
Projet de fin d'études : classification de documents papiers à l'aide de la réalité mixte

Étudiants :

Arnaud DELPEYROUX
Jimmy GOURAUD
Thibaut MONSEIGNE
Nicolas PALARD

Encadrants :

Marie BEURTON-AIMAR
Nicholas JOURNET



Université de Bordeaux UF Informatique
Master Informatique mention informatique, spécialisation Image et Son

25 mars 2018

Résumé

Dans le cadre du projet de fin d'études, il nous a été demandé d'étudier l'intérêt que peut avoir l'utilisation de la réalité mixte dans une tâche de classification de documents papier. Pour contextualiser notre sujet, nous avons choisi de nous intéresser à l'impact que peut avoir une telle technologie dans un contexte de gestion électronique de documents (GED). En effet, durant ce projet nous avons mis en place une version simplifiée et tronquée d'une partie de la chaîne de traitement de la GED à savoir la préparation, la numérisation et l'ajout de métadonnées sur un document. Nous proposons ainsi une interface homme-machine (IHM) en réalité mixte permettant d'agir dans chacune des étapes de la chaîne simulée pour pouvoir démontrer l'intérêt que peut avoir une telle technologie. Notre interface s'appuie sur des algorithmes de traitement d'images comme la détection de document pour construire une expérience de GED innovante où le document physique est placé au cœur des interactions.

Le projet a été réalisé avec comme support technologique les lunettes de réalité mixte **Microsoft HoloLens** [16] et le moteur de jeu **Unity** [32]. Dans ce rapport, nous montrons qu'il est possible que la réalité mixte ait un réel impact dans une chaîne n'évoluant plus depuis quelques années, mais nous expliquons aussi les limites de cet impact au vu de l'évolution actuelle de la technologie avec, par exemple, du nombre réduit des interactions et gestes disponibles.

Table des matières

1 Contexte	3
1.1 Définition des réalités	3
1.1.1 La réalité virtuelle	3
1.1.2 La réalité augmentée	3
1.1.3 La réalité mixte	4
1.2 État de l'art	4
1.3 Problématique	5
2 Cahier des charges	6
2.1 Notions	6
2.1.1 Document papier	6
2.1.2 Fonctionnalité dynamique	6
2.2 Besoins fonctionnels	7
2.3 Besoins non fonctionnels	9
2.4 Contraintes	9
2.5 Jeu de tests	10
2.6 Cas d'utilisations et scénario	10
3 Solution	14
3.1 Architecture	14
3.1.1 HoloLens	14
3.1.2 Serveur	15
3.2 Interface homme machine (IHM)	16
3.2.1 Spécificité de l'HoloLens	17
3.2.2 Numérisation des documents	18
3.2.3 Métadonnées des documents	20
3.2.4 Lien entre les documents	21
3.2.5 Problèmes rencontrés et solutions apportées	22
3.3 Détection de Document	24
3.3.1 Binarisation de l'image	24
3.3.2 Extraction des documents	25
3.4 Reconnaissance de Document	27
3.4.1 Méthode	27

3.4.2	Comparaison	29
3.4.3	Perspective	30
3.5	Base de données	31
3.6	Serveur web	32
4	Tests	35
4.1	Interface Homme-Machine	35
4.1.1	Conclusion	39
4.2	Traitement d'image	39
4.2.1	Binarisation	40
4.2.2	Extraction des documents	43
4.2.3	Conclusion	44
4.3	Base de données	45
4.4	Serveur	46
Bibliographie		53
A	Jeu de Test	54
B	Détection de Document	57
C	Protocole d'installation	60
C.1	Côté Serveur	60
C.2	Côté Développeur	60
D	Questionnaire Utilisateur	62
E	Gantt prévisionnel	65
F	Gantt réel	66

Introduction

La gestion des documents ainsi que leur sauvegarde représentent un point important pour différents types d'organismes. En effet, que ce soit des entreprises privées ou publiques, gouvernementales ou non, la gestion des documents est une fonction essentielle pour le bon fonctionnement de l'entreprise. Il faut pouvoir trier l'ensemble des éléments utilisés dans ces structures afin de traiter chaque papier de la meilleure manière possible. Or, c'est une problématique difficile, car c'est une tâche ni automatique ni systématique. C'est aussi une opération fastidieuse qui demande une intervention humaine, car requérant du discernement et parfois des jugements subjectifs. Enfin, la complexité de la tâche va en augmentant du fait de l'accroissement continu du nombre de documents à traiter.

Il existe des méthodes pour aider les personnes à la gestion de documents. Ces moyens sont souvent organisationnels : ils se basent sur l'attribution d'une valeur à chaque élément et à l'application d'une méthode visant à traiter chaque papier selon sa valeur. Néanmoins, le point-clé de cette méthodologie étant l'attribution d'une valeur au document, l'utilisation de processus automatique n'est pas envisagable. En effet, c'est une tâche subjective effectuée par un utilisateur compétent. On pourrait par exemple utiliser des systèmes de classification afin d'attribuer ces valeurs, mais cette attribution répond à trop de critères et il serait difficile de mettre en place un système robuste à tous les cas pouvant se présenter à lui. On comprend donc très facilement que cette tâche ne peut se passer de l'intervention humaine. De plus, de tels systèmes demandent à l'utilisateur de permute entre la manipulation physique du document et l'utilisation d'un terminal spécialisé dans la gestion de documents.

Ainsi une réponse technologique possible serait de ne pas remplacer l'humain dans sa tâche, mais de la lui faciliter. De plus, du fait de l'aspect matériel du problème il apparaît que la réalité augmentée s'avérerait être un choix judicieux quant aux possibilités qu'elle offre. En effet, au regard de l'évolution de cette technologie dans les dernières années, la réalité mixte possède tous les atouts espérés pour une telle tâche. Cette dernière permet d'obtenir une fusion entre le monde réel et virtuel où tout objet physique du

monde réel et tout objet numérique du monde virtuel peuvent coexister et interagir en temps réel. La question qu'on peut donc se poser ici est de savoir si l'utilisation d'une telle technologie peut apporter une solution au problème de la gestion de document et plus précisément si l'utilisation particulière d'un système de réalité mixte de type **HoloLens** [16] convient à ce type de tâches.

L'idée de notre projet est donc d'offrir une façon d'assister un opérateur dans sa tâche de classification de documents. Ce projet est très prospectif, avant-coureur et n'apporte donc aucune réelle solution, mais plutôt des pistes à explorer quant à cette problématique. Nous proposons donc une nouvelle forme d'interface homme-machine (IHM) permettant diverses actions liées à la classification de documents. Au-delà des fonctionnalités proposées par notre IHM, nous avons vraiment voulu travailler sur l'aspect ergonomique pour pouvoir créer une expérience fluide et facile à prendre en main.

Dans un premier temps, nous contextualiserons notre sujet en définissant les différents types de réalité (virtuelle, mixte et augmentée) qui existe, et nous ferons un état de l'art sur les différentes parties de notre sujet à savoir la GED, la détection de document et la réalité mixte. La seconde partie de notre rapport portera sur notre cahier des charges ainsi que les besoins fonctionnels et non fonctionnels. Une partie "Solution" mettra en avant les détails notre projet, comme l'architecture, les problèmes rencontrés, les solutions apportées et les différentes fonctionnalités présentes. Enfin, le chapitre "Test" permettra de présenter les différents tests effectués au cours du PFE.

Chapitre 1

Contexte

1.1 Définition des réalités

Avant de rentrer dans le vif du sujet, il nous semble important de définir et de distinguer les trois types de réalités du domaine informatique, à savoir :

- la réalité virtuelle
- la réalité augmentée
- la réalité mixte

1.1.1 La réalité virtuelle

La réalité virtuelle (VR : Virtual Reality) consiste à créer un monde complètement virtuel, réaliste ou non, sans interaction possible avec le monde réel environnant. L'utilisateur, équipé d'un casque de réalité virtuelle, est immergé dans ce monde virtuel et ne peut interagir qu'avec celui-ci. Dans le cas de notre sujet, nous avons besoin de manipuler des documents physiques et ne pouvons donc pas utiliser ce type de réalité qui coupe tous liens avec le monde réel.

1.1.2 La réalité augmentée

Contrairement à la réalité virtuelle, la réalité augmentée rajoute simplement des informations superposées au monde réel, à l'aide d'hologrammes. On n'est donc pas immergé dans un monde complètement virtuel, mais des objets virtuels viennent se rajouter à notre vision réelle du monde, nous permettant d'acquérir plus d'informations que le monde réel nous en fournit. On "augmente" ainsi notre vision de la réalité. Par abus de langage, la réalité augmentée englobe souvent le terme de réalité mixte, qui est détaillé dans la partie suivante.

1.1.3 La réalité mixte

La réalité mixte (MR : Mixed Reality) ou réalité hybride (xR : Cross Reality) est la fusion du monde réel et d'un monde virtuel. Ce mélange produit un nouvel environnement et des visualisations où les objets physiques du monde réel et numérique du monde virtuel coexistent et peuvent interagir en temps réel. Grâce à une technologie immersive, un monde nouveau mêlant le réel et le virtuel est donc créé. À la différence de la réalité augmentée, la réalité mixte n'ajoute pas seulement des objets de synthèse dans l'environnement réel sous la forme d'un hologramme, mais permet aussi à l'utilisateur d'interagir avec. Un dispositif de type lunettes est indispensable pour la réalité mixte.

1.2 État de l'art

Le domaine de la gestion électronique de document est vaste. En effet, il existe de nombreuses problématiques au sein de ce domaine. Dans notre cas, nous nous intéresserons aux méthodes d'acquisition, et de labellisations de documents. En effet, nous étudions les cas où un utilisateur manipule physiquement des documents et où il cherche à obtenir ou à apposer des informations à ces documents, de ce fait toute manipulation uniquement électronique de documents est à exclure (gestions d'e-mail, pages web, etc.). Il existe des méthodes permettant de répondre à cette problématique. Aussi, l'apport d'une visualisation 3D a déjà été étudié [7] et cela montre qu'une visualisation 3D des documents est plus appréciée par les utilisateurs. Certaines de ces méthodes proposent de prendre en photo un ou plusieurs documents, qui seront labellisés automatiquement à l'aide d'OCR ou de classifieur puis proposeront à l'utilisateur de consulter ou modifier les informations des différents documents. Néanmoins, cela entraîne une alternance entre deux tâches. La première est la manipulation physique du document suivi par sa capture. Ensuite vient le moment d'utiliser un terminal informatique afin de consulter et modifier au besoin les informations. Or, une telle alternance de tâches a des impacts négatifs sur les performances et la précision de l'opérateur [1]. De plus, cette alternance amplifie la nature déjà répétitive et fastidieuse de ce genre d'opération.

La réalité augmentée et réalité mixte sont deux domaines en pleine expansion. En effet, ce sont des applications qui émergent au public avec des appareils comme l'**HoloLens** [16] de Microsoft récemment ou les **GoogleGlass** plusieurs années auparavant. Ces technologies permettant d'augmenter la réalité en superposant des informations supplémentaires à celle-ci. À part ces nombreux avantages et apports, la réalité augmentée est couramment utilisée pour divers domaines tels que la médecine [33], les jeux [25], l'assemblage de pièces [36] ou même la conception [9]. Malgré les nombreuses utilisations déjà existantes, il est difficile d'appréhender l'ensemble des possibilités offertes

par de telles technologies. De plus, l'état des nous permet aujourd'hui de créer des dispositifs accessibles et performants nous permettant d'explorer le champ des possibles. Néanmoins, les utilisations existantes ont déjà permis de mettre en évidence certaines limitations des outils de réalité augmentée ou mixte. L'enjeu de ses technologies est donc de proposer des interfaces homme-machine suffisamment ergonomique et intuitive pour améliorer l'expérience de l'opérateur.

Avant de nous plonger dans l'implémentation de notre sujet, nous avons cherché un maximum d'informations sur les différents outils auxquels nous allions avoir besoin au cours de notre PFE. Ainsi, nous nous sommes renseignés sur les différentes méthodes de détections de documents [5] [8] [13] [21] [22] [23] [24] [27] [28] [29] [35] afin d'avoir une vision globale assez globale sur celle-ci et pour pouvoir cibler les mieux adaptés à notre cas. Nous avons aussi chercher des méthodes concernant les déformations de documents [6] [34] [37] afin de les remettre à plat et nous avons aussi chercher des menus interactifs en réalité augmentée [12] [18] [31].

1.3 Problématique

Comme énoncé précédemment, notre sujet est principalement prospectif et nous ne sommes pas là pour apporter une réponse avec une solution précise à un sujet donné. L'objectif principal est d'avoir une réflexion sur ce que la réalité augmentée/mixte permet d'apporter dans la chaîne de traitement de la gestion électronique de document (GED). En effet, la GED a peu évolué au cours des 30 dernières années et la réalité augmentée/mixte est une technologie relativement nouvelle dans le sens où elle commence à être suffisamment mature pour être utilisé à grande échelle depuis quelques années seulement.

Chapitre 2

Cahier des charges

2.1 Notions

Afin de mieux comprendre notre projet, certaines notions doivent être définies au préalable, comme la notion de "document papier" et de "fonctionnalité dynamique".

2.1.1 Document papier

Tout d'abord, il faut définir ce qu'est un document papier. Dans notre cas, cela n'est pas défini précisément et cela peut être : une page de roman, un manuscrit, un papier administratif (facture, fiche de paye...), la photo d'un objet, la réplique d'un tableau, un CV, etc. Tout dépend de l'utilisateur final. Si nous prenons l'exemple d'un conservateur de musée, il peut avoir sous les yeux d'anciens parchemins ainsi que des tableaux. Une autre personne peut vouloir classer des CV, des factures ou d'autres papiers administratifs.

Dans le cas de notre application, on prendra en charge des documents complets (nous ne gérerons pas des fragments de documents) de différentes tailles. Ces documents seront principalement des impressions d'images ou de textes ou bien des feuilles de couleurs unies.

2.1.2 Fonctionnalité dynamique

Une fonctionnalité est dite dynamique si l'évaluation de celle-ci ne nécessite pas de demandes de l'utilisateur. C'est-à-dire, si la fonctionnalité est évaluée en continu par l'application. Par exemple, la mise au point d'un appareil photographique en mode automatique est dynamique, elle s'effectue automatiquement au cours du temps selon le flux d'images en entrée. Par extension, une fonctionnalité non dynamique demande une action ou plusieurs actions pour être évaluée.

2.2 Besoins fonctionnels

Numériser des documents - Le système doit être capable d'accéder à la caméra physique des lunettes et d'en extraire l'image courante. Une fois l'extraction terminée, le système doit sauvegarder cette image pour une future utilisation. Pour accéder à la caméra physique, le système utilisera l'API fourni par la surcouche système de la plateforme de développement.

Créer des objets représentant les informations d'un document - Le système devra, lors de la numérisation d'un document, créer des objets permettant de stocker les informations dudit document.

Les informations devant être stockées sont les suivantes :

- Le label : information élémentaire traduisant la catégorie à laquelle appartient ce document.
- L'auteur : information secondaire traduisant à qui appartient le document ou qui l'a numérisé.
- La date : information secondaire traduisant soit la date de capture, soit la date de création du document.
- La description : information secondaire permettant de sauvegarder plus d'informations sur ce document.
- Les liens : information élémentaire traduisant les liens que ce document possède.

L'objet devra être sérialisable pour pouvoir être stocké dans une base de documents.

Visualiser les informations d'un document - L'interface du système devra proposer une visualisation des informations des documents.

La visualisation des informations sera composée de plusieurs hologrammes (objets 3D) agencés ensemble.

Editer les informations d'un document - En utilisant l'interface de visualisation, le système devra aussi proposer des méthodes permettant d'éditionner les informations. Le système devra donc être capable de traiter des demandes d'éditions et d'y répondre en proposant des méthodes cohérentes dans le cadre de l'application.

Proposer une méthode de saisie virtuelle - Dans le cas où une saisie utilisateur serait nécessaire, le système doit être capable de fournir une interface adaptée avec laquelle l'utilisateur pourra interagir pour saisir des informations.

Créer une base de données permettant de sauvegarder les documents

- La solution devra proposer une architecture de base permettant de stocker les documents numérisés. Le système devra donc proposer des

méthodes d'enregistrement de données dans la base ainsi que des méthodes de récupération et mise à jour de ces données.

Ces opérations devront s'effectuer en temps raisonnable pour assurer le bon fonctionnement du système.

Déetecter un document dans une image - Le système doit être capable, à partir d'une photographie acquise via les lunettes, de détecter la présence d'un document dans cette dernière. Cette étape doit s'effectuer automatiquement après chaque prise de photographies. Si aucun document n'est trouvé dans l'image, le système ne continuera pas les traitements d'image et conservera la photographie d'origine pour la suite de son fonctionnement. Cette étape est le premier maillon de la chaîne de traitement d'image. La chaîne de traitement doit s'exécuter dans son entièreté en pseudo temps réel pour assurer la réactivité et la fluidité générale du système. Cette étape devra donc assurer des performances suffisantes pour respecter cette contrainte. Dans le cas où cette contrainte n'est pas respectée, ces calculs pourront être déportés hors des lunettes.

Extraire un document dans une image - Le système devra, après détection d'un document, extraire ce dernier de l'image. L'extraction de celui-ci se fera à l'aide de la boîte englobante. Cette étape doit donc s'effectuer automatiquement après la détection du document, si un document a bel et bien été détecté. L'extraction d'un document est la seconde étape de la chaîne de traitement d'image et doit assurer des performances suffisantes pour respecter une contrainte de pseudo temps réel sur toute la chaîne.

Redresser un document - Après extraction d'un document, le logiciel devra être capable de corriger les distorsions géométriques liées à l'optique de l'appareil photo. En effet, selon l'angle de vue, la position, et l'état du document physique la perspective peut le faire apparaître déformé. Il faut donc pouvoir appliquer un traitement permettant d'annuler ces distorsions afin de pouvoir obtenir un document non déformé avant de le stocker dans la base. Ceci doit s'effectuer automatiquement après extraction du document et permettra d'uniformiser tous les documents dans la base. Cette étape est la troisième partie de la chaîne de traitement d'image et de ce fait devra avoir des performances suffisantes pour que la chaîne de traitement complète puisse se faire en pseudo temps réel.

Analyser et identifier un document - À l'aide des documents stockés dans la base, le système doit être capable d'identifier des documents. Pour ce faire, le système devra aussi être capable d'extraire des données caractéristiques des documents avant leur enregistrement dans la base. Ces données caractéristiques devront être définies en prenant en

compte le type de document que le système devra analyser à savoir des documents papier comme définis plus haut. L'identification des documents est la dernière étape de la chaîne de traitement d'image, elle devra donc aussi respecter une contrainte de pseudo temps réel pour assurer la réactivité générale du système.

2.3 Besoins non fonctionnels

Ergonomie - Le logiciel doit être ergonomique. La notion d'ergonomie suppose ici que le logiciel doit pouvoir être utilisé de façon intuitive pour un opérateur dont la connaissance en réalité mixte est limitée. Ainsi, l'utilisateur final doit pouvoir utiliser l'ensemble des fonctionnalités avec un nombre d'actions minimales (nombre maximal d'actions utilisateur pour accéder à une fonctionnalité). Aussi, l'accès aux fonctionnalités doit être rapide (temps maximal d'accès à une fonctionnalité).

Pseudo temps réel - Pour avoir une fluidité visuelle suffisante et ne pas donner une impression de saccade ni de générer des artefacts visuels, il est nécessaire de synchroniser l'affichage avec la fréquence de rafraîchissement de l'écran qui est de 30Hz ainsi le système doit avoir un taux de rafraîchissement supérieur ou égal à 30 images par secondes sur les lunettes de réalité mixte HoloLens [16]. L'ensemble des fonctionnalités dynamiques doivent être calculées et rendues en moins de 1/30 seconde (33,3 millisecondes).

Saisie virtuelle adaptée - Le système devra proposer une méthode de saisie virtuelle pour que l'utilisateur puisse être capable d'avoir un rythme de saisie dépassant les cinq mots par minutes. Ce besoin a pour but d'assurer que la saisie d'informations sur les documents ne soit pas une contrainte pour l'utilisateur.

La solution doit pouvoir évoluer - Afin d'assurer la maintenabilité et l'évolution de l'application, les bibliothèques externes ainsi que les technologies utilisées devront être les plus récentes possible.

2.4 Contraintes

Plusieurs contraintes matérielles et logicielles ont été imposées par le sujet :

- Le moteur de jeu Unity [32]
- Le langage de programmation C#
- Le casque de réalité mixte HoloLens [16]

2.5 Jeu de tests

Comme mentionné dans les besoins, il est nécessaire d'avoir une base de document pour pouvoir comparer nos nouveaux documents avec ceux présents dans celle-ci. Pour cela, il va donc être nécessaire de constituer des jeux de tests composés de documents types pour pouvoir remplir cette base d'images. La construction du jeu de test peut se faire de deux façons : soit en numérisant des documents réels, soit en générant des documents fictifs. Cette méthode de génération de donnée importe peu, mais il faut que chacun de ces documents soit annoté avec les informations nécessaires définies plus haut. Aussi, pour chacun de ces documents, il faut évaluer les caractéristiques nécessaires à la recherche et l'identification d'un document et les stocker.

La partie classification et reconnaissance de documents n'étant pas le sujet principal du projet, nous utiliserons pour nos tests 10 documents fortement discriminants.

Le jeu de test choisi (voir Annexe A) possède plusieurs particularités permettant de mettre à l'épreuve nos algorithmes de détection de documents. Pour les pages de bande dessinée ou de manga comportant des cases, par exemple, il faudra détecter l'ensemble de la page et non seulement une case composant celle-ci. De même, les pages d'articles scientifiques, formatées sur deux colonnes, ne devront pas être séparées. Ces deux pages sont d'ailleurs similaires à la différence que l'une d'entre elles possède plusieurs figures. La reconnaissance de document, bien que basique, aura un défi à relever pour les différencier. Nous avons deux photos pleine page, une en A5 et l'autre en A4 (une fois imprimé, un encadrement blanc contourne l'image). Les feuilles unies se différencient essentiellement par leur couleur et leur taille.

2.6 Cas d'utilisations et scénario

Sélection des utilisateurs - Plusieurs types d'utilisateurs interviendront pour tester l'application, les quatre membres du groupe, évidemment, seront les premiers alpha testeurs. Ensuite, nous essayerons de faire tester l'application à une dizaine de personnes extérieures au projet ayant des connaissances en informatiques et ayant déjà utilisé des systèmes de réalité virtuelle ou augmentée. Pour finir, nous ferons appel à une dizaine de personnes extérieures n'ayant aucune connaissance particulière dans ce domaine ou dans l'informatique en général.

Conditions d'utilisations - L'application est conçue pour fonctionner dans un environnement possédant un éclairage suffisant pour la prise de photographies nettes sans flash. L'utilisateur est assis ou debout face à son bureau avec les documents présents dans son espace de travail. Les

bureaux unis et possédant un contraste avec la couleur des documents sont privilégiés et les blancs sont donc généralement à éviter. Ces différentes contraintes d'utilisations pourront évoluer en fonction de la robustesse de la détection de document et de la malléabilité des interfaces sur l'HoloLens.

Cas d'utilisation 1 - Test d'ergonomie de l'application

Acteur(s) : utilisateur volontaire dont la sélection a été détaillée plus haut.

Description : l'ergonomie de l'application doit faire l'objet de tests pour assurer l'accessibilité et l'interactivité de l'application.

Pré-conditions : l'utilisateur doit porter les lunettes de réalité mixte HoloLens et avoir lancé l'application HoloDoc.

Démarrage : l'utilisateur souhaite commencer la classification de document.

Scénario 01 - Numérisation et sauvegarde

1. *L'utilisateur* possède un document qu'il souhaite enregistrer dans la base de documents.
2. *L'utilisateur* place ce document dans le champ de vision de l'HoloLens.
3. *L'utilisateur* effectue un *air tap*.
4. Le système numérise le document, met à jour l'affichage et propose à l'utilisateur un outil de labellisation.
5. *L'utilisateur* labellise le document.
6. *L'utilisateur* valide la labellisation.
 - (a) *L'utilisateur* corrige la labellisation.
 - (b) *L'utilisateur* annule la labellisation.
7. **Le système** enregistre les informations dans la base de documents.
8. *L'utilisateur* continue d'utiliser l'application.
 - (a) *L'utilisateur* quitte l'application.

Scénario 02 - Reconnaissance

1. *L'utilisateur* possède un document déjà présent dans la base de documents.

2. *L'utilisateur* place ce document dans le champ de vision de l'HoloLens.
3. *L'utilisateur* effectue un *air tap*.
4. **Le système** numérise le document, l'identifie dans la base de documents et met à jour l'affichage avec les informations du document identifié.
5. **Le système** propose à l'utilisateur un outil de labellisation.
6. *L'utilisateur* valide la labellisation.
 - (a) *L'utilisateur* corrige la labellisation.
 - (b) *L'utilisateur* annule la labellisation.
7. *L'utilisateur* modifie les informations du document à l'aide de l'outil de la labellisation.
8. *L'utilisateur* continue d'utiliser l'application.
 - (a) *L'utilisateur* quitte l'application.

Scénario 03 - Création de liens

1. *L'utilisateur* possède plusieurs documents déjà numérisés dans son espace de travail et souhaite lier ces documents entre eux.
2. *L'utilisateur* souhaite commencer la liaison et fait un *air tap* maintenu sur un document.
3. **Le système** envoie un retour sonore à l'utilisateur.
4. *L'utilisateur* regarde un second document et relâche le *air tap*
 - (a) *L'utilisateur* ne souhaite finalement pas lié ce document et relâche le *air tap* sur ce même document.
5. **Le système** crée un lien entre les deux documents et envoie un retour visuel et sonore à l'utilisateur.
 - (a) **Le système** termine la création de lien prématurément et informe l'utilisateur de l'échec de l'opération.
6. *L'utilisateur* continue d'utiliser l'application.
 - (a) *L'utilisateur* quitte l'application.

Scénario 04 - Suppression de liens

1. *L'utilisateur* possède plusieurs documents liés déjà numérisés dans son espace de travail et souhaite délier un des documents présents.
2. *L'utilisateur* ouvre un des documents numérisé et possédant des liens qu'il souhaite supprimer.

3. **Le système** propose à l'utilisateur une fonctionnalité lui permettant de supprimer tous les liens d'un document.
4. *L'utilisateur* fait un *air tap* ou utilise la commande vocale pour déclencher la suppression des liens.
5. **Le système** brise les liens et envoie un retour visuel et sonore à l'utilisateur.
6. *L'utilisateur* continue d'utiliser l'application.
 - (a) *L'utilisateur* quitte l'application.

Scénario 05 - Mise à jour de la numérisation d'un document

1. *L'utilisateur* possède plusieurs documents déjà numérisés dans son espace de travail et souhaite mettre à jour un des documents présents.
2. *L'utilisateur* ouvre le document dont il souhaite mettre à jour la numérisation.
3. **Le système** propose à l'utilisateur une fonctionnalité lui permettant de mettre à jour la numérisation.
4. *L'utilisateur* fait un *air tap* ou utilise la commande vocale pour utiliser la fonctionnalité.
5. **Le système** bascule en mode numérisation de document.
6. *L'utilisateur* d'un *air tap* déclenche la numérisation.
7. **Le système** numérise le document et met à jour l'affichage du document.
8. *L'utilisateur* continue d'utiliser l'application.
 - (a) *L'utilisateur* quitte l'application.

Chapitre 3

Solution

3.1 Architecture

Notre solution se divise en trois parties (fig 3.1 et 3.2) :

- Le premier est le serveur possédant la base de données de documents.
- Le second est l'**HoloLens**, les lunettes de réalité mixte.
- le troisième est l'application **Unity** gérant les hologrammes



FIGURE 3.1 – Architecture Matérielle

3.1.1 HoloLens

Microsoft HoloLens est une paire de lunettes de mixte augmentée permettant d'afficher des hologrammes dans le champ de vision de l'utilisateur.

D'un point de vue strictement matériel [17], le casque est un ordinateur complet équipé d'une version de Windows adaptée et compatible avec Windows 10. Trois processeurs sont utilisés : le premier est le CPU principal, le

deuxième est un processeur graphique (GPU) et le troisième gère les hologrammes (baptisé HPU pour « Holographic Processing Unit »). Des capteurs de mouvements permettent à l'utilisateur de se déplacer en l'utilisant, le son produit par le casque est spatialisé. La simulation des hologrammes fonctionne avec les gestes de l'utilisateur, une commande vocale est aussi disponible, le casque ne nécessite pas d'être connecté à Internet ou à un autre appareil pour fonctionner. Il pèse 579 grammes et propose un champ de vision d'environ 30° par 17.5°. L'HoloLens est équipé de 2 Go de mémoire vive, 64 Go de capacité de stockage, et des connexions Wi-Fi 802.11ac et Bluetooth 4.1.

Pour la partie logicielle, notre application sera développée à l'aide d'**Unity** en utilisant le langage **C#**. Cette application **Unity** s'appuie, pour effectuer tous les traitements d'images, sur une bibliothèque externe **C++** utilisant **OpenCV**[30]. Cette application devra : récupérer le flux vidéo, gérer les interactions utilisateurs (IHM), détecter des documents, afficher des menus, envoyer des requêtes au serveur, prendre des photos de la vision/documents, effectuer des traitements sur les photos, attribuer des informations sur le document, déterminer la position dans l'espace virtuel des documents.

3.1.2 Serveur

Il n'y a aucune contrainte technique pour le serveur, cela pourrait être un ordinateur dédié surtout si la base de données devient conséquente. Il s'agira d'une application **Node.js**[11] implémentant un serveur web proposant une API pour accéder et modifier la base de données.

Afin de stocker et accéder aux données associées aux documents nous avons mis une base de données en place sur le serveur. Cette base de données nous permet de stocker deux choses importantes, les informations de chaque document et les liens entre les documents. La base de données a donc une structure extrêmement simple. Nous avons fait le choix d'utiliser une base de données pour avoir un accès rapide et simple aux données. De plus, le choix de **MongoDB**[19] et donc d'une base de données **NoSQL** rend la structure de la base et de ses tables évolutives et adaptatives.

Nous avons préféré l'utilisation d'une base de données (BDD) comparée à l'utilisation de fichiers structurés tels que des XML pour diverses raisons. Tout d'abord, une base de données **MongoDB** n'est pas difficile à mettre en œuvre, la définition des modèles constituant la base de données est simple et rapide. De plus, l'utilisation de la bibliothèque **Mongoose**[2] en **Node.js** nous permet de lancer des requêtes de manière naturelle depuis du code **JavaScript**. De plus, l'utilisation d'une BDD permet simultanément de modifier ou accéder

à des mêmes données. Cette notion de concurrence n'est pas possible avec des fichiers XML. Enfin, l'utilisation de MongoDB nous permet de faire évoluer et donc de stocker facilement toutes informations supplémentaires que l'on souhaite sans devoir reconstruire toute la base ou devoir modifier toutes les données existantes. Néanmoins, ce choix nous oblige à être dépendant d'un gestionnaire de base de données.

Un serveur web reposant sur un ensemble de requêtes HTTP nous permet de lire ou d'écrire dans la base de données. Nous avons donc fait le choix de faire tourner une BDD et un serveur web en parallèle et donc de décentraliser la gestion des données. Ceci nous a semblé évident à la vue des caractéristiques de l'*HoloLens*, en effet ses performances et plus particulièrement sa mémoire sont des freins pour stocker toutes les informations en local sur le casque. De plus, cette décentralisation permettrait du travail collaboratif à l'aide de plusieurs paires de lunettes ou de différents dispositifs capables d'envoyer des requêtes au serveur web. Nous avons fait le choix de mettre en place un serveur HTTP, car l'utilisation de ce protocole et plus particulièrement l'utilisation de Node.js et du framework Express[10] nous ont permis de rapidement mettre un serveur en place. Nous avons ainsi été rapidement capables de recevoir des requêtes et de renvoyer le résultat de ces requêtes. Néanmoins, il pourrait être envisageable de développer un serveur dédié pour des raisons de performance ou tout du moins de ne pas s'appuyer sur Node.js. En effet, Node.js ne permet pas le multi-threading et donc pourrait ne pas être efficace si le serveur reçoit de nombreuses requêtes nécessitant des calculs plus ou moins coûteux en même temps.

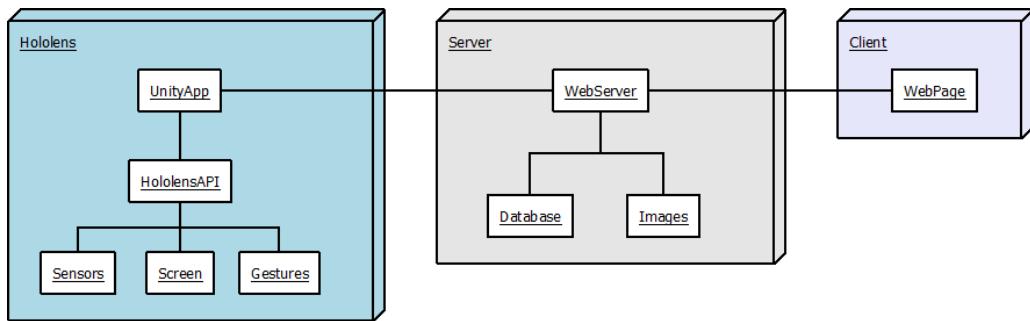


FIGURE 3.2 – Diagramme de déploiement prévisionnel

3.2 Interface homme machine (IHM)

Pour toute application, l'utilisateur a une nécessité absolue d'interagir avec le système et donc une interface homme-machine (IHM) cohérente et

ergonomique est nécessaire. Une IHM a pour rôle de trouver et répertorier les façons les plus efficaces, les plus accessibles et les plus intuitives pour compléter une tâche le plus rapidement et précisément possible. Pour s'inscrire dans un contexte de classification / tri de documents, une IHM efficace permet de sauver un temps précieux et rendre la tâche plus agréable. Cette tâche devant s'effectuer en réalité mixte, une IHM classique ne permet pas de mettre à profit les opportunités qu'offre une telle technologie. En contrepartie, elle impose aussi un lot de contrainte auxquelles il faut faire attention lors du développement de notre interface. En effet, d'après une étude de Microsoft[15] réalisée lors de la sortie de leur première application pour les lunettes HoloLens, certaines pratiques sont à éviter, car elles peuvent venir déranger et perturber l'utilisateur. Par exemple, dans une IHM classique l'utilisation d'une fenêtre de type popup, pour notifier un utilisateur, peut être une bonne idée. En réalité mixte c'est tout l'inverse : l'apparition brutale d'une information au centre du champ de vision de l'utilisateur peut être vécue comme quelque chose de très désagréable, qui leur "bloque le chemin", et donc déclencher un comportement de rejet immédiat.

3.2.1 Spécificité de l'HoloLens

De plus, il faut noter que lors de l'utilisation d'un HoloLens l'utilisateur devient la caméra (fig 3.3) et ainsi l'interaction avec les hologrammes (gestes, commandes vocales, etc.) n'est possible que lorsque ceux-ci se trouvent exactement là où l'utilisateur regarde (*Gaze targeting* fig 3.4). Ce mode de fonctionnement est très particulier et impose donc de fortes contraintes : l'utilisation d'un affichage tête haute est possible, mais aucune interaction avec celui-ci n'est envisageable du fait du *Gaze targeting*. Si l'utilisateur bouge sa tête, l'affichage tête haute se déplace en conséquence et il en résulte une impossibilité pour l'utilisateur de viser l'affichage tête haute et donc d'interagir avec celui-ci. Cliquer sur un bouton pendant que l'on regarde un autre endroit n'est pas non plus possible par exemple.

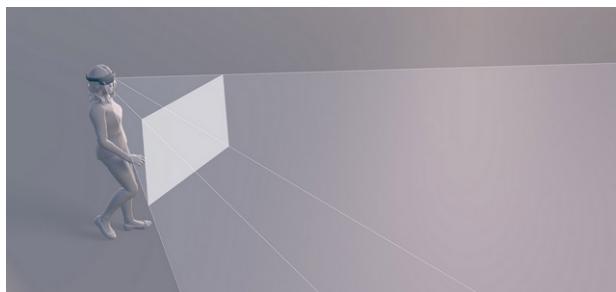
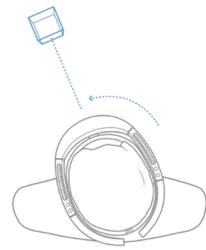
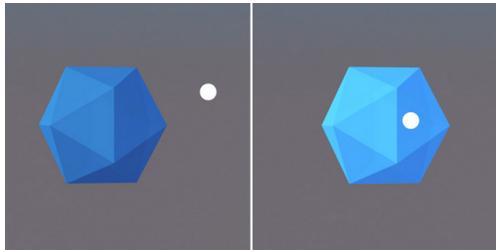


FIGURE 3.3 – HoloLens : l'utilisateur est la caméra



(a) Gaze targeting



(b) Gaze targeting effect

FIGURE 3.4 – HoloLens : Gaze targeting et gaze targeting effect

En prenant en compte toutes ces remarques nous avons prototypé plusieurs versions de notre IHM afin de trouver une version de celle-ci à la fois complète, compréhensive, *user friendly* et naturelle. Pour construire ces prototypes, nous avons tout d'abord dû définir les fonctionnalités principales de notre application pour ensuite en déduire les blocs composant notre IHM. Nous avons structuré notre IHM en trois parties principales.

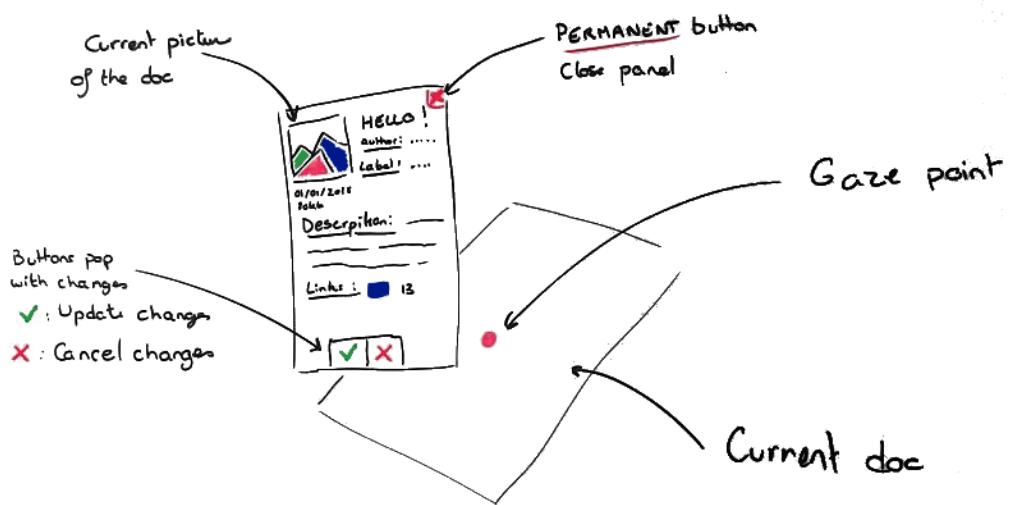


FIGURE 3.5 – Prototypage de l'IHM

3.2.2 Numérisation des documents

La première partie correspond au besoin principal de l'application à savoir pouvoir prendre des photographies de nos documents. Dans un premier temps, nos réflexes d'utilisateur lambda nous ont poussé à nous tourner vers un menu classique. L'opérateur aurait pu ouvrir ce dernier à sa guise *via* un click sur un bouton ou un geste dédié. Il aurait pu ainsi sélectionner la fonctionnalité lui permettant de prendre des photographies des documents.

Nous nous sommes rendu compte que l'utilisation d'un menu de ce type aurait pu être intrusive, non agréable à utiliser et que nous ne mettions pas à profit l'opportunité que nous offraient les lunettes à savoir la possibilité d'utiliser l'espace physique qui nous entourait. De plus, ce prototype n'était pas du tout ergonomique, car pour prendre une photographie, il fallait ouvrir le menu, cliquer sur la fonctionnalité photographie, puis regarder le document à prendre en photo et réaliser un geste pour pouvoir finalement prendre la photographie.

Nous avons donc réfléchi à une seconde version où le menu ne serait plus un menu global, mais un menu associé à chaque document. Pour ce faire, nous avons eu besoin de développer un algorithme de détection de document expliqué plus en détail dans la partie 3.3. Ce menu comme montré sur le croquis (fig 3.6) est accroché à chaque document et les fonctionnalités qu'il propose ne sont applicables que sur le document auquel il est associé. Cette option comportait alors quelques problèmes comme le fait qu'il y ait encore un menu à proprement parlé. En effet, la présence d'un élément externe au document force l'utilisateur à regarder autre chose que le document sur lequel il veut agir. Même si le menu est collé au dit document, lors de la prise d'une photo il aurait fallu cliquer sur le menu puis regarder le document pour avoir une bonne photo ce qui n'est pas très naturel. De plus sur des petits documents ou des documents éloignés dans le champ de vision, le menu aurait été trop petit pour que chaque fonctionnalité soit accessible et donc il aurait été difficile à viser comme expliqué plus haut.

Nous avons finalement décidé de supprimer le menu et d'utiliser chaque document physique comme un bouton comme présenté sur le prototype (fig 3.5). Pour prendre une photographie d'un document, il suffit donc de le regarder et de réaliser le geste associé. Cette dernière version est bien plus naturelle et corrige tous les défauts des versions précédentes. Les indices visuels ne sont pas intrusifs, le regard est toujours porté sur le document et tout le processus est simplifié.

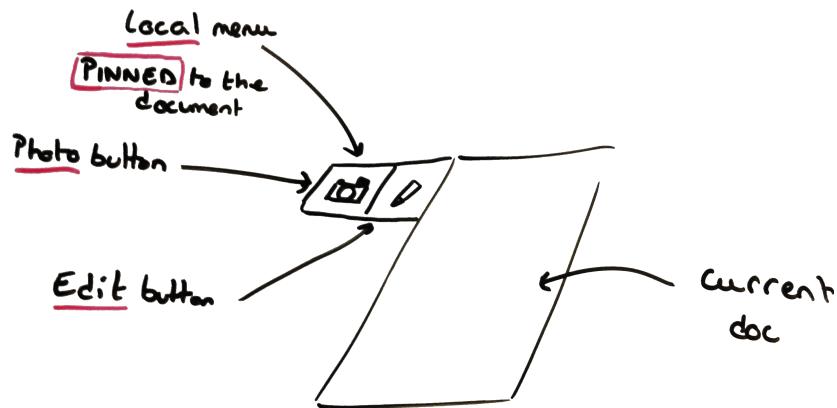
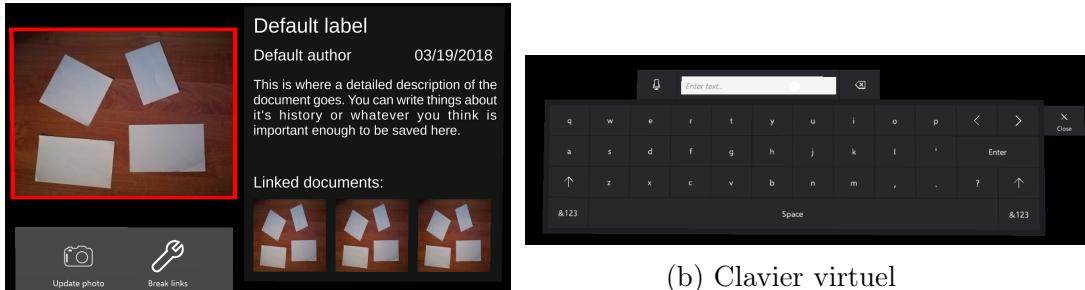


FIGURE 3.6 – Menu accroché au document

3.2.3 Métadonnées des documents

Notre IHM se compose aussi d'une partie dédiée à afficher les informations des documents. Ces informations sont présentées sous forme de fiches (fig 3.7a) et sont affichables lors d'une interaction avec un document. Dans un premier temps, un click sur un bouton du menu permettait d'afficher ces fiches, mais avec les multiples évolutions qu'a subies l'IHM, cette interaction n'était plus en adéquation avec notre façon de voir les choses, à savoir placer le document physique au centre des interactions. Nous avons donc choisi d'associer le même geste que pour la prise de photo à l'affichage d'une fiche sur un document. En effet, pour pouvoir afficher ladite fiche, il faut au préalable que le document ait été pris en photo, comme pour une chaîne de traitement classique en gestion électronique de document, l'opérateur ne peut rajouter des métadonnées qu'à un document précédemment numérisé. Ainsi, ceci ne pose pas de problèmes et permet de simplifier les interactions. Un premier *air tap* sur un document permet de prendre celui-ci en photographie tandis qu'un second permet d'afficher les informations associées à celui-ci.

Une fois la fiche affichée, il est possible d'éditer les informations qui la composent. Encore une fois, un simple *air tap* sur la donnée que l'on souhaite éditer permet d'ouvrir un clavier virtuel (fig 3.7b) permettant de modifier cette dernière. Pour un confort d'utilisation, la saisie vocale est aussi possible et est à préconiser lors de la saisie d'une information de taille conséquente.



(a) Fiche d'information d'un document

FIGURE 3.7 – Interface : Fiche et édition

3.2.4 Lien entre les documents

La dernière grosse partie de notre IHM correspond à la création de liens entre les documents. En effet, dans un cas d'utilisation classique un utilisateur souhaite pouvoir lier deux documents ensemble selon les critères de son choix. Pour manifester ce souhait, dans le monde réel un utilisateur aurait tendance à prendre en main les deux documents et soit à former un tas avec ces documents soit à les placer côté à côté. En prenant ces informations en compte, nous avons choisi de créer un nouveau geste qui s'inscrit dans cette façon de penser et qui peut s'apparenter au glisser-déposer présent dans beaucoup d'interfaces de nos jours. Ce geste permet à l'utilisateur de choisir un premier document, puis d'un pincement de doigt (comme pour un *air tap*) sélectionner celui-ci. Une fois le premier document sélectionné, en gardant les doigts bien pincés l'utilisateur peut regarder un second document. Lorsque celui-ci relâche la pression, si un deuxième document est ciblé, comme expliqué plus haut (partie fonctionnement de l'*HoloLens*), un lien se créera entre ces deux documents. Si l'utilisateur ne souhaite finalement pas créer de liens, une commande vocale ainsi qu'un bouton sont disponibles pour permettre à l'utilisateur de briser ce dernier.

Pour manifester un lien entre deux documents, nous avons choisi d'ajouter des stimulus visuels et sonores. Au début, nous avions pensé créer des liens (des traits) entre les documents, mais dans le cas où un grand nombre de documents seraient présents dans un même lien, l'affichage aurait fini par être illisible (fig 3.8a) et donc incompréhensible pour l'utilisateur. Pour garder cette lisibilité, nous avons donc décidé de ne pas surcharger notre IHM. Nous avons donc fait le choix de changer la couleur des contours (provenant de la détection de documents) pour que tous les documents présents dans un même lien possèdent une couleur commune (fig 3.8b). Nous avons réalisé que ce choix d'IHM pourrait être impactant pour les personnes distinguant mal les couleurs, c'est pourquoi nous avons aussi ajouté dans la fiche du

document une prévisualisation des documents liés à celui-ci.



FIGURE 3.8 – Différentes versions de l'affichage des liens

3.2.5 Problèmes rencontrés et solutions apportées

Comme présenté dans les parties précédentes, nous avons choisi de mettre au cœur de notre IHM le document physique (voir fig 3.5). Les interactions n'étant possibles qu'avec des hologrammes, nous avons voulu créer un hologramme simulant le contour des documents que l'on viendrait simplement superposer au document physique. Ainsi, l'interaction avec les hologrammes simulerait l'interaction directe avec les documents physiques. Afin de mettre en place cette IHM, nous avions besoin d'avoir accès à plusieurs informations. Tout d'abord, nous devions récupérer le flux vidéo de la caméra de l'HoloLens, afin de le traiter et de détecter les documents présents dans le monde réel en temps réel. Ensuite, après les avoir détectés, nous devions convertir les pixels représentant les documents en coordonnées 3D afin de générer un hologramme représentant le contour du document. Pour convertir des coordonnées 2D (pixel de l'image) en coordonnées 3D, nous devons utiliser la matrice de projection de la caméra. Ensuite, nous devons faire appel à une matrice de transformation afin de convertir la position 3D du repère de la caméra dans le repère d'Unity. En effet, la caméra d'Unity servant à visionner les hologrammes, n'est pas la même caméra que celle de l'HoloLens utilisé pour récupérer le flux vidéo. Après avoir récupéré les positions 3D des pixels, nous devions lancer des rayons depuis la caméra Unity vers ses positions et récupérer leur point d'impact. Afin de pouvoir obtenir des points d'impact dans le monde Unity simulant le monde réel, nous devions utiliser la cartographie spatiale (spatial mapping) permettant d'obtenir une représentation 3D du monde réel et donc la possibilité d'interagir avec directement dans Unity.

Maintenant que nous avons défini l'ensemble des informations nécessaires pour mettre en place notre IHM, nous devons les récupérer depuis Unity. Concernant le flux vidéo de la caméra, la récupération de celui-ci se fait

assez rapidement à l'aide d'Unity : il suffit de créer un objet `WebCamTexture` qui permet de récupérer les informations d'un appareil de prise de vues, par exemple la caméra de l'`HoloLens`. Concernant la cartographie spatiale, nous avons utilisé un `prefab` (c'est-à-dire un objet préfabriqué qui contient généralement un ensemble de scripts) fourni par le `MixedRealityToolkit` de Microsoft. Sur l'image 3.9, nous voyons la scène reconstruite (les triangles blancs), les rayons lancés (trait jaune) et la schématisation du document détecté à l'aide des quatre impacts trouvés.

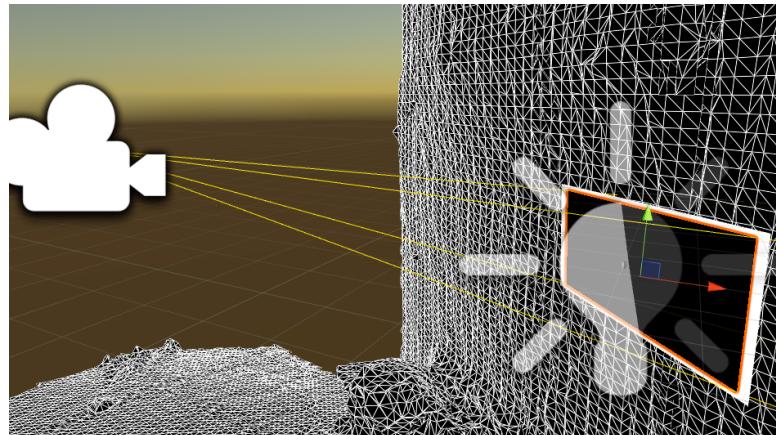


FIGURE 3.9 – Spatial Mapping avec lancer de rayon

Pour récupérer les matrices de projection et de transformation, le seul moyen de le faire avec l'API d'Unity est de créer un `PhotoCapture`, qui permet uniquement de prendre des photos. Malheureusement, l'`HoloLens` ne permet pas la prise de photographie en même temps que l'utilisation du flux vidéo. Concernant la matrice de projection, celle-ci étant toujours identique, nous pouvions contourner le problème en prenant par exemple une photographie au lancement de l'application et en sauvegardant ainsi la matrice de projection. Mais le problème reste entier concernant l'obtention de la matrice de transformation. Nous avons finalement réussi à trouver une DLL permettant de récupérer directement ses matrices à partir du flux vidéo de la caméra. Néanmoins, nous n'avons pas eu le temps de mettre en place cette solution et nous avons dû nous tourner vers une autre façon de présenter notre IHM afin de pouvoir rendre un projet fonctionnel dans le temps imparti. Nous avons donc dû abandonner l'idée de placer le document physique au centre de notre IHM en échange d'une IHM plus "classique" formant une fiche résumant l'ensemble des informations concernant le document (sa prévisualisation, ses informations de bases, la prévisualisation des autres documents liés à lui, etc.). Pour cela, nous ne détectons plus les documents directement dans le flux vidéo, mais simplement lors de la prise de photo.

3.3 Détection de Document

Lorsque l'on parle de traitement d'image, comme pour beaucoup de problèmes informatiques, on sait qu'il existe parfois une multitude de solutions à un problème. Ici, nous devons détecter des documents au sein d'une image. Nous avons décidé de tester plusieurs méthodes et comparé leur temps d'exécution ainsi que leur efficacité. Pour ce faire, nous utilisons la bibliothèque OpenCV C++ à la fois complète et performante. Son utilisation finale est remise en question dans la partie 4.2, mais elle nous a permis de prototyper un certain nombre de méthodes afin de définir notre protocole de détection finale. Avant de commencer à chercher des solutions, il faut bien définir le problème. Comme précisé dans la partie condition d'utilisation de la section 2.6, le support des documents est uni. Les documents peuvent être à des distances, positions et orientations variables. Leurs compositions ne sont pas forcément connues à l'avance, ils peuvent être remplis de détails ou complètement unis (voir Section 2.5 et Annexe A). Nous avons défini 9 images témoins présentant divers degrés de difficultés ainsi que des particularités servant à mettre à l'épreuve nos détections. Celles-ci sont présentes dans l'Annexe B.

3.3.1 Binarisation de l'image

Comme beaucoup de détections, la nôtre commence par une binarisation¹ de l'image. Trois grandes voies s'offrent à nous, la première étant une détection de contour classique. La détection de Canny[4] a été choisi pour sa qualité et son rendu « filaire » des contours (ceux-ci ne font qu'un pixel d'épaisseur), en revanche l'utilisation d'un flou lors de détection classique est conseillée pour améliorer les résultats. La seconde est une autre façon de détecter les contours avec un seuillage adaptatif. La troisième voie est un seuillage permettant de séparer le fond (uni) des documents (potentiellement uni également). En analysant ces voies, nous avons pu définir 6 méthodes de binarisations différentes (graphiquement représenté sur la figure 3.10 à la fin de cette section).

1. Image Couleur → Niveau de Gris → Filtre Bilatéral → Canny
2. Image Couleur → Niveau de Gris → Flou → Canny
3. Image Couleur → Niveau de Gris → Seuillage adaptatif
4. Image Couleur → Seuillage Couleur
5. Image Couleur → Seuillage Couleur → Seuillage adaptatif
6. Image Couleur → Seuillage Couleur → Canny

1. Les pixels n'ont plus que deux valeurs possibles

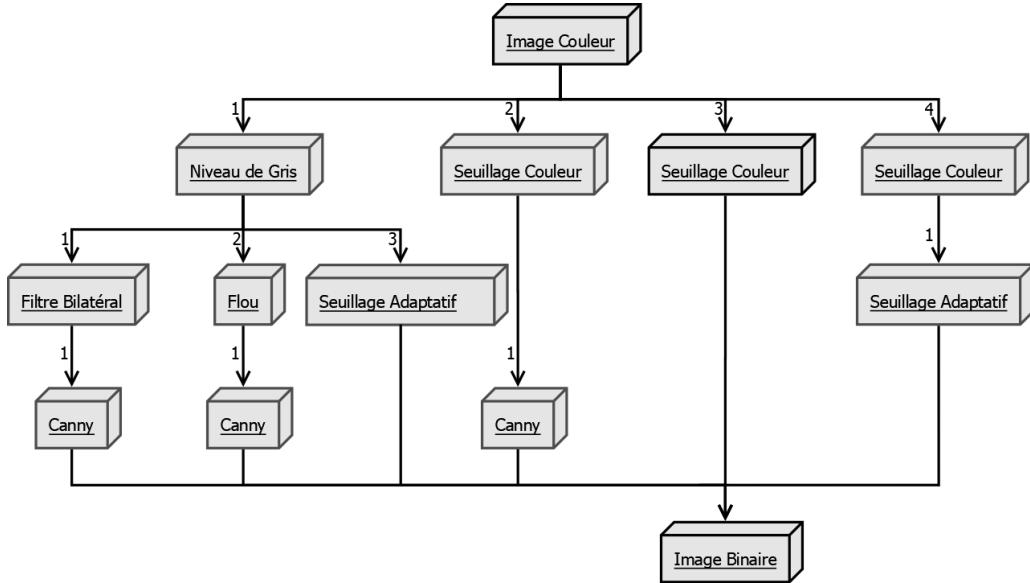


FIGURE 3.10 – Méthodes de Binarisation

Suite à l'exécution de ces méthodes, de leurs résultats et de leur temps d'exécution (analyse détaillée dans la partie 4.2.1) nous allons conserver les méthodes deux, quatre et cinq.

3.3.2 Extraction des documents

Nous allons maintenant passer à l'extraction des documents dans notre image binarisée. En réalité, nous allons devoir extraire des contours de quatre points avec un seul contour par document. La première étape consiste à détecter l'ensemble des contours de notre image binaire. Pour ce faire, nous utilisons la fonction dédiée d'OpenCV qui s'appuie sur l'algorithme de Suzuki et al.[26].

Suite à cette détection, nous avons beaucoup trop d'éléments parasites de contours trop petits (détails au sein d'un document) ou trop grands (plusieurs documents regroupés en un seul grand contour par exemple). Nous supprimons donc ce "bruit" en fonction de sa taille en pixel dans notre image, de plus nous éliminons ceux comportant moins de 4 sommets même si ils sont rares à cette étape de la détection.

La dernière étape consiste à procéder aux mêmes vérifications qu'à la première (longueur et nombre de sommets), mais aussi de vérifier si des contours sont en double ou à l'intérieur d'un autre plus grand. Pour ce faire, nous trions nos contours en fonction de leur aire et pour chacun nous regardons si le centroïde d'un des suivants est dans le premier contour, nous le supprimons.

Une ultime opération consiste à vérifier si le contour correspond à un quadrilatère proche d'un parallélogramme qui correspond peu ou prou à la définition d'un rectangle possiblement déformé par la perspective. Il suffit de vérifier que les côtés opposés de notre quadrilatère possèdent une longueur proche, nous autorisons à l'heure actuelle un seuil entre les deux distances pour éviter la suppression de contours uniquement à cause d'une déformation excessive ou d'un flou lors de la prise de photo qui perturbe la détection.

Entre ces deux étapes, nous avons comme pour la binarisation plusieurs voies de réflexion (résumé graphiquement sur la figure 3.11 à la fin de cette section). La première est d'utiliser les algorithmes présents dans OpenCV pour approximer un polygone par contour puis de créer une enveloppe convexe de cette approximation. La seconde méthode consiste à inverser ces deux étapes afin de comparer les résultats et temps d'exécution. À ce moment des tests, nous avons plus de contours de quatre sommets avec la seconde méthode qu'avec la première.

Cependant, cela ne suffisait pas, certains contours étaient visuellement bons, mais ne comportaient pas exactement quatre sommets. Il a fallu rajouter une étape permettant de sélectionner quatre sommets au sein d'un contour maximisant l'aire de celui-ci. Cela a rajouté une voie qui consiste à effectuer cette extraction de points à partir du premier nettoyage.

Une quatrième et dernière voie non valide, mais explorée quand même, consistait à calculer un rectangle orienté contenant un contour préalablement retenu. Bien évidemment, cette méthode ne prend pas en compte la perspective, mais a permis de prototyper une extraction de quatre sommets par contour rapide afin d'avancer sur d'autres points du projet notamment l'IHM.

1. Détection de contour → 1er Nettoyage → Approximation Polygonale → Enveloppe Convexe → Extraction de quatre points par contours → Dernier Nettoyage
2. Détection de contour → 1er Nettoyage → Enveloppe Convexe → Approximation Polygonale → Extraction de quatre points par contours → Dernier Nettoyage
3. Détection de contour → 1er Nettoyage → Extraction de quatre points par contours → Dernier Nettoyage
4. Détection de contour → 1er Nettoyage → Rectangle Orienté → Dernier Nettoyage

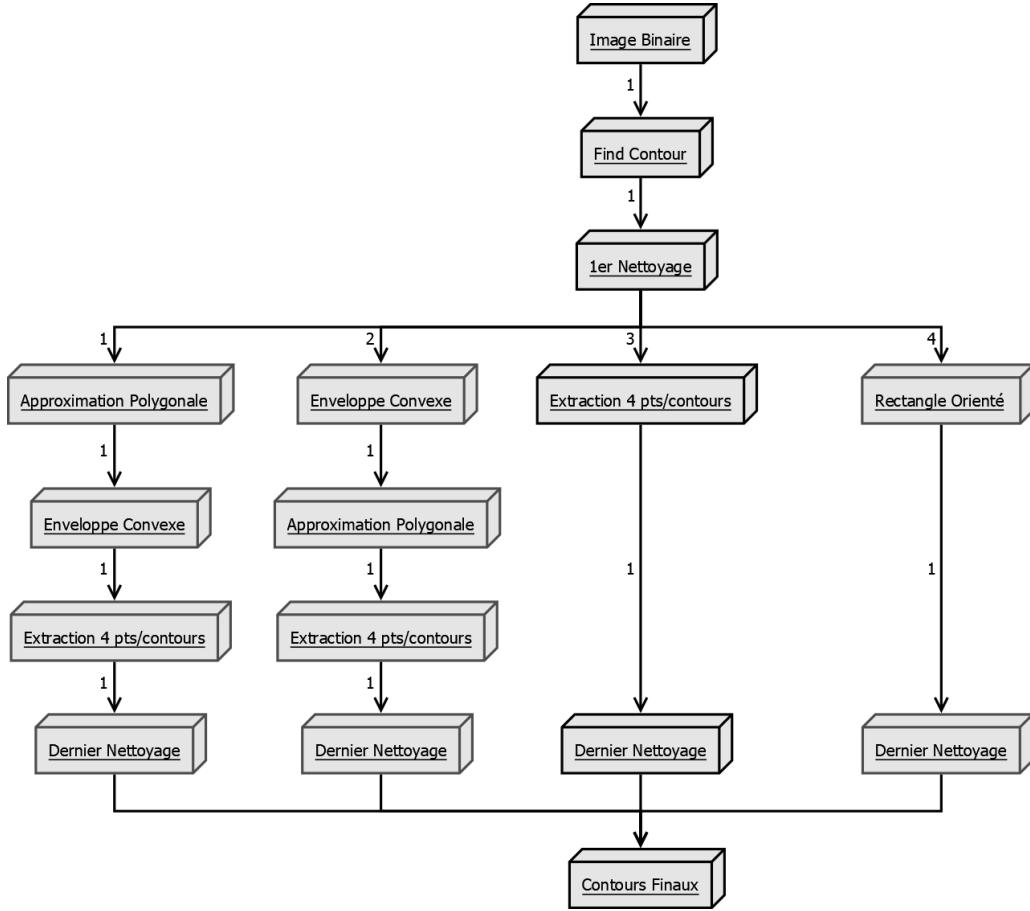


FIGURE 3.11 – Méthodes d’Extraction des contours

Les différents tests détaillés dans la partie 4.2.2 nous indiquent que la différence entre les trois premières méthodes est suffisamment négligeable pour privilégier la troisième comportant moins d’opérations, la quatrième étant à éviter.

Évidemment, pour n’extraire que le document pris en photo, il suffit de ne garder que le contour dont le centroïde est le plus proche du centre de l’image. Après cette sélection, OpenCV propose les outils permettant une déformation de la perspective afin de redresser le document.

3.4 Reconnaissance de Document

3.4.1 Méthode

La reconnaissance d’image est un sujet de recherche à part entière. Dans le cadre de notre projet, le but premier n’était pas de créer un algorithme de reconnaissance optimal. Nous souhaitions avoir un résultat rapide même si

celui-ci pouvait être sujet à des erreurs et limité à nos dix documents du jeu de test.

Nous avons pris le parti de nous inspirer de Shazam² sur le principe où nous voulions avoir une "empreinte digitale" de notre image et comparer cette empreinte à celles déjà présentes dans la base de données. Le problème évidemment était de trouver cette empreinte. Les déformations de la caméra, la résolution, l'éclairage ainsi que la manipulation de l'utilisateur ne proposent pas des conditions optimales de comparaison d'image.

Il a été décidé d'utiliser les histogrammes de deux espaces colorimétriques : le RGB (en français RVB : Rouge, Vert, Bleu) classique ainsi que le HSV (en français TSL : Teinte, Saturation, Luminance), le premier nous permet de différencier facilement les images unies les histogrammes étant binaires dans ces cas là. Mais aussi les images ayant une forte proportion d'un canal colorimétrique par rapport aux autres. Le HSV nous donne sensiblement le même genre d'informations, mais est moins impacté par l'éclairage global de l'image avec la contrepartie que dans le cadre d'image en noir et blanc, les deux premiers canaux sont entièrement à 0, seule la luminance compte.

Le problème dû à l'éclairage vu précédemment sera atténué par l'utilisation de casiers (bins). Ainsi les valeurs de l'histogramme initialement réparti de 0 à 255 le seront sur N casiers. Nous avons choisi 25 casiers par canal. Ce paramètre a été choisi empiriquement à partir de différents tests de correspondance entre images. Le nombre de casiers influe uniquement sur la place prise par l'empreinte de l'image et non sur le temps de calcul de l'histogramme.

Nous voulions également ajouter une information géométrique de l'image. La reconnaissance de forme étant elle même assez complexe. Nous avons décidé d'utiliser un autre histogramme, celui des gradients de l'image. Le principe de base de cet histogramme est de calculer pour chaque pixel (ou bloc de pixel) la direction du gradient ainsi que sa magnitude (force) et de l'ajouter dans le casier correspondant de notre histogramme à la différence que la contribution de ce gradient est répartie en fonction de ses casiers. Par exemple, si nous avons 18 casiers chacun de 20 degrés. un gradient de magnitude 4 et d'angle 10° ajoute 2 au casier 0° et 2 au casier 20°(voir figure 3.12³).

2. Logiciel de reconnaissance musicale dédié aux chansons.
3. Extrait de : www.learnopencv.com/histogram-of-oriented-gradients/

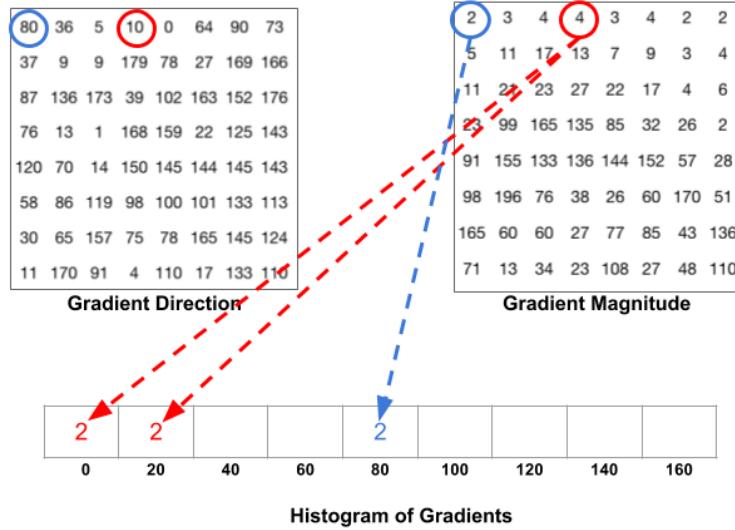


FIGURE 3.12 – Répartition de la magnitude

Une dernière méthode potentiellement plus intéressante dans la comparaison colorimétrique consistait à modifier la quantification de notre image pour retrouver les 5 couleurs prédominantes avec leur probabilité d'apparition dans l'image. La méthode de quantification implémentée utilisant un K-means clustering (regroupement par grappe) prenait plusieurs secondes sur la même configuration que lors des benchmarks de détection. Cela allait à l'encontre du fait d'avoir l'ensemble du processus (détection des documents, découpe du document central, extraction des features, comparaison à la base de données) en un temps relativement cours (idéalement en 0.25s afin de donné une impression d'instantanée et au plus inférieur à 2s).

3.4.2 Comparaison

Pour comparer nos caractéristiques, il existe plusieurs méthodes, encore une fois nous avons fait au plus simple au risque d'avoir des résultats moins déterminants, on somme la distance euclidienne de chacun de nos casiers et on exprime ce résultat en pourcentage de ressemblance.

Nous avons également ajouté un poids à chacun des sept histogrammes que nous avons. Nous avons ainsi pu voir au fil des tests que l'histogramme des gradients nous desservait lors des essais sur les images du jeu de test. Il sera donc mis en suspens pour la version finale, l'espace colorimétrique HSV étant plus discriminant, il possède un poids double par rapport au RGB. Cela donne les résultats de similarité de la figure 3.13.

	1	2	3	4	5	6	7	8	9	10
1	100									
2	77,54	100								
3	59,59	54,09	100							
4	73,74	76,10	38,21	100						
5	72,19	78,34	37,00	97,28	100					
6	59,04	53,04	72,24	34,77	33,32	100				
7	20,32	14,42	23,90	4,58	3,51	29,52	100			
8	20,32	14,42	23,90	4,58	3,51	29,52	100	100		
9	34,93	23,11	24,72	32,96	31,51	28,41	22,22	22,22	100	
10	21,34	14,45	23,64	4,60	3,52	27,62	22,22	22,22	0	100

FIGURE 3.13 – Pourcentage de similarité entre les dix images du jeu de test (Annexe A)

3.4.3 Perspective

Comme précisé plusieurs fois, notre système de reconnaissance est basique et se limite à peu de documents fortement dissociables. On remarque également sur le tableau précédent que la discrimination sur des images "propres", c'est-à-dire les images originales sans déformation et de grandes résolutions, n'est pas non plus optimale. En effet, avoir une ressemblance supérieure à 75% entre deux images est trop élevée, à l'utilisation ce niveau de ressemblance entre les descripteurs sera sans doute problématique.

Afin d'améliorer ce système, nous avons déjà pensé à plusieurs modifications importantes :

La première amélioration, plus simple, serait de ne plus avoir une distance purement analytique d'un casier d'un histogramme envers un autre, mais de voir pour chaque élément de nos tableaux sa distance avec son correspondant, mais aussi ses voisins pondérés par une gaussienne par exemple. Ainsi des différences dues à l'éclairage seraient encore plus atténées, mais la différence importante se situerait sur le fait qu'une valeur pourrait être à la limite de deux casiers. Par exemple une image uni de couleur [201, 11, 11] et une autre de [199, 9, 9] et des casiers allant de 10 en 10 auraient une ressemblance faible avec une distance euclidienne se limitant aux casiers stricts.

La seconde utiliserait le deep learning avec un système de sac de mots (Bag Of Word) avec comme caractéristique principale l'utilisation d'un algorithme tel que SURF (Speeded-Up Robust Features)[3]. L'inconvénient de cette pratique est la création d'une base de données suffisamment grande pour avoir un dictionnaire efficace.

La troisième, comme expliqué précédemment, serait d'avoir un histo-

gramme des couleurs prédominantes et de vérifier une distance entre ces couleurs en prenant en compte leur probabilité d'apparition. Ces couleurs prédominantes pourraient rendre caduc le calcul de nos six histogrammes. La prédominance serait une façon différente de stocker ces informations et serait sans doute plus efficace malgré des différences d'éclairage.

La quatrième méthode serait de trouver une information géométrique, autre que le surf (si nous n'utilisons pas de deep learning nous pensons qu'il faudrait stocker l'ensemble des points détectés c'est-à-dire plusieurs milliers voir dizaines de milliers de points pour une page d'article scientifique), suffisamment discriminante à ajouter pour pallier aux problèmes de prise de vue. L'histogramme des gradients orientés n'a pas donné de bons résultats, mais une variante serait sans doute envisageable.

3.5 Base de données

À travers ses interactions, l'utilisateur photographie et assigne des informations à des documents numériques. Afin de garder une trace de ces informations et de ces documents, nous avons fait le choix de stocker tout cela dans une base de données. Cette base est donc responsable de sauvegarder et rendre accessible l'ensemble des informations reliées à un document. Pour ce faire nous proposons une base la plus simple possible. En effet, la BDD ne contient que deux modèles qui sont :

Le modèle de document, qui contient toutes les informations des documents et donc le label, la description, l'auteur et la date. Afin de garder la BDD la plus légère et la plus simple possible, nous avons fait le choix de stocker dans ce modèle le chemin vers l'image physique du document. Cette image sera stockée sur le disque dur. Aussi, on stocke l'ensemble des caractéristiques utilisées lors de la reconnaissance dans la BDD ceci nous permet de ne pas avoir à les recalculer à chaque fois que l'on exécute la reconnaissance et la rend donc celle-ci plus rapide.

Le modèle de lien, qui contient un tableau d'identifiant de document. Un lien peut être considéré comme un groupe de documents. On stocke ici pour chaque lien l'identifiant de tous les documents dans le lien. Il faut juste garantir lors de l'ajout d'un document dans un lien que le document est dans un et un seul lien. Dans le cas, où lors de la création d'un lien, un document se retrouve dans deux liens différents, ces liens doivent être fusionnés pour n'en créer qu'un seul. Cela revient à relier deux catégories ensemble pour n'en composer plus qu'une.

Afin de mettre en œuvre la BDD, nous avons choisi d'utiliser MongoDB et de mettre en place une couche d'accès aux données sur le serveur web. Ainsi nous avons juste à lancer des requêtes au serveur web pour accéder

aux données. Notre couche d'accès nous donne l'abstraction nécessaire pour faire évoluer notre BDD ou changer de technologie au besoin. En effet le serveur ne se repose que sur cette abstraction qui permet de choisir comment stocker les données. Néanmoins, cette abstraction doit proposer les opérations suivantes :

- Créer un document : permets de créer un document avec comme données tous les paramètres donnés en entrée. Exécute une fonction de callback avec le document créé en paramètre. Cette fonction a pour objectif d'ajouter un nouveau document dans la base de données.
- Mettre à jour un document : permets la mise à jour d'un document donnée en modifiant toutes ou partie de ces informations avec celles données en paramètre.
- Obtenir un ou des documents : permets d'obtenir à l'aide d'une requête un ou plusieurs documents. La requête peut être paramétrée pour trier les documents obtenus.
- Savoir si deux documents sont liés : Retourne vrai ou faux, selon si, les documents appartiennent au même lien ou non.
- Créer un lien : à partir de deux documents, créer un lien entre eux deux. Si aucun des deux documents n'appartient à un lien alors un nouveau lien contenant ces documents est créé. Si seulement l'un d'eux est dans un lien, alors le second sera ajouté au lien du premier. Enfin, si les deux documents sont dans deux liens différents, alors il faut fusionner les deux liens en un seul.
- Retirer un document d'un lien : cette opération doit enlever un document donné d'un lien si ce document appartient à un lien.
- Obtenir un lien : Opération d'accès des liens. Cette opération peut aussi être paramétrée dans un but de filtrage.
- Obtenir le nombre de documents de la base de données.
- Retrouver un document à l'aide de ses caractéristiques : permets de retrouver un document existant en comparant les caractéristiques de tous les documents de la base avec celles données en entrée. Cette opération est nécessaire pour reconnaître un document déjà photographié et permet de récupérer toutes ses informations.

3.6 Serveur web

Afin de pouvoir communiquer entre l'HoloLens et la base de données, nous avons choisi de mettre en place un serveur web. Nous avons fait le choix d'un serveur web et de requêtes HTTP pour la simplicité de mise en œuvre. En effet, en utilisant `Node.js` et `Express`, la mise en place d'un

serveur est extrêmement naturelle et rapide. De plus, l'utilisation du paquet **Mongoose** permet depuis ce même serveur web d'accéder à la base de données. Ainsi l'objectif premier du serveur web est de proposer un certain nombre de requêtes au client pour que celui-ci ait accès à la base de données.

Dans un second temps, nous avons ajouté une librairie de traitement d'image sur le serveur. En effet, pour des raisons de tests, comparaisons et finalement de changements dans nos propositions d'IHM (comme décris dans la partie concernant l'IHM 3.2) nous avons fini par déporter toutes les tâches de traitement d'image sur le serveur web. Ce choix comporte des avantages intéressants. Tout d'abord, aucune action de traitement d'images et donc aucune dépendance à une librairie n'est requise sur l'**HoloLens**, elle l'est seulement sur un serveur qui est par nature commun à plusieurs clients. Cela facilite l'uniformisation des traitements et des versions des librairies. De plus, dans le cas d'un système embarqué comme l'**HoloLens**, cela réduit les traitements effectués localement et donc augmente l'autonomie de l'appareil. Néanmoins, cela introduit une latence due au temps de communication entre les deux entités, on perd donc en réactivité.

L'utilisation d'une bibliothèque de traitement d'image sur le serveur rend aussi possibles l'extraction de caractéristiques et leur comparaison dans l'objectif de reconnaître un document à partir d'une photo. Afin de pouvoir tirer parti de l'ensemble des prototypes existant en C++ nous avons choisi d'utiliser la librairie **OpenCV4nNdeJS**[14]. Cette librairie est un **wrapper** d'**OpenCV 3.4** en **Node.js**, il permet l'utilisation d'un grand nombre de fonctions d'**OpenCV** depuis du code **Node.js**. Ceci nous a permis de réutiliser les mêmes fonctions que pour nos prototypes et a donc accéléré la mise en place de ces algorithmes sur le serveur. L'algorithme de détection de document implémenté sur le serveur se base sur un filtre de Canny, puis un **findContours** et enfin un tri de ces contours. Cet algorithme a été choisi pour sa stabilité vis-à-vis des conditions de la détection. Concernant la reconnaissance de document et l'extraction de caractéristiques, le serveur implémente exactement la même méthode que celle décrite plus haut.

Ainsi le serveur propose au client les requêtes suivantes :

- Crédation ou reconnaissance de documents : Le serveur reçoit une image du client. Cette image doit être la photo d'un document. Sur cette photo, le document doit être contrasté avec le fond et être complet dans la photo. Le serveur va détecter ce document, extraire le document le plus au centre et extraire ses caractéristiques. Après cela, il va comparer ces caractéristiques avec celles de documents de la base de données. Dans le cas où la comparaison est suffisamment proche, alors on considère que le document dans la photo correspond à un document

dans la BDD, on va donc retourner au client ce document (on joint au document, le lien auquel appartient le document). Si l'on ne trouve pas de correspondance alors le serveur crée un document que l'on retourne au client. En plus des informations du document, le serveur retourne aussi leur image au client.

- Mise à jour de la photo d'un document : Le serveur reçoit un identifiant de document et une photo de ce document, il va donc extraire le document de la photo et mettre à jour l'image du document sur le serveur en utilisation celle extraite. Il retourne au client cette nouvelle image. Cette requête permet à l'utilisateur de corriger la photo d'un document si elle n'est pas bonne.
- Mise à jour des informations d'un document : Le serveur reçoit du client un identifiant et les informations mises à jour du document. Ces informations vont être utilisées pour mettre à jour la BDD.
- Création d'un lien : Le serveur reçoit l'identifiant de deux documents que l'utilisateur vient de lier. Il va donc mettre à jour la BDD en conséquence en créant le lien correspondant.
- Enlever un document d'un lien : Avec un identifiant de document, le serveur va sortir le document de son lien si ce document est dans un lien.

Le serveur a aussi été testé dans le cadre d'une détection de document en temps réel. Dans cette optique, l'utilisation de requêtes HTTP a montré ces limites, là où l'utilisation de requêtes TCP ou UDP a montré sa viabilité. En effet, lors de nos tests, nous avons capté le flux d'une caméra, envoyé chaque image au serveur suivant différents protocoles. Le serveur devait recevoir et traiter ces images et retourner les quatre coins de tous les documents présents dans celles-ci. Les protocoles TCP et UDP ont donné des résultats équivalents qui permettent une utilisation de la détection en temps réel. Néanmoins, l'implémentation très naïve d'UDP pourrait laisser penser qu'une utilisation plus avancée du protocole donnerait de meilleurs résultats. Enfin, l'utilisation du protocole HTTP en temps réel s'est révélée impossible à utiliser à cause de temps de réponse trop grands. Nous avons quand même gardé cette solution dans le cadre de notre IHM finale. En effet, cette solution est simple à mettre en œuvre. De plus, elle s'intègre naturellement dans le serveur web. Enfin, n'ayant plus cette contrainte de détection en temps réel, un temps de réponse plus long des requêtes est devenu acceptable.

Chapitre 4

Tests

Durant la conduite de ce projet, de nombreux tests ont été effectués.

4.1 Interface Homme-Machine

Pour valider notre Interface Homme-Machine, nous n'avons pas pu mettre en place de tests unitaires, car dans le cadre d'une IHM, cela n'a pas de sens, cependant nous avons mis en place des tests utilisateurs. Nous avons rédigé des scénarios d'utilisation que vous pouvez trouver section 2.6 visant à couvrir toutes les fonctionnalités de notre IHM, et nous avons demandé à des personnes volontaires d'essayer de réaliser ces scénarios. À l'issue de ça, nous avons fait remplir aux testeurs un questionnaire (annexe D) que nous avons rédigé pour avoir un retour sur notre besoin d'ergonomie. De ces tests et des réponses aux questionnaires, il ressort plusieurs points intéressants à aborder.

Tout d'abord, on constate en regardant les résultats du questionnaire que l'interface paraît peu intuitive (fig 4.1). En effet, la plupart des testeurs ne savaient pas ce qu'ils pouvaient ou ne pouvaient pas faire lorsqu'ils utilisaient l'application pour la première fois. Ceci souligne le manque de scène d'introduction dans laquelle l'utilisateur serait guidé à travers les fonctionnalités de base et où il lui serait demandé de réaliser plusieurs fois les opérations élémentaires proposées par la solution pour une bonne prise en main de celle-ci. Ceci peut aussi s'expliquer par le fait que les utilisateurs, pour la plupart, n'étaient pas initiés à cette technologie.

L'interface vous paraît :



FIGURE 4.1 – Questionnaire : Interface peu intuitive

On constate aussi que l'accessibilité de l'interface n'est pas celle souhaitée cependant, le retour principal sur l'accessibilité concerne le fonctionnement général de l'**HoloLens**. Devoir viser avec sa tête au lieu de simplement pouvoir pointer avec son doigt à l'endroit désiré est un gros point noir de cette technologie et ne permet pas à l'utilisateur de se sentir à l'aise dans son environnement. De plus, les gestes proposés n'ont convaincu que la moitié des utilisateurs (fig. 4.2), car il faut une certaine pratique avant de pouvoir être efficace. Là encore, une scène d'introduction aurait pu aider à se familiariser avec le fonctionnement de la technologie et amoindrir les problèmes qui lui sont directement liés. Cependant la technologie actuelle reste encore trop limitante et ne propose que deux gestes réellement distincts (le *air tap* et le *blossom*) dont un étant réservé par le système. Les testeurs restent cependant satisfaits par la présence de commandes vocales qui sont plus pratiques à utiliser que les gestes. Il faut noter toutefois qu'elles ne sont pas non plus irréprochables, car 80% des testeurs n'ont pas réussi à toutes les utiliser, et 100% des testeurs n'ont pas réussi à les utiliser du premier coup (fig 4.3). Ceci reste indépendant de notre volonté, car directement lié à la reconnaissance vocale proposée par l'**HoloLens**.

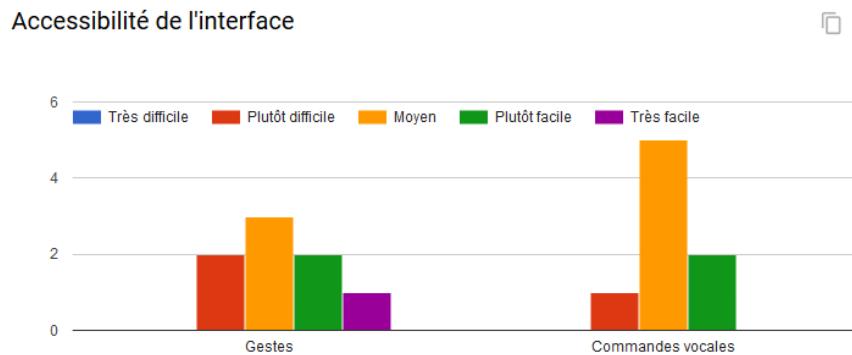


FIGURE 4.2 – Questionnaire : Interface moyennement accessible



FIGURE 4.3 – Questionnaire : Commandes vocales

Concernant la méthode de saisie virtuelle qui était une de nos priorités, les retours utilisateurs montrent de gros problèmes. Le clavier virtuel proposé n'a aucun intérêt, les utilisateurs préféreront la dictée vocale 100% du temps (fig. 4.4). Ceci s'explique par la faible accessibilité de ce dernier, pour effectuer une saisie il faut regarder chaque lettre une par une et faire un *air tap*. La dictée vocale vient donc rattraper le tout même si elle est un peu hasardeuse et pas très performante, mais encore une fois liée directement aux le système d'exploitation de l'HoloLens. Une autre méthode de saisie telle que l'utilisation d'une liste de mots déjà pré-écrits est à envisager.

Que pensez vous des méthodes de saisie proposées ?

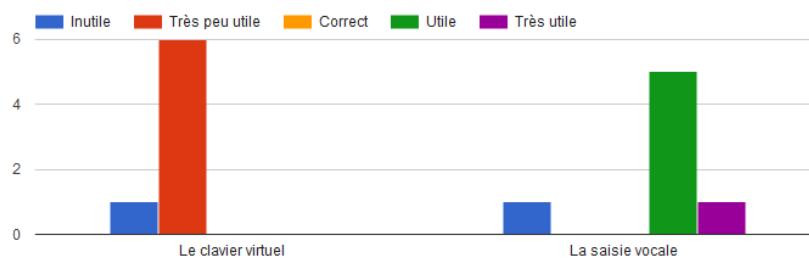


FIGURE 4.4 – Questionnaire : Méthode de saisie

Pour ce qui est, maintenant, des retours visuels et sonores dans l'interface, ils ont été très bien accueillis (fig 4.5) et ils ont permis d'aider l'utilisateur à comprendre ce qu'il se passait. On note tout de même qu'il manque encore quelques retours pour que l'utilisateur soit vraiment confiant dans ce qu'il fait par exemple lors du cadrage de la prise de photo, lors de saisie vocale ou encore lors de l'édition des métadonnées du document et la création de liens.

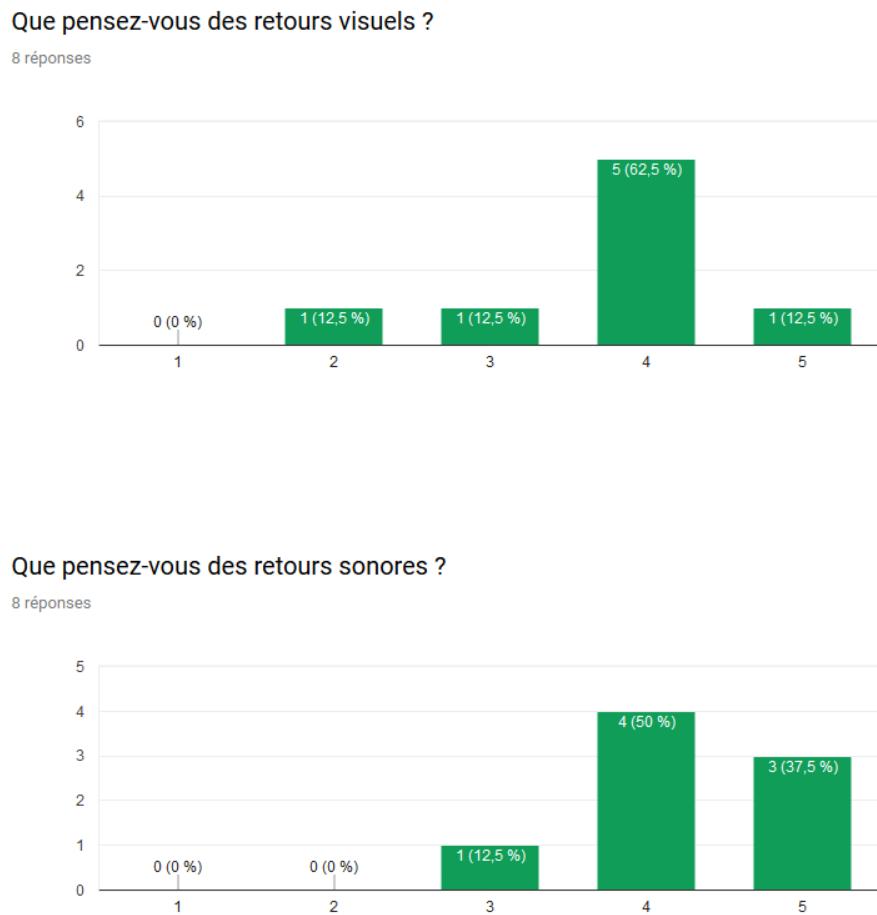


FIGURE 4.5 – Questionnaire : Retours visuels et sonores

4.1.1 Conclusion

4.2 Traitement d'image

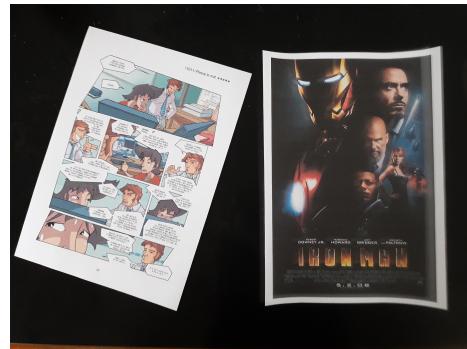
Lors des différents tests de traitement d'image, nous vérifions les temps d'exécutions. Il convient donc de préciser la configuration matérielle et logicielle. L'ordinateur utilisé est un ordinateur fixe équipé d'un processeur Intel I5-4670 à 3.40Ghz avec 8 Go de RAM DDR3 utilisant Windows 10 Éducation. La compilation C++ est faite avec Visual Studio Community en x64 et le débogage est désactivé. Sans pour autant être stricto sensu parfaitement juste les temps d'exécution, nous donnerons un ordre d'idée du résultat. De plus, les tâches étant complètement découpé certaines duplications et créations de variables intermédiaires pourront être factorisé lors d'optimisations pour la version définitive ou la poursuite du projet. De plus, nous indiquerons la

moyenne du temps d'exécution sur les 9 images pour atténuer les problèmes inhérents au système d'exploitation tel que le passage au premier plan d'une autre tâche.

Comme nous l'avons indiqué précédemment, nous avons testé chacune des méthodes implémentées pour un benchmark du temps d'exécution en C++, l'ensemble des images intermédiaires sont enregistrées afin de vérifier l'intégrité de chacune des étapes, un certain nombre de variables sont également écrites dans le terminal afin de vérifier (et stocker dans un log) l'état du système à chaque étape.

4.2.1 Binarisation

La première vérification à ce stade consiste à voir si tous nos documents sont correctement détournés et si le bruit dans l'image binaire est trop présent. Les résultats de toutes les méthodes sont présentés sur la figure 4.6.



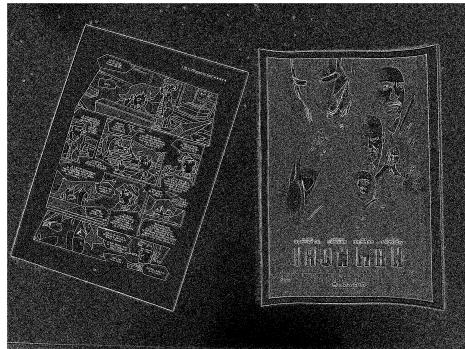
(a) Image Originale



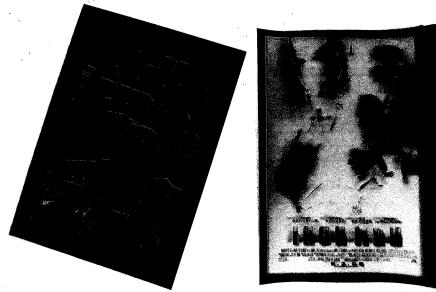
(b) Flou Bilatéral + Canny



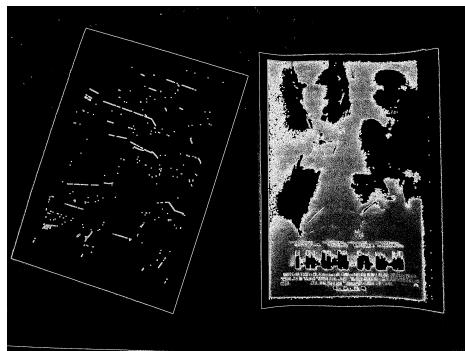
(c) Flou Gaussien + Canny



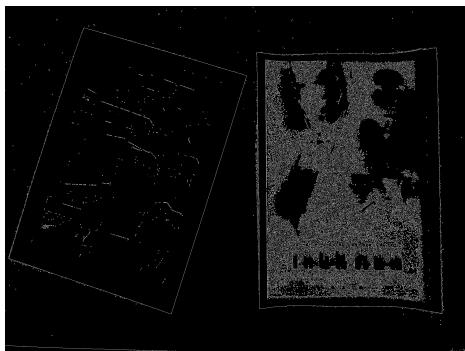
(d) Seuillage Adaptatif



(e) Seuillage Colorimétrique



(f) Seuillage Colorimétrique + Adaptatif



(g) Seuillage Colorimétrique + Canny

FIGURE 4.6 – Binarisation

À partir de ces tests, nous voyons déjà plusieurs éléments importants. Les deux premiers protocoles nous montrent que l'utilisation d'un flou bilatéral nous rajoute des informations sur la détection de contours à l'intérieur d'un document, mais cela ne nous intéresse pas spécialement. L'utilisation d'un seuillage adaptatif est complètement inutile au vu de la quantité de bruit présent. Ensuite, nous passons à un seuillage colorimétrique en fonction du fond. L'inconvénient principal est que le fond doit être uni et ne pas être encombré d'éléments perturbateurs tels que les téléphones, tapis de souris, etc. pour que cette solution puisse être viable (cette remarque est également valable pour les autres méthodes), mais également que la couleur du fond ne doit pas être trop présente au sein du document au risque d'avoir le fond détecté dans ce document. Le seuillage adaptatif appliqué au seuillage précédent nous donne une détection de bord assez précise pour le document avec une zone centrale pour l'affiche de film problématique, l'utilisation de Canny par rapport au seuillage adaptatif n'apporte pas forcément beaucoup de détails intéressants. La différence de certains n'étant pas flagrante la deuxième vérification détaillée ci-dessous est importante.

La seconde vérification consiste à évaluer le temps d'exécution. En effet, n'oublions pas que nous effectuons ces traitements sur un flux vidéo avec un retour visuel pour l'utilisateur. Une solution trop lente provoquerait une latence puis une rémanence désagréable à l'utilisateur. Nous pouvons voir le temps d'exécution (relatif à la configuration décrite plus haut) de chaque protocole pour toutes les images témoins ainsi que la moyenne de chacun sur le tableau 4.7 ci-dessous.

Image / Méthode	1	2	3	4	5	6
A	3 336,27	9,15	4,96	2,61	5,27	8,57
B	1 828,47	12,96	6,03	3,78	8,19	12,45
C	1 795,59	12,60	6,46	4,51	8,92	13,74
D	1 818,80	17,60	6,52	4,10	8,33	16,91
E	1 828,22	13,21	6,26	3,90	8,12	13,64
F	1 822,31	14,58	6,67	3,96	9,23	13,07
G	1 934,95	15,19	6,16	3,85	8,52	13,43
H	3 095,67	9,16	4,03	2,60	5,26	8,47
I	3 500,25	10,60	4,96	3,27	6,63	11,31
Moyenne	2 328,95	12,78	5,78	3,62	7,61	12,40

FIGURE 4.7 – Benchmark Binarisation

Avec ces informations, nous pouvons mettre en place une balance qualité/temps d'exécution. Très rapidement, nous pouvons supprimer la première méthode qui utilise un filtre bilatéral qui est beaucoup trop long pour être exploitable en pseudo temps réel. La démarche trois utilisant le seuillage adaptatif directement ne possède pas un temps d'exécution suffisamment bas pour pallier le défaut de la présence de bruit excessive.

Les protocoles deux et six possèdent des temps globalement proches. Le second procède à un flou plus simple que le filtre bilatéral ce qui réduit considérablement le temps par rapport à son homologue, le sixième remplace le flou par un seuillage colorimétrique, mais nous fournit trop de détails au niveau des contours notamment sur l'affiche du film. Nous supprimons donc la sixième méthode.

La seconde voie demande beaucoup de temps et fournit beaucoup de détails, mais nous décidons de la conserver au cas où la suite de la détection soit grandement impactée par ce choix. Pour continuer avec les trois dernières méthodes, il est difficile de déterminer à l'heure actuelle laquelle est la meilleure qualitativement parlant et leur temps d'exécutions bien qu'allant du simple au triple n'est pas suffisamment déterminant pour prendre une décision. Si une segmentation est suffisante sans extraction des bords, le protocole quatre ne contenant qu'une seule opération très simple est à privilégier.

Nous allons garder la méthode deux utilisant une détection de Canny, la quatre ne procédant qu'à un seuillage colorimétrique et la cinq qui ajoute un seuillage adaptatif au précédent.

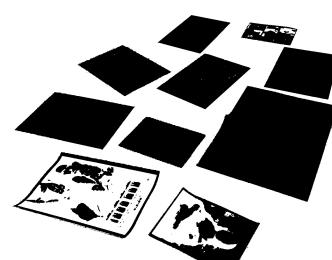
Ces derniers protocoles nous serviront de base pour tester nos détections de contours dans la section suivante. Si des différences non flagrantes à première vue sont présentes, mais que celles-ci sont déterminantes dans la suite des opérations, nous gardons en tête qu'il y a plusieurs binarisations possibles.

4.2.2 Extraction des documents

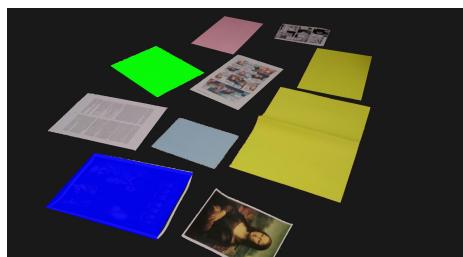
La figure 4.8 ci-dessous montre pour une image témoin avec une des solutions préférentielles précédentes (Seuillage colorimétrique uniquement) les résultats des quatre méthodes :



(a) Image Originale



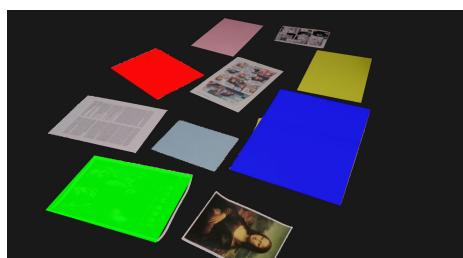
(b) Image Binaire



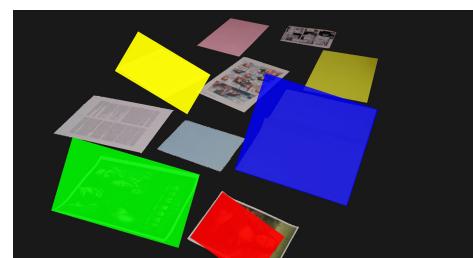
(c) Méthode 1



(d) Méthode 2



(e) Méthode 3



(f) Méthode 4

FIGURE 4.8 – Extraction des Documents

Comme précédemment nous avons effectué des tests sur le temps d'exécution (fig 4.9). Cette fois, nous indiquons directement la moyenne des temps d'exécution avec sur les lignes l'image binaire utilisée parmi les trois sélections précédentes et sur les colonnes la méthode employée.

Image / Méthode	1	2	3	4
NDG + Blur + Canny	4,18	4,48	4,20	4,51
Seuillage	3,24	3,37	3,25	3,37
Double seuillage	3,81	4,09	3,83	4,14

FIGURE 4.9 – Benchmark Extraction des documents

Nous remarquons, encore une fois, qu'évidemment la solution la plus simple consistant à récupérer quatre sommets par contours est la plus rapide. Concernant la qualité des détections, les rectangles orientés ne sont utiles que si les documents n'ont pas de déformations dû à la perspective ce qui limite grandement le champ des possibles. Les deux premières méthodes nous donnent des résultats semblables à la troisième, aucune des trois ne possède de faux positifs notamment grâce à la vérification de cohérence de forme. Selon l'image binaire de base, le résultat final reste le même. Toutefois, avec la binarisation utilisant Canny, l'élimination de contours fait face à un plus grand nombre d'entrée et de faux positifs parfois présent jusqu'à la dernière étape.

4.2.3 Conclusion

Lors de l'intégration des différentes briques du projet nous avons pu nous rendre compte que l'utilisation d'une DLL servant de lien entre Unity en C# et le traitement d'image en C++, l'application souffrait de ralentissement important. Nous avons donc décidé de passer l'intégralité du traitement d'image sur le serveur avec la bibliothèque OpenCV4NodeJS qui fait le lien d'une grande quantité de fonctions de la bibliothèque OpenCV pour Node.js.

Afin de prototyper rapidement l'extraction de caractéristiques lors de la partie 3.4 nous l'avons également implémenté en C++ même si nous savions déjà que la DLL serait inutilisable, le prototypage C++ à permis d'implémenter plusieurs méthodes plus rapidement. Une fois ceux-ci faits, nous avons testé nos méthodes en comparant chacune des dix images du jeu de test les unes aux autres. Nous avons également ajouté à notre panel dix images possédant une rotation de 90°. Puis dix nouvelles représentant les images d'origines avec des reflets plus ou moins importants. La transposition en Node.js a

ensuite pris peu de temps et des tests unitaires concernant ces fonctions ont donc été ajoutés.

Les différents tests précédents nous ont permis de chercher et trouver des méthodes parfois plus simples qui fonctionnent correctement de façon plus rapide. Évidemment, le résultat final n'est pas parfait. Nous ne cherchons pas à avoir le contour parfait de chacun des documents. Si certains documents se chevauchent, la détection croit qu'il n'y en a qu'un seul et les différentes vérifications le suppriment de la liste finale. Nous gardons pour finalité de n'avoir aucun faux positif quitte à ne pas avoir tous les documents détectés en raison de leur éloignement, chevauchement, contenu trop proche du fond...

De plus, lors de l'intégration en situation réelle, la méthode du seuillage colorimétrique préférentielle jusqu'alors est beaucoup trop influencée par l'éclairage global et la qualité de la caméra. Il faut également prendre en compte que les bureaux réellement unis ne sont pas les plus courants au profit de motifs bois, ou tachetés. La solution utilisant Canny sera alors sélectionnée avec une dilatation des contours qui permet d'éviter qu'un document soit séparé en plusieurs contours du fait de microcoupures d'une ligne dans la détection.

La méthode employée sera donc la suivante :

1. **Image Couleur**
2. Niveau de Gris
3. Flou Gaussien
4. Détection de Canny
5. Dilatation
6. **Image Binaire**
7. Détection de contour
8. Premier Nettoyage par la longueur des contours et le nombre de sommets
9. Extraction de quatre points par contours
10. Dernier Nettoyage par longueur finale des contours, suppression des doublons et contours interne et vérification de forme.
11. **Contours Finaux**

Vous trouverez dans l'annexe B la liste des images témoins ainsi que leur résultat final associé.

4.3 Base de données

Afin de garantir le fonctionnement de la base de données, nous avons choisi de mettre en place des tests unitaires pour vérifier le fonctionnement

de la couche d'accès aux données et de vérifier le contenu manuellement de la base lors de chaque test de la solution finale.

Les résultats des tests unitaires peuvent être générés à l'aide de la commande `npm test` exécutée dans le répertoire HolodocServer. Ceci générera une page web présentant les résultats des tests. De plus, on propose un outil de couverture du code permettant de s'assurer de la validité du code JavaScript. En effet, le JavaScript étant un langage interprété, des erreurs dues à l'interprétation (syntaxe, objets non définis, dots) ne seront détectées qu'à l'exécution. On cherche donc à couvrir la plus grande partie possible du code source avec les tests unitaires pour garantir la validité fonctionnelle et syntaxique du code d'accès à la base de données.

Ainsi, nous avons mis en place un framework de test unitaire `Node.js` sur le serveur et un outil de couverture de code. Nous avons choisi Mocha pour les tests unitaires et `nyc` pour la couverture de code qui sera calculée à partir des tests unitaires.

Afin de tester la couche d'accès aux données, nous avons rédigé 21 tests unitaires et une couverture de 83 %. Ces tests permettent de vérifier le comportement de la BDD selon les valeurs données en entrées. On permet donc en conclure que notre couche d'accès aux données est fonctionnelle. La couverture peut sembler faible, néanmoins, le code non couvert est exclusivement du code de gestion d'erreur due à un mauvais fonctionnement du gestionnaire de la BDD.

4.4 Serveur

Les tests présents sur le serveur sont de deux types. La première partie est des tests unitaires, qui s'exécutent en parallèle de ceux de la BDD, ces tests unitaires permettent de valider l'ensemble des algorithmes de traitement d'images sur le serveur. De la même manière que pour la BDD on propose aussi une couverture du code qui nous permet de vérifier que l'on a tout testé. Les résultats de ces tests sont consultables sur la même interface web que les résultats de la BDD. Avec un nombre de 11 tests, une couverture de 88 % a pu être atteinte pour la partie de traitement d'images sur le serveur. La seconde partie contient des tests manuels pour assurer le bon fonctionnement des requêtes.

Tous ces tests manuels se déroulent suivant le schéma suivant :

1. On met en place l'environnement de test en exécutant l'ensemble des parties prenantes au test. Cela peut être l'application HoloLens (simulé dans Unity), le serveur `Node.js` et la base de données. Il faut tout initialiser selon le test voulu. Pour vérifier que la BDD

est correctement constituée, on effectue des vérifications manuelles à l'aide d'une application graphique permettant d'accéder à la BDD et d'inspecter son contenu (dans notre cas **MongoDB Compass**[20]).

2. On effectue l'opération qui déclenchera une requête bien spécifique. Par exemple, prendre un document en photo, terminer la saisie d'un champ du document ou créer un lien.
3. Vérifier à l'aide de message dans les terminaux **Unity** et serveur que les requêtes sont bien envoyées/reçues avec les bons paramètres.
4. Vérifier que la BDD a bien été mise à jour avec les bonnes informations.

Ainsi nous avons pu tester l'ensemble des requêtes et assurer leur bon fonctionnement.

Conclusion

Concernant l'interface proposée, nos objectifs n'ont été qu'à moitié atteints. En effet, d'après les résultats du questionnaire et l'analyse des tests il reste encore beaucoup de points à améliorer. On peut cependant noter que la plupart des gros points noirs de l'IHM sont en rapport direct avec le fonctionnement de l'**HoloLens** et nous pouvons donc nous poser la question de savoir si les lunettes de réalité mixte **HoloLens** sont la bonne technologie pour ce genre d'application. Nous manquons encore de testeurs pour pouvoir réellement affirmer qu'elles ne sont pas adaptées, mais envisager une autre technologie plus libre en ce qui concerne les interactions est une piste d'amélioration qu'il faut explorer. Une autre chose importante que nous avons apprise en faisant passer ces tests est qu'il y a un manque global de clarté dans l'interface et que lors de l'utilisation de telles technologies, une telle faiblesse peut complètement perdre l'utilisateur. C'est un point sur lequel nous aurions aimé travailler un peu plus en proposant à l'utilisateur, par exemple, une scène d'introduction lui permettant de se familiariser avec les fonctionnalités et le mode de fonctionnement de la technologie. Ce manque de clarté aurait aussi pu être amoindri si nous avions pu proposer l'interface que nous souhaitions à savoir celle où le document physique était au cœur des interactions. Cette interface est pour nous une piste d'amélioration non négligeable, car beaucoup plus intuitive que celle proposée actuellement. L'ajout de plus de retours visuels et sonores par exemple lors de la numérisation de document par prise de photographie ou encore lors de la dictée vocale sont aussi des pistes d'amélioration à explorer.

Les éléments de traitements d'images étaient essentiels au fonctionnement de l'application. En revanche, ils étaient secondaires au projet. Les différentes solutions proposées ont été réfléchies et testées. Leurs découvertes et implémentations ont eu lieu après plusieurs recherches, mais d'autres solutions pourraient être plus efficace sur un flux vidéo pour une utilisation en pseudo temps réel. La reconnaissance d'image possède également des lacunes à l'heure actuelle pour les mêmes raisons.

Concernant les performances, la déportation et l'utilisation d'un serveur ont permis d'obtenir une application réactive et fluide. Il pourrait néanmoins être intéressant de mettre en place un serveur dédié utilisant un autre

protocole que HTTP pour améliorer le temps de réponse aux requêtes de l'utilisateur.

Pour conclure, les faiblesses de la technologie combinée au manque d'intuitivité de notre interface font que l'expérience n'est pas celle attendue, cependant en explorant toutes les pistes d'amélioration énoncées, l'expérience peut être vraiment améliorée et la solution proposée peut prendre tout son intérêt.

Bibliographie

- [1] Rachel F. Adler and Raquel Benbunan-Fich. Juggling on a high wire : Multitasking effects on performance. *International Journal of Human-Computer Studies*, 70(2) :156 – 168, 2012.
- [2] Automattic. Mongoose. <http://mongoosejs.com/>. Last accessed 23 March 2018.
- [3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In *In ECCV*, pages 404–417, 2006.
- [4] John Canny. A computational approach to edge detection. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 1986.
- [5] John Canny. A computational approach to edge detection. In *Readings in Computer Vision*, pages 184–203. Elsevier, 1987.
- [6] Huaigu Cao, Xiaoqing Ding, and Changsong Liu. Rectifying the bound document image captured by the camera : A model based approach. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 71–75. IEEE, 2003.
- [7] Andy Cockburn and Bruce McKenzie. 3d or not 3d ? : Evaluating the effect of the third dimension in a document management system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’01, pages 434–441, New York, NY, USA, 2001. ACM.
- [8] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1) :11–15, 1972.
- [9] Michele Fiorentino, Raffaele Amicis, Giuseppe Monno, and André Stork. Spacedesign : a mixed reality workspace for aesthetic industrial design. pages 86– 318, 02 2002.
- [10] Node.js Foundation. Express. <http://expressjs.com/>. Last accessed 23 March 2018.
- [11] Node.js Foundation. Node.js. <https://nodejs.org/>. Last accessed 23 March 2018.

- [12] Matrix Inception. Hololens d3d keyboard - prototype demo. <https://www.youtube.com/watch?v=VL2c-axyZDg>, 2016. Last accessed 23 March 2018.
- [13] Masakazu Iwamura, Tomohiro Nakai, and Koichi Kise. Improvement of retrieval speed and required amount of memory for geometric hashing by combining local invariants. In *BMVC*, pages 1–10, 2007.
- [14] justadudewhohacks. opencv4nodejs. <https://github.com/justadudewhohacks/opencv4nodejs>. Last accessed 23 March 2018.
- [15] Microsoft. Case study - 3 holostudio ui and interaction design learnings. <https://docs.microsoft.com/en-us/windows/mixed-reality/case-study-3-holostudio-ui-and-interaction-design-learnings/>. Last accessed 23 March 2018.
- [16] Microsoft. Hololens. <https://www.microsoft.com/en-us/hololens/>. Last accessed 23 March 2018.
- [17] Microsoft. Hololens hardware details. <https://docs.microsoft.com/fr-fr/windows/mixed-reality/hololens-hardware-details>. Last accessed 23 March 2018.
- [18] Microsoft. Keyboard input in unity. <https://docs.microsoft.com/fr-fr/windows/mixed-reality/keyboard-input-in-unity>, 2018. Last accessed 23 March 2018.
- [19] Inc MongoDB. Mongodb. <https://www.mongodb.com/>. Last accessed 23 March 2018.
- [20] Inc MongoDB. Mongodb compass. <https://www.mongodb.com/products/compass>. Last accessed 23 March 2018.
- [21] Tomohiro Nakai, Koichi Kise, and Masakazu Iwamura. Camera-based document image retrieval as voting for partial signatures of projective invariants. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 379–383. IEEE, 2005.
- [22] Tomohiro Nakai, Koichi Kise, and Masakazu Iwamura. Hashing with local combinations of feature points and its application to camera-based document image retrieval. *Proc. CBDAR05*, pages 87–94, 2005.
- [23] Tomohiro Nakai, Koichi Kise, and Masakazu Iwamura. Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. In *International Workshop on Document Analysis Systems*, pages 541–552. Springer, 2006.
- [24] Tomohiro Nakai, Koichi Kise, and Masakazu Iwamura. Real-time retrieval for images of documents in various languages using a web camera. In *Document Analysis and Recognition, 2009. ICDAR’09. 10th International Conference on*, pages 146–150. IEEE, 2009.

- [25] W. Piekarski, B. Gunther, and B. Thomas. Integrating virtual and augmented realities in an outdoor application. In *Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on*, pages 45–54, 1999.
- [26] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1) :32–46, 1985.
- [27] Kazutaka Takeda, Koichi Kise, and Masakazu Iwamura. Memory reduction for real-time document image retrieval with a 20 million pages database. *Proc. CBDAR*, 1, 2011.
- [28] Kazutaka Takeda, Koichi Kise, and Masakazu Iwamura. Real-time document image retrieval for a 10 million pages database with a memory efficient and stability improved llah. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1054–1058. IEEE, 2011.
- [29] Kazutaka Takeda, Koichi Kise, and Masakazu Iwamura. Real-time document image retrieval on a smartphone. In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pages 225–229. IEEE, 2012.
- [30] OpenCV team. Opencv. <https://opencv.org/>. Last accessed 23 March 2018.
- [31] Iotas Technologies. Slice keyboard introduction. <https://www.youtube.com/watch?v=YVhDbhH7nZ0>, 2018. Last accessed 23 March 2018.
- [32] Unity Technologies. Unity. <https://unity3d.com/>. Last accessed 23 March 2018.
- [33] Rong Wen, Wei-Liang Tay, Binh P. Nguyen, Chin-Boon Chng, and Chee-Kong Chui. Hand gesture guided robot-assisted surgery based on a direct augmented reality interface. *Computer Methods and Programs in Biomedicine*, 116(2) :68 – 80, 2014. New methods of human-robot interaction in medical practice.
- [34] Shi-rong Xiang, Guo-ying Zhao, Rui Chen, et al. Restoration of images scanned from thick bound documents. *Journal of Computer-Aided Design&Computer Graphics*, 2005.
- [35] Ying Xiong. Fast and accurate document detection for scanning. <https://blogs.dropbox.com/tech/2016/08/fast-and-accurate-document-detection-for-scanning/>, 2016. Last accessed 23 March 2018.
- [36] M. L. Yuan, S. K. Ong, and A. Y. C. Nee. Augmented reality for assembly guidance using a virtual interactive tool. *International Journal of Production Research*, 46(7) :1745–1767, 2008.

- [37] Zheng Zhang and Chew Lim Tan. Restoration of images scanned from thick bound documents. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 1074–1077. IEEE, 2001.

Annexe A

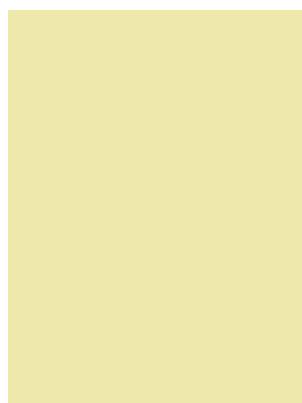
Jeu de Test



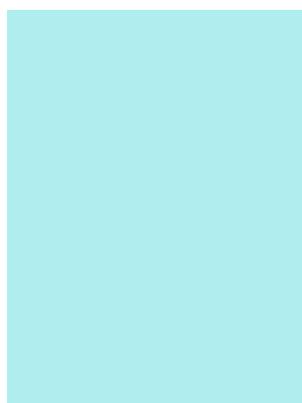
(a) Les Geeks Tome 1 Page 29 A4



(b) Affiche Iron Man A4



(g) Feuille Jaune A4 et A3



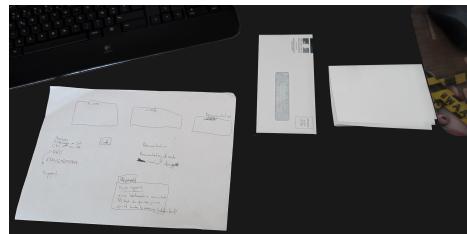
(h) Feuille Bleue A5



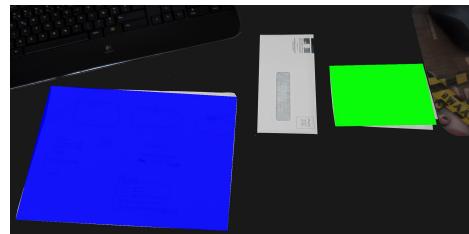
(i) Feuille Rose A4

Annexe B

Détection de Document



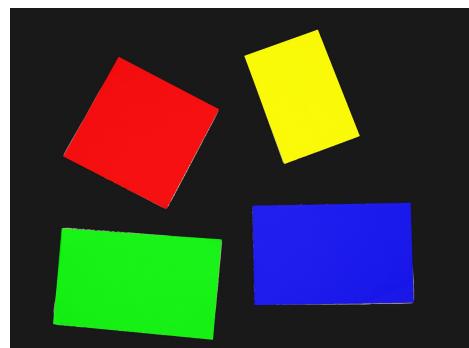
(a) Image A Originale



(b) Détections Image A



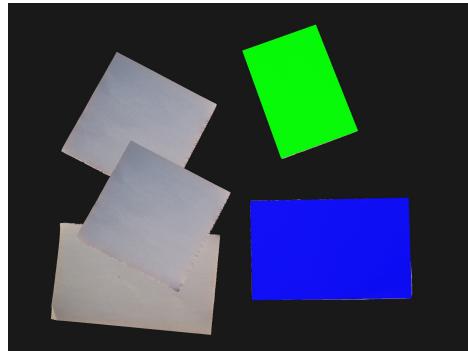
(c) Image B Originale



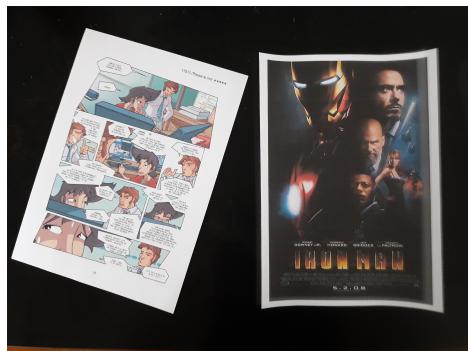
(d) Détections Image B



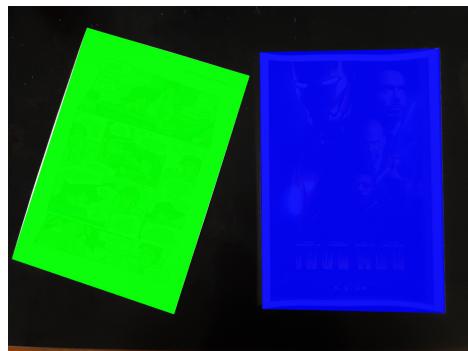
(e) Image C Originale



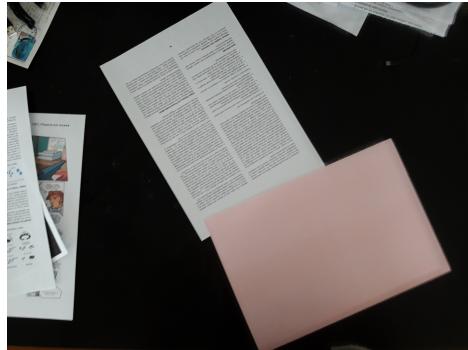
(f) Détections Image C



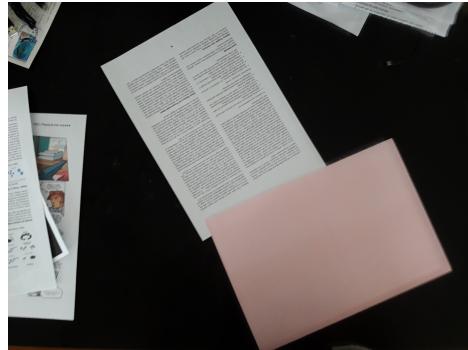
(g) Image D Originale



(h) Détections Image D



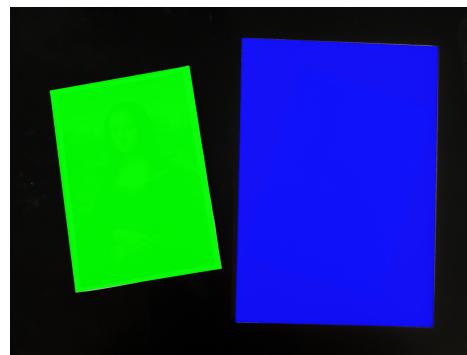
(i) Image E Originale



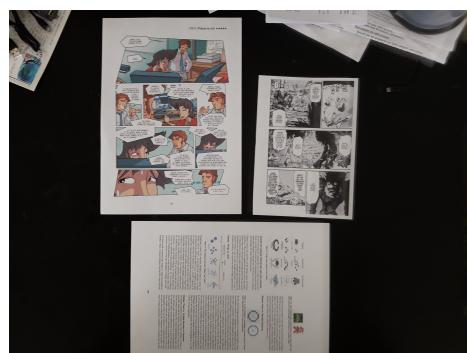
(j) Détections Image E



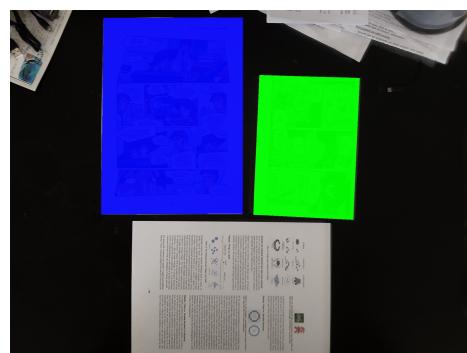
(k) Image F Originale



(l) Détections Image F



(m) Image G Originale



(n) Détections Image G



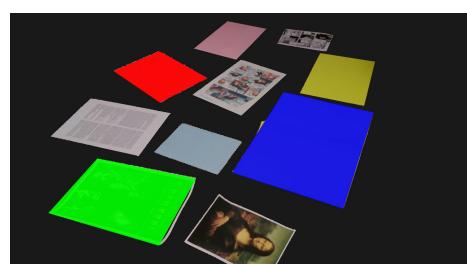
(o) Image H Originale



(p) Détections Image H



(q) Image I Originale



(r) Détections Image I

Annexe C

Protocole d'installation

C.1 Côté Serveur

Requis :

- Télécharger et installer Python 2.7.
- Télécharger et installer Git.
- Télécharger et installer CMake.
- Télécharger et installer Perl.
- Télécharger et installer Node.js.

Après cela, lancez l'invite de commande du programme Node.js ("Node.js command prompt") et allez dans le dossier HoloDocServer avec la commande `cd [APP_PATH] HoloDocServer`. Ensuite, tapez la commande `npm start` pour installer et lancer le serveur.

Les erreurs pouvant survenir sont souvent du à un mauvais ajout aux variables d'environnements de Perl et CMake.

C.2 Côté Développeur

Configuration requise :

Pour pouvoir développer sur notre application, il existe une configuration requise, définie dans la partie "System requirements" du site officiel (developer.microsoft.com/en-us/windows/mixed-reality/install_the_tools)

En résumé :

- Vous avez besoin d'un Windows 10 Pro, Entreprise ou Education 64 bits.
- 8 Go de RAM ou plus

- Dans le BIOS, les fonctionnalités suivantes doivent être prises en charge et activées :
 - Virtualisation assistée par matériel
 - Traduction d'adresses de second niveau (SLAT)
 - Prévention de l'exécution de données basée sur le matériel (DEP)
- GPU (L'émulateur peut fonctionner avec un GPU non supporté, mais sera beaucoup plus lent)
 - DirectX 11.0 ou plus récent
 - Pilote WDDM 1.2 ou plus récent

De plus, vous devez vous assurer que la fonction "**Hyper-V**" a été activée sur votre système :

Panneau de configuration → Programmes → Programmes et fonctionnalités
 → Activer ou désactiver les fonctionnalités Windows → "**Hyper-V**"

Maintenant, vous devez installer différents logiciels décrits dans liste d'installation pour HoloLens et la liste d'installation pour les casques immersifs.

En résumé :

- Activer le mode développeur : Accédez à Paramètres → Mise à jour et sécurité → Pour les développeurs
- Télécharger et installer Windows 10 Fall Creators Update
- Télécharger et installer Windows 10 Fall Creators Update SDK
- Télécharger et installer Visual Studio 2017. Lors de l'installation, sélectionnez les charges de travail suivantes : "Développement de la plate-forme Windows universelle", "Développement de jeux avec Unity" et "Développement de bureau avec C ++"
- Télécharger et installer HoloLens Emulator et les modèles holographiques
- Télécharger et installer Unity 2017 et sélectionnez "Windows Store .NET Scripting Backend" lors de l'installation
- Mettre à jour les pilotes graphiques

Annexe D

Questionnaire Utilisateur

Questionnaire

Questionnaire sur l'utilisation de l'application HoloDoc.

*Obligatoire

Vous êtes : *

un homme une femme

Quel âge avez-vous ? *

moins de 18 ans entre 18 et 30 ans plus de 30 ans

Interface

L'interface vous paraît : *

	pas	peu	plutôt	assez	très
lisible	<input type="radio"/>				
réactive	<input type="radio"/>				
fluide	<input type="radio"/>				
intuitive	<input type="radio"/>				

Accessibilité de l'interface *

	Très difficile	Plutôt difficile	Moyen	Plutôt facile	Très facile
Gestes	<input type="radio"/>				
Commandes vocales	<input type="radio"/>				

Avez-vous pu travailler au rythme souhaité ? *

	1	2	3	
Non	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Oui

Que pensez vous des méthodes de saisie proposées ?

	Inutile	Très peu utile	Correct	Utile	Très utile
Le clavier virtuel	<input type="radio"/>				
La saisie vocale	<input type="radio"/>				

Que pensez-vous des commandes vocales ? *

	1	2	3	
Inutile	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Utile

Que pensez-vous des retours visuels ? *

	1	2	3	4	5	
Insuffisants	<input type="radio"/>	Suffisants				

Que pensez-vous des retours sonores ? *

	1	2	3	4	5	
Insuffisants	<input type="radio"/>	Suffisants				

Remarques sur l'interface

Votre réponse

Fonctionnalités

Avec les geste, avez-vous réussi à : *

	Oui	Non
Photographier un document	<input type="checkbox"/>	<input type="checkbox"/>
Lier/Délier deux documents	<input type="checkbox"/>	<input type="checkbox"/>
Ouvrir/Fermer un document	<input type="checkbox"/>	<input type="checkbox"/>
Ouvrir/Fermer le panel	<input type="checkbox"/>	<input type="checkbox"/>
Mettre à jour la photo d'un document	<input type="checkbox"/>	<input type="checkbox"/>

Avec les commandes vocales, avez-vous réussi à : *

	Oui	Non
Lier/Délier deux documents	<input type="checkbox"/>	<input type="checkbox"/>
Ouvrir/Fermer un document	<input type="checkbox"/>	<input type="checkbox"/>
Mettre à jour la photo d'un document	<input type="checkbox"/>	<input type="checkbox"/>

Remarques sur les fonctionnalités

Votre réponse

Globale

Quel est votre impression générale sur : *

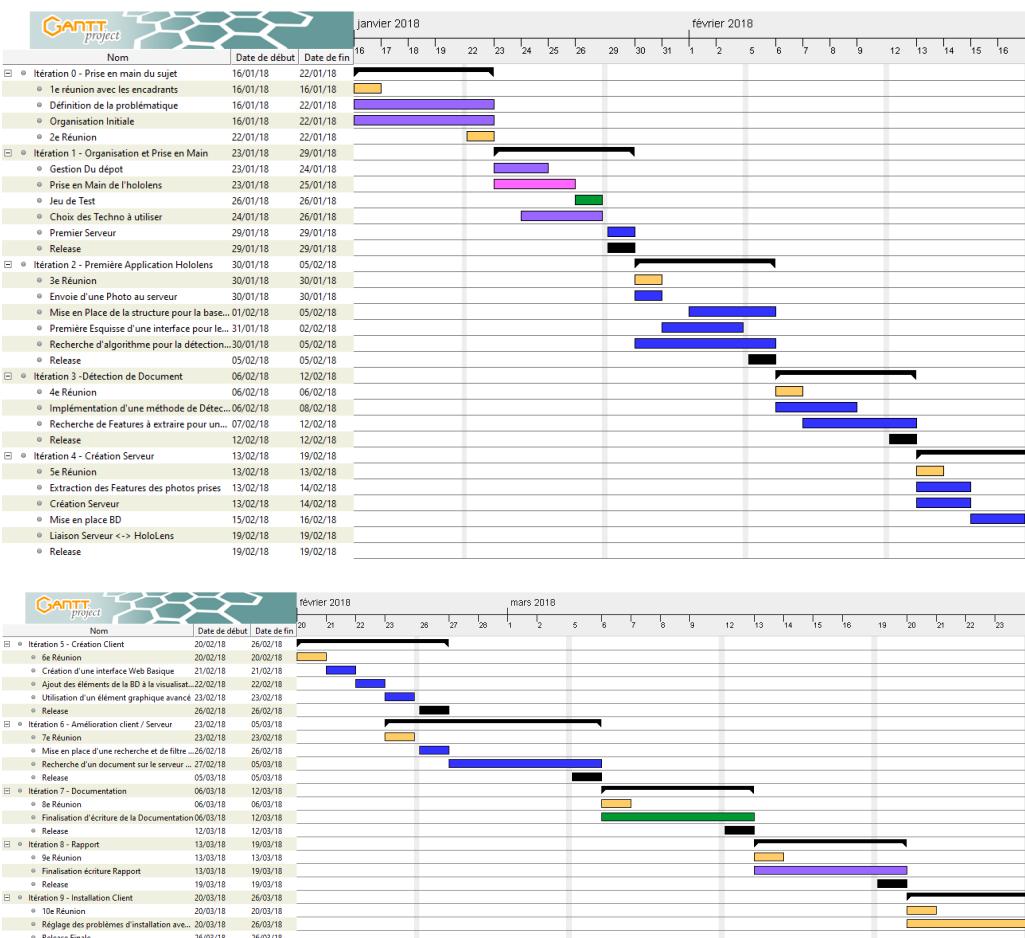
	Très Désagréable	Désagréable	Correcte	Agréable	Très agréable
Les retours visuels	<input type="radio"/>				
Les retours sonores	<input type="radio"/>				
L'interface	<input type="radio"/>				

Remarques sur l'impression globale de la solution

Votre réponse

Annexe E

Gantt prévisionnel



Annexe F

Gantt réel

