3.30). Dans les deux cas, lorsqu'un client se connecte au serveur, il est en mesure de s'inscrire aux évènements de détection d'un ou plusieurs gestes. Par conséquent, le serveur conserve une liste des clients et les gestes auxquels ils sont inscrits. Ainsi, lorsqu'un geste est reconnu, le serveur ne notifiera que les clients inscrits au geste détecté afin de réduire toute communication réseau inutile. Dans la même optique, un client est en mesure de se désinscrire d'un geste auquel il s'est inscrit préalablement.

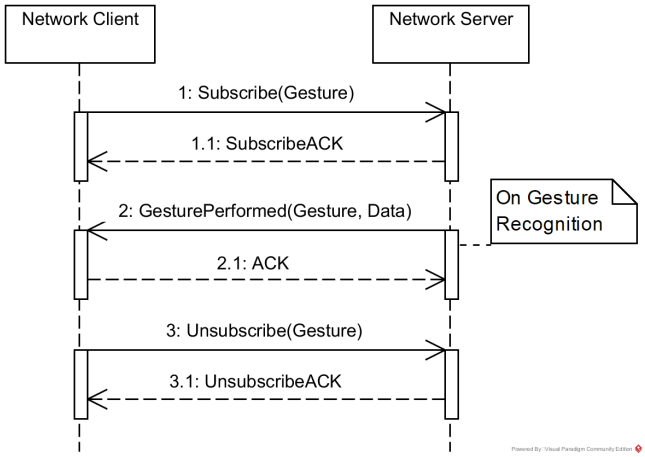

FIGURE 3.29 – Diagramme de séquences de la communication réseau d'un geste statique

Pour la notification d'un geste statique, la séquence d'actions est triviale : lorsqu'un geste statique est reconnu par le serveur et sujet à une inscription de la part d'au moins un client, le serveur notifiera les clients inscrits. Cette

notification est matérialisée par un message *GesturePerformed* indiquant le geste reconnu et contenant d'éventuelles données complémentaires le concernant.

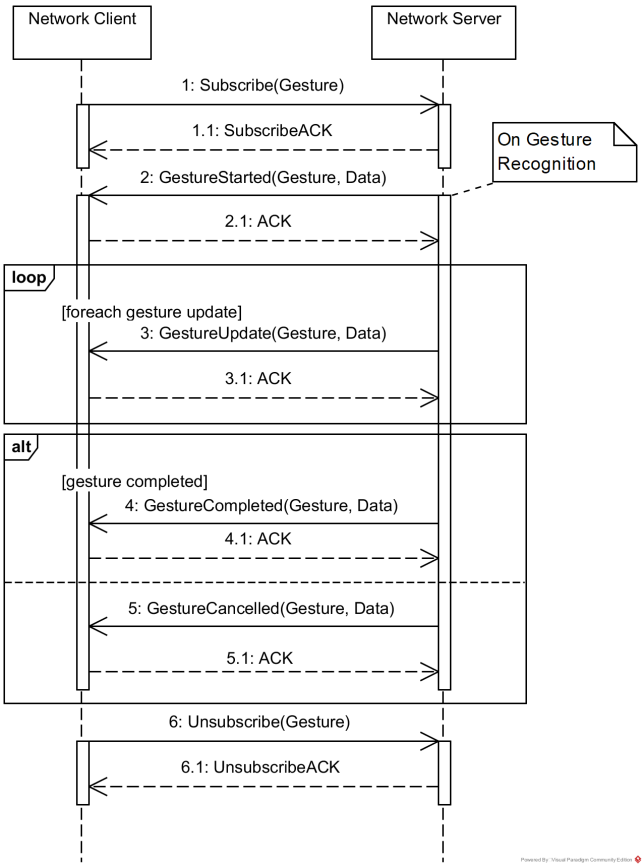

FIGURE 3.30 – Diagramme de séquences de la communication réseau d'un geste dynamique

D'autre part, la notification d'un geste dynamique implique une communication plus importante du serveur avec le client. Un geste dynamique, par définition, est composé d'un ou plusieurs mouvements contrairement à une pose comme pour un geste statique. Un exemple de geste dynamique est le tournoiement de l'index, détaillé précédemment. En plus de reconnaitre la pose de la main, il est nécessaire de suivre le déplacement effectué par l'index pour en déterminer la rotation effectuée. Dès lors, quatre types d'informations sont identifiables et nécessaires pour gérer un tel type de gestes, pour chacun, un message sera communiqué aux clients :

— Début du geste, avec le message *GestureStarted* ;
— Geste en cours, avec le message *GestureUpdate* ;
— Fin du geste, avec le message *GestureCompleted* ;
— Annulation du geste, avec le message *GestureCancelled*.

Notons qu'un message *GestureUpdate* sera envoyé plusieurs fois étant donné l'aspect dynamique des gestes. L'annulation du geste peut être assimilée à deux raisons : le geste dynamique possède une pose de fin de geste qui n'a pas été

effectuée ou reconnue ou encore le système de détection à perdu le suivi de la main de l'utilisateur.

Enfin, cette architecture autorise une liberté d'implémentation importante concernant le serveur de détection et reconnaissance de geste. En effet, la communication étant effectuée via le réseau, seules la sérialisation et la désérialisation des messages doivent être communes entre l'application HoloLens et le serveur.

3.4.3.2 Architecture multijoueur Unity

Pour proposer un support d'interactions gestuelles naturelles (directes), une seconde architecture est proposée. Comme mentionné précédemment, cette dernière, en plus de détecter et reconnaitre les mains de l'utilisateur, matérialise les mains au sein du monde virtuel géré par l'application HoloLens. Pour ce faire, la scène virtuelle doit être partagée, en partie ou complètement, entre l'application et le système d'interactions gestuelles étendues. Cette synchronisation du monde virtuel est nécessaire pour obtenir une incarnation physique des mains au sein de la scène. Dès lors, en ayant recours à la *Multiplayer High Level API* d'Unity (Unity, 2017b), l'architecture illustrée par la figure 3.31 est proposée. L'API multijoueur d'Unity facilite le développement d'application multijoueur en synchronisant, de façon transparente, le monde virtuel dans lequel l'ensemble des clients se trouvent. L'ensemble des éléments et leurs comportements sont partagés entre les différents clients. De plus, cette API ne nécessite pas de serveur dédié, une instance du serveur et d'un client coexistant sur la même machine.

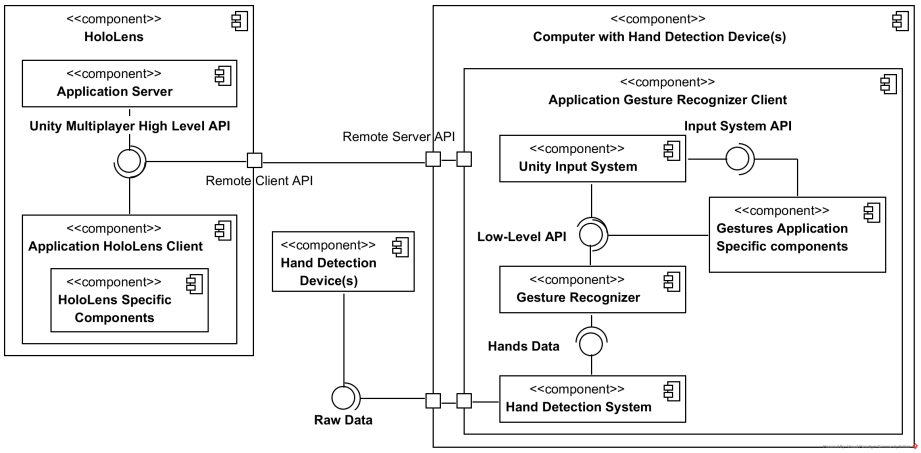

FIGURE 3.31 – Architecture mutlijoueur Unity

Dans cette architecture, nous retrouvons certains composants de l'architecture précédente, à savoir les périphériques destinés à la récupération des données brutes, les composants de détection et de reconnaissance de gestes et le système de gestion des entrées d'Unity. Néanmoins, l'ajout majeur de cette architecture est la séparation de l'application HoloLens en deux composants distincts : *Application Server* et *Application HoloLens Client* ainsi que l'application *Gesture Recognizer Client* s'exécutant sur la machine disposant des périphériques d'acquisition des données. Grâce à l'API multijoueur d'Unity, les deux applica-

tions clientes partagent le même monde virtuel et par conséquent ont accès à l'ensemble des éléments le composant, en plus de bénéficier de l'intégralité des fonctionnalités offertes par le moteur Unity. Par conséquent, l'application *Gesture Recognizer Client* est en mesure de créer une représentation physique des mains de l'utilisateur. Représentation qui sera ensuite synchronisée par l'API multijoueur.

Enfin, avec une telle organisation des différents composants en présence, l'extension des interactions gestuelles est contrainte aux fonctionnalités du moteur Unity. Dès lors, les techniques de détection et de reconnaissance de gestes se doivent d'être implémentées au sein de la pile technologique supportée par Unity. Cette limitation est cependant réduite par la possibilité d'importation de librairies externes. D'autre part, cette architecture implique une séparation claire des fonctionnalités reposant sur les méthodes d'interactions de l'HoloLens et celles liées aux interactions gestuelles étendues. En effet, l'API multijoueur d'Unity autorise le partage d'un même monde virtuel, mais il ne s'agit pas d'une réplication complète de l'état de l'application. Par conséquent, une division est introduite entre les fonctionnalités spécifiques à l'HoloLens et celle du client de détection et de reconnaissance de gestes. Par ailleurs, le choix d'instancier le serveur de l'application sur l'HoloLens et pas sur la machine de détection et reconnaissance de gestes est justifié par le besoin de réduire au maximum la communication réseau en présence. Avec le serveur s'exécutant sur l'HoloLens, seules les synchronisations avec le client distant nécessitent une réelle communication réseau, ce qui ne serait pas le cas en exécutant le serveur auprès du *Gesture Recognizer Client*.

3.4.4 Preuve de concept

Pour démontrer la faisabilité de l'extension des interactions gestuelles du casque HoloLens, une preuve de concept intégrant un Leap Motion a été réalisée. En conséquence, un geste supplémentaire est ajouté à la liste des gestes supportés par l'HoloLens. Cette section détaille brièvement les capacités de détection des mains du Leap Motion pour ensuite décrire l'intégration de ce dernier avec le casque et l'ajout du geste de suppression de relations de disjonction proposée dans la section 3.4.1 Gestes supplémentaires.

3.4.4.1 Leap Motion

Le contrôleur Leap Motion est un périphérique (cf. figure 3.32) constitué de deux caméras et trois LED infrarouges permettant de détecter les mains d'un utilisateur au sein d'un espace d'interactions aux dimensions d'une demi-sphère d'un rayon d'approximativement $60cm$ (LeapMotion, 2014). Comme mentionné dans (LeapMotion, 2014), la configuration matérielle du Leap Motion est assez simple, c'est au niveau logiciel que la majorité du traitement de détection des mains est effectué. En effet, à l'aide des deux caméras infrarouges, le Leap Motion dispose d'une paire de flux vidéo infrarouge à partir duquel le *Leap Motion Service* va effectuer une série de traitements produisant une représentation 3D des éléments situés dans son espace d'interactions. Par la suite, une couche de suivi va en extraire des informations comme les doigts d'une main.

Concernant les fonctionnalités offertes par le Leap Motion, il peut notamment être intégré au sein du moteur de jeu Unity. Il propose deux types d'ex-

FIGURE 3.32 – Contrôleur Leap Motion (LeapMotion, 2013)

périence : une expérience de bureau ou une expérience de réalité virtuelle. Pour la première, le contrôleur Leap Motion est placé à l'horizontale sur le bureau de l'utilisateur tandis qu'avec la seconde, il est attaché, à l'aide d'un dispositif dédié, à un casque de réalité virtuelle pour proposer une interaction naturelle avec les éléments du monde virtuel dans lequel l'utilisateur est immergé.

Par ailleurs, les données déduites par le *Leap Motion Service* concernent les mains, les bras et les doigts sont obtenues (LeapMotion, 2016) et accessibles aux applications. Dès lors, l'API contrôlant le Leap Motion permet de récupérer ces informations.

Mains Lorsqu'une ou plusieurs mains sont détectées par le Leap Motion, les développeurs sont en mesure de récupérer leur position et leur orientation dans l'espace d'interactions du contrôleur Leap Motion. En plus de ces informations, les bras et les doigts d'une main peuvent être obtenus et par conséquent, les données qui les concernent (voir ci-dessous). Par ailleurs, la modélisation d'une main étant inférée à l'aide des flux vidéo infrarouges, la détection des mains est soumise à une métrique de confiance indiquant la fiabilité de la détection. L'ensemble des mains détectées, et les données les concernant sont enregistrées dans une liste. Le contrôleur Leap Motion est en mesure de détecter plus de mains que celles de l'utilisateur, néanmoins il est conseillé de restreindre ce nombre à deux.

Bras Les avant-bras sont également modélisés par le contrôleur Leap Motion. Concernant ces derniers, l'API donne accès à des informations telles que leur longueur, leur orientation et leurs extrémités. Enfin lorsque le coude n'est pas visible dans la zone d'interaction, le *Leap Motion Service* en déduit sa position, soit sur base d'une détection précédente, soit selon des proportions caractéristiques du corps humain.

Doigts Le contrôleur Leap Motion est en mesure de représenter l'ensemble des doigts d'une main. Comme pour le coude de l'avant-bras, lorsque des doigts sont hors du champ de vision, leur dimension, position et orientation sont estimées selon les récentes observations et les proportions du corps humain. Les différents types de doigts sont également classifiés (pouce, index, majeur, annulaire et auriculaire) et chacun d'eux contient une liste d'os comme le montre la figure 3.33. La position, l'orientation et la dimension de l'ensemble des éléments présentés sont récupérables à l'aide de l'API.

3.4.4.2 Intégration

Pour intégrer un Leap Motion avec le casque HoloLens et ainsi étendre les interactions gestuelles de ce dernier, l'application *Remote Holographic Player*[2] est

2. https://www.microsoft.com/en-us/store/p/holographic-remoting-player/9nblggh4sv40

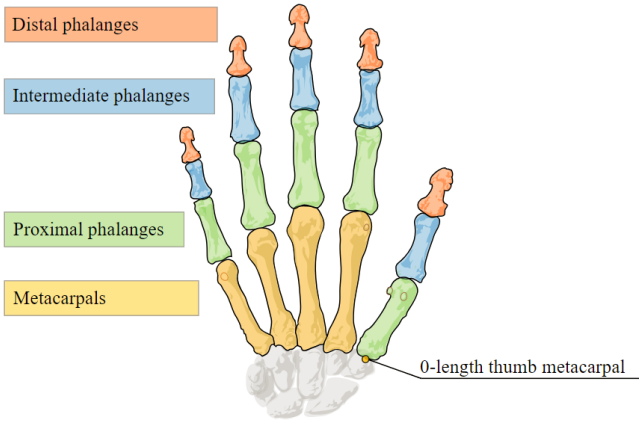

FIGURE 3.33 – Modélisation des os d'une main par le Leap Motion (LeapMotion, 2016)

utilisée. Cette application dédiée à l'HoloLens permet, en connectant le casque HoloLens avec un ordinateur situé dans le même réseau local, d'utiliser une application holographique sur le casque tandis que son exécution est effectuée sur l'ordinateur. L'ensemble des données recueillies par le casque sont alors transmises à l'ordinateur tandis que les données d'affichage sont transmises au casque. L'éditeur du moteur Unity est compatible avec cette application. Par conséquent, l'installation illustrée par la figure 3.34 est mise en place. Nous y retrouvons un ordinateur, muni de l'éditeur Unity, connecté au casque HoloLens sur lequel un contrôleur Leap Motion est attaché et relié par USB à l'ordinateur. Dès lors, l'application développée avec le moteur Unity ne doit pas être déployée sur l'HoloLens. *Remote Holographic Player* a été créée pour faciliter le développement d'application HoloLens en éliminant le temps nécessaire au déploiement d'une application sur le casque.

En ayant recours à *Remote Holographic Player*, l'application développée à l'aide du moteur Unity dispose d'un accès aux interfaces de développement du casque HoloLens, mais également à l'API Leap Motion de façon tout à fait transparente. Ainsi, le projet Unity s'exécute comme s'il était déployé sur un HoloLens disposant d'une connexion directe avec un Leap Motion et d'un accès à ses interfaces de programmation. Cependant, cette technique exécute l'entièreté de l'application sur l'ordinateur et les données nécessaires (données des capteurs HoloLens et l'affichage de l'application) sont toutes transférées via la connexion Wi-Fi. Ce qui implique un transfert de données important en plus de nécessiter un ordinateur capable d'exécuter une application 3D efficacement.

Ensuite, l'interaction avec le contrôleur Leap Motion a été ajoutée dans l'application de visualisation de mobile pour laquelle des interactions gestuelles étendues sont nécessaires. Disposant d'un accès à l'API du Leap Motion, les composants Unity la constituant permettent de créer une représentation 3D des mains détectées (cf. figure 3.35) intégrée avec le moteur physique d'Unity en plus de l'accès à l'ensemble des données inférées par le *Leap Motion Service* (cf. section 3.4.4.1 Leap Motion).

Enfin, en ayant accès aux données des mains, un composant *Leap Gesture*

FIGURE 3.34 – Installation de l'intégration d'un Leap Motion avec un casque HoloLens

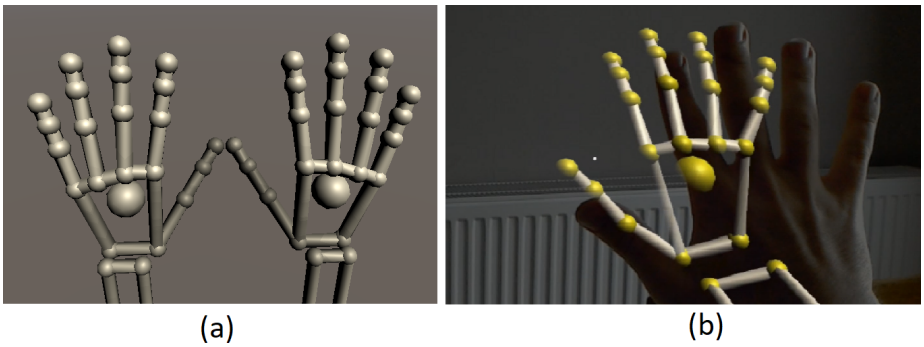

FIGURE 3.35 – (a) Représentation 3D des mains détectées par le contrôleur Leap Motion, (b) Modèle 3D des mains sur l'HoloLens

Recognizer a été développé. Ce composant est responsable de la détection du geste de suppression de relations de disjonction (la main fermée). À l'aide des données recueillies et inférées par le Leap Motion, le geste de suppression est défini selon deux paramètres : le taux d'agrippement (*GrabStrength*) et la direction du vecteur normal de la paume de la main. Le premier représente le taux d'ouverture des doigts d'une main par une valeur comprise entre l'intervalle [0..1] avec $GrabStrength = 0$ pour une main ouverte et $GrabStrength = 1$ pour une main fermée. Le deuxième paramètre a pour objectif de s'assurer que la reconnaissance du geste est validée uniquement lorsque la paume de la main est face au sol, évitant ainsi des faux positifs avec d'autres gestes tels que le *Bloom* reconnu par l'HoloLens. Pour ce faire, à chaque mise à jour de l'application (en moyenne de 60 fois/s), le processus de reconnaissance, représenté par le listing 3.1, est exécuté par *Leap Gesture Recognizer*. Avec une telle fréquence de mise à jour inhérente aux applications 3D en temps réel, le processus veille à ce que les actions correspondantes au geste détecté ne soient exécutées qu'une seule fois.

Dès lors, l'état précédent de la reconnaissance est conservé entre deux mises à jour et les actions du geste ne sont exécutées qu'au moment ou l'état de détection précédent est sujet à une transition de l'état « non-détecté » à « détecté ». Cet état est incarné par la valeur booléenne `previousState` dont la valeur `true` indique que le geste a été reconnu précédemment et `false` dans le cas contraire.

Listing 3.1 – Processus de reconnaissance du geste de suppression

```
boolean previousState = false;


onUpdate() :
    foreach Hand in HandsList
        boolean handClosed = hand.GrabStrength == 1
        boolean palmDown = hand.palmNormal == Vector.down
        boolean gesturePerformed = handClosed && palmDown
        if previousState != gesturePerformed :
            if gesturePerformed :
                execute gesture tasks
            endif
            previousState = gesturePerformed
        endif
        if gesturePerformed :
            break loop
        endif
    endforeach
end
```

Lorsque le geste est détecté, la relation de disjonction visée avec le regard est supprimée. Si cette relation est incluse dans une structure, seule la relation visée est supprimée conformément au comportement de la tâche de suppression de relation de disjonction discutée précédemment. La figure 3.36 illustre cette fonctionnalité actionnée par la détection du geste proposé pour cette dernière.

Pour conclure, la preuve de concept présentée dans cette section démontre la faisabilité d'interactions gestuelles étendues du casque HoloLens. À l'aide d'un Leap Motion, des données qu'il infère et sa compatibilité avec le moteur Unity, la reconnaissance de gestes supplémentaires est réalisable et les mains de l'utilisateur disposent d'une représentation physique au sein du monde virtuel. De plus, cette preuve de concept met en œuvre l'implémentation d'un geste proposé pour la réalisation d'une tâche au sein de la visualisation de mobiles en réalité augmentée.

3.5 Évaluation

3.5.1 Visualisation de mobiles

Concernant le prototype de visualisation de mobiles, l'atout majeur proposé vient du fait de pouvoir représenter des relations de disjonction et les différentes structures qu'elles sont susceptibles de former, conformément à la technique d'analyse structurale définie dans (Piret *et al.*, 1996). Par ailleurs, un soin particulier a été apporté sur l'évolutivité de la visualisation en définissant un mobile selon un ensemble de paramètres permettant d'en modifier aisément l'apparence.

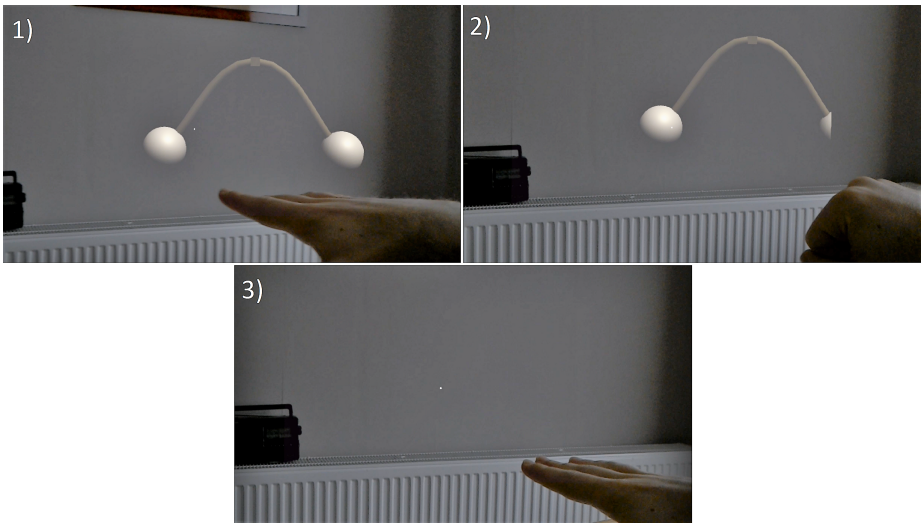

FIGURE 3.36 – Suppression d'une relation de disjonction à l'aide d'un geste reconnu à l'aide d'un Leap Motion

Le moteur Unity facilite cette évolutivité, car il propose de nombreuses fonctionnalités intégrables à la visualisation si elles sont nécessaires.

Cependant, la structure et l'implémentation de la visualisation comportent des limites. Un mobile artistique constitue une structure à configurations changeantes selon les forces qui lui sont appliquées alors que la visualisation proposée n'est pas soumise à de tels mouvements, forces ou reconfigurations. Le projet EFFaTA-MeM mène une réflexion sur les forces qui interviennent entre les différents mots et relations du texte, par conséquent, le caractère statique de la visualisation de mobiles incarne une limitation de la visualisation proposée.

De plus, bien que la grande innovation proposée par le casque HoloLens soit l'affichage d'hologrammes au sein du monde réel, la seconde innovation majeure du casque est sa capacité à cartographier l'environnement de l'utilisateur. Par ailleurs, dans le cadre de la visualisation de mobiles, la cartographie n'est pas exploitée et bien que son intégration puisse faire l'objet de travaux futurs, à première vue, ce type de fonctionnalité n'apporte pas une plus-value aux visualisations d'informations contrairement à d'autres applications du casque HoloLens au sein de l'industrie manufacturière par exemple.

Enfin, d'un point de vue de la modélisation 3D, la visualisation propose une représentation graphique rudimentaire reposant sur des modèles 3D et des textures basiques. La visualisation gagnerait en immersion en ayant une apparence plus naturelle et semblable aux éléments d'un mobile artistique. Néanmoins, la flexibilité du moteur Unity facilite l'évolution et la mise à jour de l'apparence sans impacter le reste de la visualisation. Dès lors, avec une modélisation adéquate, l'apparence peut être grandement améliorée sans effort conséquent.

3.5.2 Interactions gestuelles

En proposant des interactions gestuelles au sein de la visualisation, cette dernière est interactive en permettant à l'utilisateur, un analyste de texte, de

construire une visualisation de mobiles au fur et à mesure de son analyse. En effet, à l'aide des différentes tâches réalisables sur le mobile, il est en mesure de créer des relations de disjonction, d'en stipuler les termes en présence ou encore d'en modifier l'apparence. L'ensemble de ses tâches sont réalisables à l'aide de gestes statiques et dynamiques sélectionnés parmi les gestes de réalité augmentée établis dans (Piumsomboon *et al.*, 2013). Lorsque la gestuelle le permet, différents gestes alternatifs sont proposés pour fournir une plus grande liberté d'action à l'utilisateur qui sera libre d'utiliser le geste qu'il estime le plus cohérent conformément à la tâche qu'il souhaite voir réaliser.

De plus, les architectures proposées autorisent l'extension des interactions gestuelles du casque HoloLens ce qui représente une plus-value non négligeable pour la création d'applications AR sur ce dernier. Néanmoins, la solution proposée réduit la portabilité du casque en introduisant une dépendance avec un système de détection et reconnaissance de gestes nécessitant un ordinateur disposant d'un ou plusieurs périphériques récupérant les données brutes nécessaires aux diverses techniques de reconnaissance. Cette dépendance implique également une communication réseau ayant pour effet d'ajouter une latence dans la détection des mains de l'utilisateur. Les gestes supportés nativement par l'HoloLens sont reconnus au niveau matériel grâce à l'HPU qui permet une détection jusqu'à 20 fois plus rapide qu'une solution logicielle (cf. section 2.4.2 Spécifications). Dès lors, à moins que le système de reconnaissance utilisé pour étendre les interactions gestuelles ne soit également effectué au niveau matériel, le processus de détection et reconnaissance des gestes étendus sera plus lent de par son implémentation logicielle et la latence induite par la communication réseau nécessaire à l'intégration avec le casque HoloLens à savoir le transfert des différents messages pour la première architecture ou la synchronisation du monde virtuel dans la seconde.

Concernant l'intégration d'un Leap Motion en tant que preuve de concept de l'extension des interactions gestuelles de l'HoloLens, le Leap Motion apporte une détection précise des mains de l'utilisateur en allant jusqu'à la position et l'orientation de chaque os constituant une main. Cette quantité de données permet d'obtenir une détection en temps réel complète et flexible au détriment de la simplicité de définition d'un geste. En effet, pour reconnaitre un geste statique, il est nécessaire d'évaluer, sur l'ensemble des paramètres définissant les mains détectées, la configuration de chaque entité constituant une main. Pour la détection d'un geste dynamique, la problématique de définition d'un geste statique est intensifiée par le besoin de détection de l'évolution des paramètres sur lesquels le geste est défini. Par ailleurs, en étant compatible avec le moteur Unity, le Leap Motion permet de matérialiser les mains de l'utilisateur au sein de l'environnement virtuel et par conséquent d'interagir selon les lois de la physique avec les éléments virtuels.

3.5.3 HoloLens

Pour le casque HoloLens, la grande majorité des fonctionnalités qu'il propose ont été expérimentées, soit avec le prototype de cathédrale de texte soit avec la visualisation de mobile. Dès lors, différentes observations sont détaillées ci-dessous.

Tout d'abord, bien que la technologie d'affichage d'hologramme au sein de l'environnement soit fonctionnelle et permet de visualiser des éléments virtuels

superposés au monde réel, le champ de vision du casque, d'approximativement 35°, limite la quantité ou la taille des éléments du monde réel. En effet, avec un tel champ de vision, le monde virtuel semble visible à travers une fenêtre rectangulaire.

Ensuite, l'impossibilité d'accès aux données brutes recueillies par le casque représente un frein important aux expérimentations et recherches à l'aide de l'HoloLens malgré que cette limitation soit probablement une conséquence de l'architecture matérielle du casque. Dans le cas du Microsoft Kinect, de nombreuses recherches ont pu en tirer parti comme énoncé précédemment, grâce à l'accessibilité des données brutes qu'il obtient.

Enfin, le manque de maturité des interfaces de développement incarne une limitation importante du Microsoft HoloLens. En effet, alors que les fonctionnalités propres à l'HoloLens sont fiables, des surcouche logicielles telles la librairie open source Mixed Reality Toolkit Unity (MRTK [3]) sont instables. MRTK est une collection de composants réutilisables pour le développement d'applications holographiques avec le moteur Unity proposant des fonctionnalités telles qu'une librairie de post-traitement de la cartographie spatiale, des curseurs pour le regard, un clavier virtuel, une intégration des gestes HoloLens au sein du système d'entrée d'Unity ou encore un outil de déploiement automatique de l'application. Au jour du 8 mai 2018, 11 versions de MRTK ont été publiées depuis le 15 septembre 2017, chacune apportant des nouvelles fonctionnalités, des corrections de bogues, des changements de comportements des composants existants ou encore l'introduction de nouveaux bogues impliquant de nombreuses corrections dans une application utilisant la version précédente. La difficulté majeure étant l'obligation de mettre la librairie à jour, car une fonctionnalité nécessaire, mais défaillante est corrigée dans la nouvelle version. Par ailleurs, la documentation de MRTK se résume à des commentaires au sein du code et des projets Unity d'exemple qui sont, dans la plupart des cas, non fonctionnels, car ils ne sont pas adaptés pour correspondre à la nouvelle version de MRTK. Bien que Microsoft invite les développeurs à préférer le développement d'application avec Unity pour profiter de ses nombreuses fonctionnalités, certaines versions de MRTK nécessitent le passage sur une version de test de l'éditeur Unity impliquant une instabilité supplémentaire au sein de l'environnement de développement. Un exemple d'instabilité de MRTK est l'introduction d'un bogue dans le comportement des boutons virtuels, entre deux versions de la librairie (v2017.2.0 et v2017.2.1.3), ayant pour effet d'exécuter plusieurs fois les actions qui y sont liées lorsqu'il est activé. Ce bogue n'étant pas encore corrigé dans la version la plus récente de la librairie.

3. https://github.com/Microsoft/MixedRealityToolkit-Unity

Chapitre 4

Conclusion

Dans ce mémoire, une exploration approfondie des fonctionnalités du casque de réalité Microsoft HoloLens est réalisée par la création de différents prototypes de visualisations de données de l'analyse structurale de texte et l'intégration d'interactions gestuelles dans ces dernières.

Deux questions de recherches ont guidé la recherche présentée dans ce mémoire :

— Quelles visualisations de réalité augmentée peuvent être élaborées pour l'analyse structurale avec le casque HoloLens ?

— Quelles sont les méthodes d'interactions gestuelles, appliquées à un cas de manipulation de visualisations, pouvant être mises en oeuvre avec le casque Microsoft HoloLens ?

Pour ce faire, la première phase de recherche s'est concentrée sur l'étude de la technologie et des outils nécessaires à la réalisation d'applications destinées au casque HoloLens. Cette exploration eut pour résultat un prototype de création de structure de piliers et de voûtes de cathédrales dont l'intérêt fut cependant nuancé lors de réunions de travail du projet de recherche EFFaTA-MeM. Néanmoins, ce prototype mettait à profit la majorité des fonctionnalités de l'HoloLens et a donc permis une analyse détaillée de ces dernières. Durant la deuxième phase, se déroulant pendant le deuxième quadrimestre de l'année académique 2017-2018, l'évaluation du premier prototype a mené à un changement de métaphore de visualisation pour laquelle les interactions gestuelles du casque HoloLens sont étendues, par le biais d'une preuve de concept, pour mettre en oeuvre les différentes tâches nécessaires à l'utilisation du second prototype.

Les différentes contributions de ce mémoire sont les suivantes :

— Visualisation des éléments d'analyse structurale sous la forme de mobiles artistiques à savoir les relations de disjonction et les structures parallèles, hiérarchiques et croisées qu'elles peuvent former.

— Définition d'un ensemble de gestes destinés à la manipulation de la visualisation de mobiles basés sur les gestes adaptés à l'AR présentés dans (Piumsomboon *et al.*, 2013) et (Wang *et al.*, 2011).

— Deux architectures logicielles autorisant l'extension des capacités de détection de gestes statiques et dynamiques du casque HoloLens ainsi que l'ajout d'interactions directes basées sur une présence physique des mains au sein du monde virtuel grâce à la seconde architecture proposée.

— Un système extensible de création dynamique de trajectoires permettant leur définition selon une formulation mathématique, mais également la décoration de ces dernières en générant un modèle 3D correspondant à leurs définitions mathématiques.

— Un système d'encapsulation de fonctionnalités qui facilite leur développement et leur gestion à l'exécution tout en réduisant le couplage entre des fonctionnalités distinctes.

En plus des contributions présentées et l'étude réalisée sur cette technologie, de nombreuses améliorations ou recherches futures sont à prévoir. Tout d'abord, une évaluation auprès des praticiens de l'analyse structurale doit être réalisée afin d'obtenir un ensemble d'avis des experts du domaine auxquels la visualisation s'adresse. Dans cette évaluation, les interactions gestuelles doivent également être évaluées. Ensuite, lors de l'élaboration des gestes supplémentaires destinés aux interactions manquantes de la visualisation de mobiles, les travaux de Piumsomboon *et al.* ont servi de références. Dans ces derniers, diverses observations et lignes directrices sont présentées, par conséquent, ces observations peuvent être intégrées dans la visualisation proposée. De plus, avec la preuve de concept réalisée, la reconnaissance de geste est un processus complexe régi par de nombreux paramètres. Dès lors, un système d'apprentissage automatisé représente un travail futur pouvant grandement faciliter l'intégration de nouveaux gestes pour le casque HoloLens. Par ailleurs, les architectures d'extension des interactions gestuelles du casque peuvent être mises en oeuvre. Enfin, des méthodes d'interactions alternatives telles que les interactions multimodales ou tangibles peuvent être étudiées. En effet, le casque HoloLens est capable de reconnaitre la voix de l'utilisateur, mais également d'appliquer des techniques de Computer Vision grâce au flux vidéo accessible.

Bibliographie

Azuma, R. (1997). A Survey of Augmented Reality. *Presence*, 6:355–385.

Bach, B., Sicat, R., Beyer, J., Cordeil, M. et Pfister, H. (2018). The Hologram in My Hand : How Effective is Interactive Exploration of 3D Visualizations in Immersive Tangible Augmented Reality ? *IEEE Transactions on Visualization and Computer Graphics*, 24(1):457–467.

Bai, H., Lee, G. et Billinghurst, M. (2015). Free-hand Gesture Interfaces for an Augmented Exhibition Podium. *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction on - OzCHI '15*, pages 182–186.

Belcher, D., Billinghurst, M., Hayes, S. E. et Stiles, R. (2003). Using augmented reality for visualizing complex graphs in three dimensions. *Proceedings - 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR 2003*, pages 84–93.

Billinghurst, M., Clark, A. et Lee, G. (2015). A Survey of Augmented Reality. *Foundations and Trends® in Human–Computer Interaction*, 8(2-3):73–272.

Bolt, R. A. (1980). Put-that-there : Voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.*, 14(3):262–270.

Brath, R. (2014). 3D InfoVis is here to stay : Deal with it. *2014 IEEE VIS International Workshop on 3DVis, 3DVis 2014*, pages 25–31.

Buchmann, V., Violich, S., Billinghurst, M. et Cockburn, A. (2004). FingARtips : gesture based direct manipulation in Augmented Reality. *Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 212–221.

Clarinval, A. (2017). STAVIZ : Application of Text Visualization Techniques to Structural Analysis. Mémoire de D.E.A., Université de Namur.

Colaner, S. (2016). What's Inside Microsoft's HoloLens And How It Works. http://www.tomshardware.com/news/microsoft-hololens-components-hpu-28nm,32546.html. Consulté le 27/03/2018.

Corbett-Davies, S., Dunser, A., Green, R. et Clark, A. (2013). An advanced interaction framework for augmented reality based exposure treatment. *2013 IEEE Virtual Reality (VR)*, (March 2013):19–22.

Garon, M., Boulet, P. O., Doironz, J. P., Beaulieu, L. et Lalonde, J. F. (2016). Real-time high resolution 3d data on the hololens. *In 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pages 189–191.

Hempel, J. (2015a). Microsoft in the age of Satya Nadella. https://www.wired.com/2015/01/microsoft-nadella/.

Hempel, J. (2015b). Project HoloLens : Our exclusive hands-on with Microsoft's holographic goggles. https://www.wired.com/2015/01/microsoft-hands-on/.

Hilliges, O., Kim, Izadi, S., Weiss, M. et Wilson, A. (2012). Holodesk : Direct 3d interactions with a situated see-through display. *Proceedings of CHI 2012*, pages 2421–2430.

Karam, M. et Schraefel, M. C. (2005). A taxonomy of gestures in human computer interactions.

Kendon, A. (2004). *Gesture : Visible Action as Utterance*. Gesture : Visible Action as Utterance. Cambridge University Press.

Kumar, P., Verma, J. et Prasad, S. (2012). Hand Data Glove : A Wearable Real-Time Device for Human-Computer Interaction. *International Journal of Advanced Science and Technology*, 43:15–26.

LeapMotion (2013). Designing the Leap Motion Controller. http://blog.leapmotion.com/designing-leap-motion-controller/. Consulté le 06/05/2018.

LeapMotion (2014). How Does the Leap Motion Controller Work ? http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/. Consulté le 06/05/2018.

LeapMotion (2016). API Overview. https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html. Consulté le 06/05/2018.

Lee, J. Y., Rhee, G. W. et Seo, D. W. (2010). Hand gesture-based tangible interactions for manipulating virtual objects in a mixed reality environment. *International Journal of Advanced Manufacturing Technology*, 51(9-12):1069–1082.

Lee, T. et Höllerer, T. (2007). Handy AR : Markerless inspection of augmented reality objects using fingertip tracking. *Proceedings - International Symposium on Wearable Computers, ISWC*, 2007:83–90.

Mäntyjärvi, J., Kela, J., Korpipää, P. et Kallio, S. (2004). Enabling fast and effortless customisation in accelerometer based gesture interaction. *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia - MUM '04*, (November 2014):25–31.

McNeill, D. (1992). *Hand and Mind : What Gestures Reveal about Thought*. University of Chicago Press.

Meiguins, B. S., do Carmo, R. M. C., Almeida, L., Gonçalves, A. S. o., Pinheiro, S. C. V., de Brito Garcia, M. et Godinho, P. I. A. (2006). Multidimensional information visualization using augmented reality. *In Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications*, VRCIA '06, pages 391–394, New York, NY, USA. ACM.

Microsoft. Cursors. https://developer.microsoft.com/en-us/windows/mixed-reality/cursors. Consulté le 10/03/2018.

Microsoft. Gaze targeting. https://developer.microsoft.com/en-us/windows/mixed-reality/gaze_targeting. Consulté le 10/03/2018.

Microsoft. Gestures. https://developer.microsoft.com/en-us/windows/mixed-reality/gestures. Consulté le 14/03/2018.

Microsoft. Microsoft HoloLens. https://www.microsoft.com/en-us/hololens. Consulté le 24/02/2018.

Microsoft (2016). Using the Windows Device Portal. https://developer.microsoft.com/en-us/windows/mixed-reality/using_the_windows_device_portal. Consulté le 24/02/2018.

Microsoft (2017). A medical revolution. https://news.microsoft.com/europe/features/a-medical-revolution/. Consulté le 09/05/2018.

Microsoft (2018a). App views. https://docs.microsoft.com/en-us/windows/mixed-reality/app-views. Consulté le 04/04/2018.

Microsoft (2018b). Development overview : Tools for developing for mixed reality. https://docs.microsoft.com/en-us/windows/mixed-reality/development-overview#tools-for-developing-for-mixed-reality. Consulté le 02/04/2018.

Microsoft (2018c). DirectX development overview : Adding mixed reality capabilities and inputs. https://docs.microsoft.com/en-us/windows/mixed-reality/directx-development-overview#adding-mixed-reality-capabilities-and-inputs. Consulté le 02/04/2018.

Microsoft (2018d). DirectX development overview : Getting started. https://docs.microsoft.com/en-us/windows/mixed-reality/directx-development-overview#getting-started. Consulté le 02/04/2018.

Microsoft (2018e). Gaze in Unity. https://docs.microsoft.com/en-us/windows/mixed-reality/gaze-in-unity. Consulté le 02/04/2018.

Microsoft (2018f). Gestures and motion controllers in Unity : High-level composite gesture APIs (GestureRecognizer). https://docs.microsoft.com/en-us/windows/mixed-reality/gestures-and-motion-controllers-in-unity#high-level-composite-gesture-apis-gesturerecognizer. Consulté le 02/04/2018.

Microsoft (2018g). HoloLens hardware details. https://docs.microsoft.com/fr-be/windows/mixed-reality/hololens-hardware-details. Consulté le 27/03/2018.

Microsoft (2018h). Performance recommendations for HoloLens apps : Content guidance. https://docs.microsoft.com/en-us/windows/mixed-reality/performance-recommendations-for-hololens-apps#content-guidance. Consulté le 30/03/2018.

Microsoft (2018i). Spatial mapping in Unity : Using the low-level Unity Spatial Mapping API. https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-mapping-in-unity#using-the-low-level-unity-spatial-mapping-api. Consulté le 02/04/2018.

Microsoft (2018j). Unity development overview : Adding mixed reality capabilities and inputs. https://docs.microsoft.com/en-us/windows/mixed-reality/unity-development-overview#adding-mixed-reality-capabilities-and-inputs. Consulté le 02/04/2018.

Microsoft (2018k). Updating 2D UWP apps for mixed reality : New input possibilities. https://docs.microsoft.com/en-us/windows/mixed-reality/building-2d-apps#new-input-possibilities. Consulté le 02/04/2018.

Microsoft (2018l). Voice input : Dictation. https://docs.microsoft.com/fr-be/windows/mixed-reality/voice-input#dictation. Consulté le 24/03/2018.

Microsoft (2018m). Voice input : Hey Cortana. https://docs.microsoft.com/fr-be/windows/mixed-reality/voice-input#hey-cortana. Consulté le 24/03/2018.

Microsoft (2018n). Voice input in Unity. https://docs.microsoft.com/en-us/windows/mixed-reality/voice-input-in-unity. Consulté le 02/04/2018.

Microsoft (2018o). Voice input : "See It, Say It". https://docs.microsoft.com/fr-be/windows/mixed-reality/voice-input#see-it-say-it. Consulté le 23/03/2018.

Microsoft (2018p). Voice input : The "select" command. https://docs.microsoft.com/fr-be/windows/mixed-reality/voice-input#the-select-command. Consulté le 24/03/2018.

Microsoft (2018q). Voice input : Voice commands for fast Hologram Manipulation. https://docs.microsoft.com/fr-be/windows/mixed-reality/voice-input#voice-commands-for-fast-hologram-manipulation. Consulté le 24/03/2018.

Milgram, P. et Kishino, F. (1994). A Taxonomy of Mixed Reality Visual-Displays. *Ieice Transactions on Information and Systems*, E77d(12):1321–1329.

Munzner, T. (2014). *Visualization Analysis and Design*, chapitre 6, pages 119, 124, 129. AK Peters Visualization Series. CRC Press.

Pattison, T., Vernik, R., Goodburn, D. et Phillips, M. (2001). Rapid assembly and deployment of domain visualisation solutions. *In Australian symposium on Information visualisation*, volume 9, pages 19–26.

Piret, A., Nizet, J. et Bourgeois, E. (1996). *L'analyse structurale : une méthode d'analyse de contenu pour les sciences humaines*. De Boeck Supérieur.

Piumsomboon, T., Altimira, D., Kim, H., Clark, A., Lee, G. et Billinghurst, M. (2014). Grasp-Shell vs gesture-speech : A comparison of direct and indirect natural interaction techniques in augmented reality. *ISMAR 2014 - IEEE International Symposium on Mixed and Augmented Reality - Science and Technology 2014, Proceedings*, pages 73–82.

Piumsomboon, T., Clark, A., Billinghurst, M. et Cockburn, A. (2013). User-defined gestures for augmented reality. *In CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 955–960. ACM.

Posada, J., Toro, C., Barandiaran, I., Oyarzun, D., Stricker, D., Amicis, R., Pinto, E., Eisert, P., Döllner, J. et Vallarino, I. (2015). Visual Computing as Key Enabling Technology for Industry 4.0 & Industrial Internet. *IEEE Computer Graphics and Applications*, 35(2):26–40.

Quek, F., McNeill, D., Bryll, R., Duncan, S., Ma, X.-F., Kirbas, C., McCullough, K. E. et Ansari, R. (2002). Multimodal human discourse : gesture and speech. *ACM Transactions on Computer-Human Interaction*, 9(3):171–193.

Rekimoto, J. et Nagao, K. (1995). The World through the Computer : Computer Augmented Interaction with Real World Environments. *Proceedings of the 8th annual ACM symposium on User interface and software technology*, pages 29–36.

Slay, H., Phillips, M., Vernik, R. et Thomas, B. (2001). Interaction Modes for Augmented Reality Visualization. *Australian Symposium on Information Visualisation. Conferences in Research and Practice in Information Technology*, 9(December):71–75.

St. John, M., Cowen, M. B., Smallman, H. S. et Oonk, H. M. (2001). The Use of 2D and 3D Displays for Shape-Understanding versus Relative-Position Tasks. *Human Factors : The Journal of the Human Factors and Ergonomics Society*, 43(1):79–98.

Tate (2008). 'Mobile', Alexander Calder, c.1932 | Tate . http://www.tate.org.uk/art/artworks/calder-mobile-l01686. Consulté le 20/04/2018.

Tuliper, A. (2017). HoloLens - Introduction to the HoloLens, Part 2 : Spatial Mapping. https://msdn.microsoft.com/en-us/magazine/mt745096.aspx. Consulté le 25/02/2018.

Unity (2017a). Manual : Event System. https://docs.unity3d.com/Manual/EventSystem.html. Consulté le 05/05/2018.

Unity (2017b). Manual : The Multiplayer High Level API. https://docs.unity3d.com/Manual/UNetUsingHLAPI.html. Consulté le 05/05/2018.

Unity (2017c). Spatial Mapping components. https://docs.unity3d.com/Manual/windowsholographic-sm-component.html. Consulté le 02/04/2018.

Van den Bergh, M. et Van Gool, L. (2011). Combining RGB and ToF Cameras for Real-time 3D Hand Gesture Interaction. pages 66–72. IEEE Computer Society.

Wang, B. et Mueller, K. (2015). Does 3D really make sense for visual cluster analysis ? yes ! *2014 IEEE VIS International Workshop on 3DVis, 3DVis 2014*, pages 37–44.

Wang, R., Paris, S. et Popović, J. (2011). 6D Hands : Markerless Hand-Tracking for Computer Aided Design. *Proceedings of UIST 2011*, pages 549–557.

Wang, R. Y. et Popović, J. (2009). Real-time hand-tracking with a color glove. *ACM SIGGRAPH 2009 papers on - SIGGRAPH '09*, page 1.

Weissmann, J. et Salomon, R. (1999). Gesture recognition for virtual reality applications using data gloves and neural networks. *In Neural Networks, 1999. IJCNN '99. International Joint Conference on*, volume 3, pages 2043–2046 vol.3.

Wexelblat, A. (1998). Research challenges in gesture : Open issues and unsolved problems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1371:1–11.

Wobbrock, J. O., Morris, M. R. et Wilson, A. D. (2009). User-defined gestures for surface computing. *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*, page 1083.

Zhou, F., Duh, H. B.-L. et Billinghurst, M. (2008). Trends in augmented reality tracking, interaction and display : A review of ten years of ismar. *In Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 193–202. IEEE Computer Society.