

UNIVERSITÉ DE BORDEAUX - REALITY TECH

**Mémoire de stage
Master 2 Informatique
Image et son**

Auteur :
Nicolas PALARD

Client :
Jérémy LAVIOLE
Référent :
Vincent LEPESTIT



Résumé

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis. Curabitur aliquet pellentesque diam. Integer quis metus vitae elit lobortis egestas. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat. Proin feugiat, augue non elementum posuere, metus purus iaculis lectus, et tristique ligula justo vitae magna. Aliquam convallis sollicitudin purus. Praesent aliquam, enim at fermentum mollis, ligula massa adipiscing nisl, ac euismod nibh nisl eu lectus. Fusce vulputate sem at sapien. Vivamus leo. Aliquam euismod libero eu enim. Nulla nec felis sed leo placerat imperdiet. Aenean suscipit nulla in justo. Suspendisse cursus rutrum augue. Nulla tincidunt tincidunt mi. Curabitur iaculis, lorem vel rhoncus faucibus, felis magna fermentum augue, et ultricies lacus lorem varius purus. Curabitur eu amet.

Table des matières

1	Introduction	1
1.1	Contexte et cadre	1
1.2	Objectifs	2
2	Notions	3
3	État de l'art	7
3.1	PapARt	7
3.2	Systèmes de réalité augmentée spatiale	8
3.3	Bilan	8
4	Développement d'application	9
4.1	ReARTable	9
4.1.1	Besoins de l'application	10
4.1.2	Choix et implémentation	10
4.2	Extraction de document	11
4.2.1	Besoins de l'application	11
4.2.2	Choix et implémentation	12
5	Prototype haute performance	15
5.1	Amélioration logiciel	15
5.1.1	Convolution - Théorie	15
5.1.2	Convolution - Optimisation	17
5.2	Matériel	19
5.3	Nectar - Architecture micro services	19

Introduction

Ce mémoire retracera les missions réalisées durant mon stage de Master 2 Informatique pour l'Image et le Son à l'Université de Bordeaux 1 effectué entre Avril et Septembre 2018 (6 mois) dans la société **RealityTech**¹. Ce rapport ne couvrira cependant que les 5 premiers mois du stage car la date de rendu de ce dernier précède d'un moi la date de fin du stage.

Le stage a donc été effectué chez **RealityTech** une jeune start-up de réalité augmentée spatiale. Issue de l'Inria de Bordeaux, l'institut national de la recherche en information et en automatique, cette dernière est la continuité d'un projet de recherche mené par M. Laviole, ex ingénieur de recherche à l'Inria. Ce projet, PapARt² Paper Augmented Reality ToolKit, est un kit de développement (SDK) Processing permettant de créer des expériences de réalité augmentée sous forme d'applications de projection interactive dans des feuilles de papier. Pour réaliser ce type d'application du matériel spécifique est nécessaire tel que des caméra couleurs, des caméra de profondeur qui sont utilisées pour capter le monde réel, un projecteur utiliser pour pouvoir visualiser le contenu numérique dans l'espace et un ordinateur pour effectuer tous les calculs nécessaire au bon fonctionnement des Applications. C'est dans un premier temps du côté matériel que **RealityTech** intervient. En effet, celle ci propose des systèmes pré calibré et prêt à l'emploi résolvant donc tous les problèmes de calibration et d'installation de l'environnement nécessaire à l'utilisation de PapARt. En parallèle de la partie matériel, la société développe et améliore activement le kit et créer aussi diverses applications de démonstration, afin de montrer et d'étendre les possibilités de la technologie. Le but est de pouvoir ouvrir la technologie à d'autres domaines d'activités que celui de la recherche pour mettre les outils collaboratifs et systèmes interactifs aux services de l'éducation, du vidéo ludique, de la vulgarisation etc....

Contexte et cadre

Actuellement, **RealityTech** se développe dans un incubateur de start-up appelé **La Banquiz**³ situé 4 rue Eugène et Marc Dulout, à Pessac Centre. L'objectif de **La Banquiz** est de promouvoir des start-ups Open Source⁴ et innovantes en apportant à leurs dirigeants des formations, du coaching individuel et collectif, de l'aide pour la recherche de financement et tout ce qui gravite autour de l'accompagnement de jeunes entreprises.

À ce jour **RealityTech** ne travail qu'avec des laboratoire de recherche tel que l'Inria et cherche à étendre son secteur d'activité. Les systèmes et services proposés par la société fournissent les résultats espérés et la dynamique de celle ci s'oriente donc vers une industrialisation du produit. Ainsi l'entreprise souhaite passer de prototype FabLab créé à l'unité basé sur des technologies de recherche, à prototype industriel basé sur des technologies populaires dans le monde de l'industrie. **RealityTech** se lance donc dans la création d'une plateforme haute performance appelé Nectar et dans le développement d'un nouvel SDK Unity⁵ utilisant cette plateforme. Le SDK Processing actuel est très accessible et permet de prototyper bon nombre d'applications rapidement mais il ne permet cependant pas de répondre aux besoins de l'industrie plus spécialement quand il s'agit de développer des applications client ayant besoin d'un moteur 3D, ou d'un moteur physique performant.

Pour débuter ce stage, j'ai eu l'occasion de partir à Laval pendant une semaine pour assister au Laval Virtual, le plus grand salon international sur la réalité augmentée et virtuelle, en tant qu'exposant. Grâce à ce salon, j'ai pu développer une bonne connaissance du produit et ai pu observer de nombreuses

1. <http://rea.lity.tech/>

2. <https://project.inria.fr/papart/fr/>

3. <http://labanquiz.com>

4. Open Source - Wikipédia

5. <https://unity3d.com/fr>

technologies à l'état de l'art dans ces domaines. C'est aussi ce salon qui m'a permis de bien comprendre les besoins auxquels pouvait répondre une technologie comme celle de RealityTech et par la même occasion les enjeux et les apports de celle ci. Cela à aussi été une très bonne expérience au niveau relationnel car elle m'a permis de créer rapidement une relation avec M. Laviole, mon tuteur de stage.

Le reste de mon stage a été réalisé à La Banquiz où j'ai été au contact de toutes les entreprises officiant en son sein. J'ai notamment pu rencontrer d'autres stagiaires tel que Rémi Kressmann, développeur, et Gabin Andrieux, commercial chez LockEmail⁶, une entreprise de cyber-sécurité, mais aussi des dirigeants comme Jean François Schaff, docteur en physique quantique ayant créer Postelo⁷ une plateforme de télé expertise pour les professionnels de santé, avec qui j'ai énormément échangé aussi bien sur des concepts de programmation, que sur la culture de l'informatique en générale.

Problématique du sujet

Objectifs

Le déroulement du stage a été fortement guidé par les besoins de la société.

Applications de démonstration Le premier gros objectif du stage était le développement d'applications de démonstration en utilisant le produit de l'entreprise. Le but était de comprendre l'essence, le fonctionnement global du produit et ce qu'il était possible/impossible de réaliser avec celui ci. Cet objectif m'a permis d'acquérir à la fois une vision globale de l'architecture logiciel et du fonctionnement interne du kit de développement, et de l'architecture matérielle nécessaire a l'utilisation du kit. En développant ces applications de démonstration, j'ai acquis une vision globale du projet qui m'a permis d'avoir une certaine autonomie assez rapidement

Plateforme haute performance Le deuxième objectif était de réaliser une preuve de concept haute performance du produit. En effet, comme expliqué dans la partie sur le contexte (sec. 1.1), l'entreprise se lançait dans le développement d'une nouvelle plateforme haute performance. Aussi bien au niveau matériel, ordinateur, caméra, projecteur, que logiciel, algorithmes, communication inter processus, accès au matériel, il a fallut réaliser des tests complets.

Kit de développement Comme expliqué dans le cadre, les besoins auxquels répond le kit de développement Processing ne sont plus suffisants lorsqu'il est question d'applications devant faire le rendu de grosse scène 3D, ou des simulation physique,. Le développement d'un plugin Unity permettant de créer des applications de projection interactive était donc nécessaire pour la suite du développement de RealityTech. L'objectif était d'intégrer et d'utiliser les micro services du prototype haute performance Nectar dans Unity pour gérer tout ce qui concerne le monde physique (caméras, projecteurs, suivi d'objet, touch) et d'utiliser Unity dans son rôle de moteur 3D, moteur physique pour gérer le rendu de la scène, la physique des objets dans la scène, la lumière etc et créer des expériences de projection encore plus évolués. Le plugin Unity permet aussi d'ouvrir la technologie de RealityTech à un plus grand nombre d'utilisateur car c'est le moteur le utilisé actuellement.

6. <http://www.lockemail.fr/>

7. <https://www.postelo.fr/>

Notions

Le domaine d'activité qui entoure ce stage est très riche en termes de notions et de vocabulaire. Afin de mieux comprendre de quoi il va être question tout au long de ce rapport, il est nécessaire d'en définir les notions de base.

Réalité virtuelle La réalité virtuelle plus communément appelé *Virtual Reality (VR)* désigne l'ensemble des environnements purement numériques (fig 2.1), qu'ils soient réalistes ou non, dans lesquels aucune interaction avec l'environnement réel n'est possible et inversement. Cette réalité se base très généralement sur un casque *Head Mounted Display (HMD)* dont l'utilisateur doit se munir afin d'être immergé dans un monde numérique avec lequel il peut interagir. Dans la réalité virtuelle, l'immersion est une notion importante lorsqu'il s'agit de la différenciée d'un simple programme informatique.



FIGURE 2.1 – Représentation de continuum de la virtualité par Milgram et Kishino, 1995[?]

Réalité augmentée La réalité augmentée plus communément appelé *Augmented Reality (AR)* quant à elle est un sous domaine de la réalité virtuelle. L'idée de la réalité augmentée est de venir superposer à l'environnement réel des éléments virtuels. Ces éléments vont alors venir "augmenter" notre monde en apportant le plus souvent des compléments d'informations. Elle est donc qualifiée de sous domaine de la réalité virtuelle car l'utilisateur n'est plus immergé dans un environnement complètement numérique mais du contenu virtuel est ajouté en contexte à la vision réelle. Par abus de langage le terme de réalité augmentée est souvent utilisé pour parler de réalité mixte dont la notion est détaillé dans cette partie. Il faut noter que ce type de réalité ne se base pas uniquement sur des *HMD* mais peut être aussi apprécié à l'aide d'un téléphone par exemple (fig 2.2).



FIGURE 2.2 – Réalité augmentée vu au travers d'un téléphone¹

Réalité mixte La réalité mixte, ou hybride, plus communément appelé *Mixed Reality (MR)*, ou *Crossed Reality (XR)*, est la fusion parfaite de l'environnement numérique et de l'environnement physique (fig 2.1). Dans ce "nouvel" environnement, les objets physiques et numériques coexistent et peuvent interagir entre eux et par exemple une table peut devenir une plateforme pour un personnage virtuel (fig 2.3). Souvent confondu avec la réalité augmentée, cette dernière se différencie car elle ne propose pas seulement une visualisation des objets numériques, elle propose aussi des méthodes d'interactions avec ce contenu et c'est cette notion d'interaction qui permet de la différencier. A l'heure actuelle la réalité mixte nécessite un dispositif de type *HMD* pour être appréciée comme par exemple l'*HoloLens* de Microsoft.

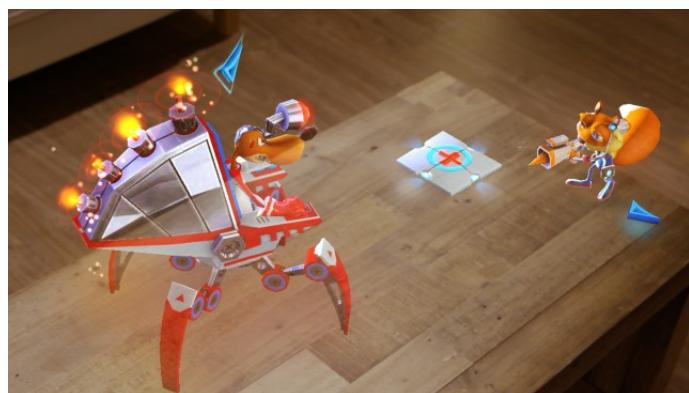


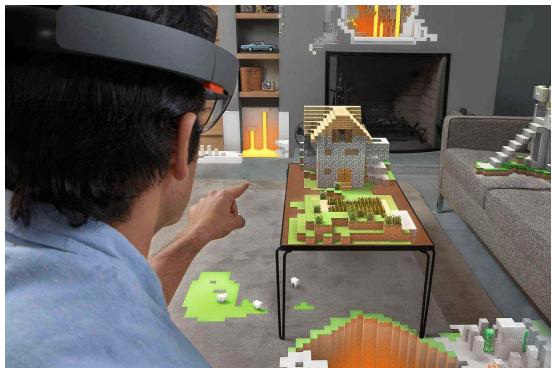
FIGURE 2.3 – Asobo Studio™- Young Conker©

Réalité augmentée vue au travers La réalité augmentée vue au travers, plus communément appelé *See Through Augmented Reality (STAR)* est une technique de visualisation de la réalité augmentée où les éléments numériques sont vu au travers d'un écran (fig 2.4a) ou d'un *HMD* (fig 2.4b). C'est le type de visualisation le plus utilisé actuellement. L'un des défauts majeur de ce type de visualisation est que la plus part du temps, chaque utilisateur a besoin de son propre écran ou casque pour pouvoir en profiter pleinement ce qui limite grandement les expériences collaboratives. Aussi les principaux défauts liés aux écrans s'appliquent aussi, a savoir fatigue visuel etc.

1. Source : <https://www.engadget.com>



(a) Pokémon GO - Vue au travers téléphone²



(b) Microsoft HoloLens - Vue au travers casque³

FIGURE 2.4 – Réalité augmentée vue au travers

Réalité augmentée spatiale La réalité augmentée spatiale, plus communément appelé *Spatial Augmented Reality (SAR)* est une technique de visualisation de la réalité augmentée se basant sur un dispositif de projection. Les éléments virtuels qui viennent "augmenter" le monde réel sont alors projetés dans l'espace (fig 2.5), d'où le terme spatial. Cette notion d'augmentation de l'espace tend à rendre cette technologie naturellement collaborative car les projections ne dépendent pas d'un dispositif visuel personnel et sont obligatoirement partagées. La SAR permet aussi de favoriser le développement d'interface tangible, en effet, la visualisation se faisant directement sur les objets physiques, la tendance à développer des interfaces en communion avec ceux ci est très forte car très naturel.



FIGURE 2.5 – Présentation d'une voiture en utilisant la réalité augmentée spatiale⁴

Interface tangible Une interface utilisateur tangible ou *Tangible User Interface (TUI)* est une interface utilisateur via laquelle des objets physiques, ou encore le toucher, permettent de manipuler des données numériques (fig 2.6b). Les interfaces utilisateurs tangibles remplacent très souvent les interfaces utilisateur graphiques (fig 2.6a) où *Graphical User Interface (GUI)* dans la plupart des application de réalité augmentée car elles fournissent un contrôle direct à l'utilisateur sur ce qu'il souhaite manipuler (par opposition au contrôle indirect, comme la souris, nécessaire à la manipulation des GUI).

-
- 3. Source : Pokemon GO
 - 3. Source : Microsoft HoloLens
 - 4. Source : Google Image

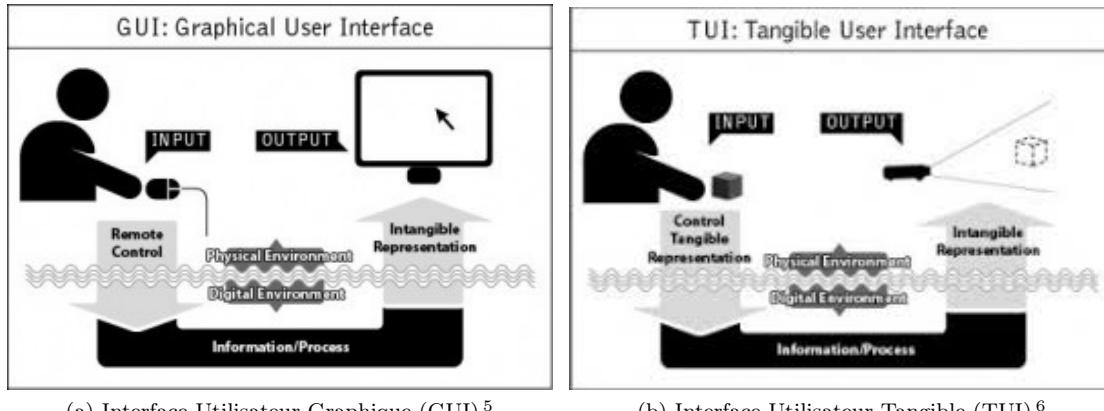


FIGURE 2.6 – Différences des interfaces utilisateurs

Calcul haute performance Le calcul haute performance ou *General-Purpose computing on Graphics Processing Units (GPGPU)* désigne une méthode de calcul utilisant la carte graphique (GPU) plutôt que le processeur (CPU). Cette technique permet de bénéficier de la puissance de la carte graphique afin de réaliser du calcul en parallèle et est très souvent utilisée pour la plupart des traitement lourd comme par exemple le rendu d'une scène 3D, l'encodage de vidéo, les simulations physiques (particules) etc. Cette technique repose sur le grand nombre de coeurs présent dans les cartes graphiques (contrairement aux processeurs) et sur la capacité de chacun de ces coeurs à effectuer des opérations simples de manière très efficace. Le calcul haute performance ne peut cependant pas se passer du CPU qui va être principalement utilisé pour récolter et transférer les données traitées ou à traiter.

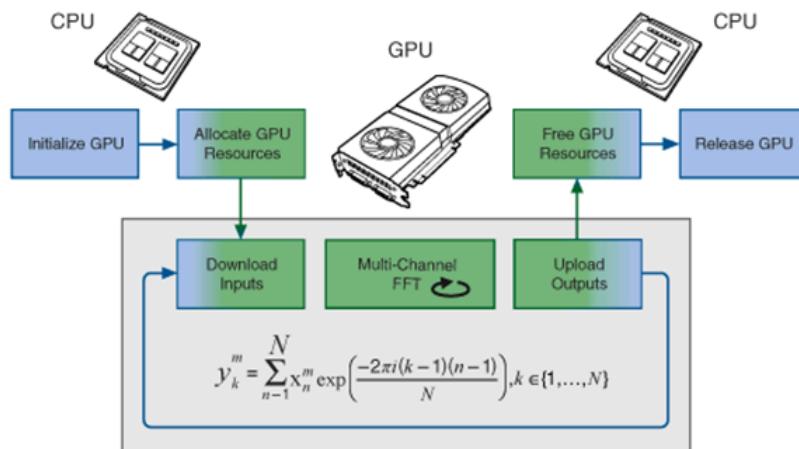


FIGURE 2.7 – Exemple de calcul de la FFT sur GPU⁷

6. Source : Icon Library - From GUI to TUI

6. Source : Icon Library - From GUI to TUI

7. Source : National Instruments

État de l'art

Intro

PapARt

Comme décrit dans l'introduction chapitre 1, PapARt ou Paper Augmented Reality Toolkit se présente sous la forme d'un kit de développement permettant de créer des applications interactives en réalité augmentée de création de dessins ou de peinture (fig 3.1). L'idée est de proposer une technique numérique non intrusive pour faciliter une tâche complexe, tel que le dessin tout en permettant à l'utilisateur de s'exprimer.



FIGURE 3.1 – Jeremy Laviole utilisant PapARt pour dessiner en réalité augmentée¹

Le système interactif (fig 3.2) permettant d'utiliser tout le potentiel de PapARt est très spécifique.

Il se compose de 2 dispositifs d'acquisitions, une caméra couleur observant la zone de travail pour venir y détecter les feuilles de papier qui serviront de base à la projection. Les feuilles de papier détectées par PapARt sont ornés de marqueurs ARToolKitPlus², est une bibliothèque de détection de marqueurs fiduciaires³. Ces marqueurs une fois détectés permettent d'estimer assez précisément la position de la feuille. Une fois la feuille détectée, PapARt se charge ensuite d'interpréter les marqueurs pour y projeter le contenu adéquat comme par exemple un dessin ou un menu.

Le deuxième dispositif d'acquisition est une caméra de profondeur qui a pour rôle de détecter les différents utilisateurs et les potentiels interactions. Grâce aux informations de profondeur, les interactions peuvent être détecté soit sur le plan de la zone de travail, ce sont des interactions qualifiées de "touch", soit dans l'espace au dessus de la zone de travail, qu'on qualifiera de "pointage 3D". En plus de ces deux caméra, un projecteur est présent pour gérer toute la partie visualisation. Son rôle est de projeter dans les zones adéquates (i.e détectées via des feuilles de marqueur) le contenu numérique désiré.

Pour avoir une représentation à l'échelle comme mentionné, il est nécessaire d'avoir un calibration caméras/projecteur très précise. Cette calibration très précise permet aux système complet d'avoir des

1. Source : Inria - PapARt

2. <https://github.com/paroj/ar toolkitplus>

3. Un marqueur fiduciaire est un objet placé dans le champ de vision le plus souvent de système d'imagerie qui apparaît sur l'image produite et qui va servir de point de repère ou de référence.

capacités d'interaction et de manipulation (toucher, balayage, balayage à deux doigts) d'une tablette tactile.

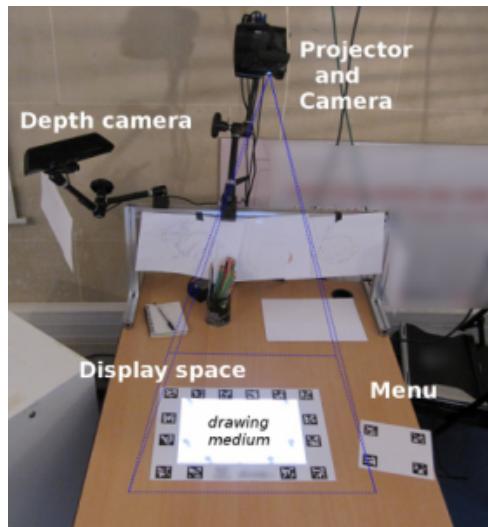


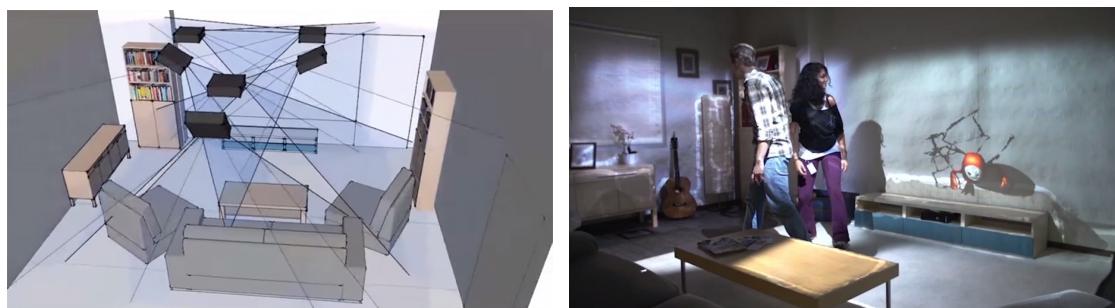
FIGURE 3.2 – Système interactif utilisant PapARt⁴

Au-delà des applications d'aide au dessins qui sont extrêmement nombreuses, PapARt ouvre un champ des possibles assez large. Le rôle de RealityTech est d'explorer ce champ des possibles en améliorant PapARt et en fournissant de nouveaux cas d'utilisation toujours plus innovants.

Systèmes de réalité augmentée spatiale

RoomAliveToolKit RoomAliveToolKit[?] est un projet tout droit sorti des laboratoires de recherche de Microsoft. RoomAliveToolKit est un kit de développement créé en 2013 par Nikunj Raghuvanshi, Eyal Ofek et Andy Wilson qui permet, à l'instar de PapARt, de créer des expériences de projection interactive. La principale différence réside dans le fait que RoomAliveToolKit a pour but de donner vie à des pièces entières en utilisant plusieurs projecteurs et plusieurs caméras qui fonctionnent en unisson.

RoomAliveToolKit a permis entre autres de développer de nombreux projets basés sur la projection interactive tel que RoomAlive, Room2Room, IllumiRoom et bien d'autre.



(a) Système de projection interactif nécessaire à l'utilisation de RoomAliveToolkit

(b) RoomAlive - Démonstration

FIGURE 3.3 – Microsoft Research : RoomAliveToolkit et RoomAlive⁵

Bilan

4. Source : Inria - PapARt

5. Source : RoomAliveToolkit

Développement d'application

Comme expliqué dans l'introduction chapitre 1 le premier objectif du stage était le développement d'applications de réalité augmentée spatiale. Il était important pour commencer, d'évaluer les possibilités mais aussi les contraintes qu'offrait le kit de développement. Ainsi un travail d'analyse et de critique de l'API à été effectué en parallèle du développement d'applications.

ReARTable

D'après le contexte et le public ciblé par l'entreprise, il m'a paru intéressant de développer une démonstration a but à la fois ludique et éducatif. J'ai donc choisi de recréer une Reactable[?] proposé par la société du même nom en réalité augmentée spatiale.

La Reactable est un instrument de musique électronique permettant la génération de son en direct développé depuis 2003. Présenté sous forme d'une table interactive, le son est généré via des éléments tangibles (fig. 4.1) placés a sa surface.



FIGURE 4.1 – Élément tangible utilisé pour la génération d'un élément de synthétiseur sur la Reactable¹

Chaque élément tangible représente un élément de synthétiseur qu'il est possible de contrôler de plusieurs façon :

- La distance de l'élément par rapport a un autre élément. Cette propriété peut être utilisée pour contrôler, par exemple, l'interaction entre deux éléments.
- L'orientation de l'élément sur la table. Cette propriété peut être utilisée pour contrôler, par exemple, la fréquence de l'élément ce qui va avoir pour effet par exemple pour un battement de ralentir ou d'accélérer ce dernier.
- La disposition de l'élément. Cette propriété permet entre autre de combiner des éléments pour créer des nouveaux son plus riches et plus complexes.
- La position du doigt de l'utilisateur par rapport a un élément. On peut venir contrôler divers paramètre comme l'amplitude par exemple en venant faire graviter son doigt autour d'un élément. Ainsi, c'est en combinant plusieurs éléments entre eux avec différentes orientation et différentes dispositions que l'utilisateur va pouvoir peu a peu "construire" sa musique. Au delà de la détection des éléments tangible, la table est rétro éclairée et permet donc la visualisation en directe de la musique générée (fig. 4.2).

1. Source : Reactable : Elements tangibles



FIGURE 4.2 – Visualisation du son sur la Reactable²

Besoins de l'application

Le but de l'application était de proposer une démonstration ce de qu'est capable de faire le système proposé par RealityTech et non pas de créer une simulateur de musique en direct fini reprenant tous les points de la Reactable. Un tel développement pourrai faire l'objet d'un stage entier et ce n'était pas le cas ici.

Pour être en adéquation avec l'idéologie de l'entreprise, l'interface tangible et les modes d'interactions avec la musique était le point le plus cruciale. En gardant ça en tête nous avons défini les besoins fonctionnels principaux :

- Générer du son en direct.
- Créer une représentation physique du son. Chaque son ou élément sonore devait avoir une représentation physique qui lui était associé, c'est à dire, un élément ou groupement d'éléments tangibles représentant ce son.
- Déetecter des éléments physiques représentant les éléments sonores dans une image. L'application devait pouvoir détecter dans une image de caméra les divers éléments physiques présents de façon a ce qu'ils soient utilisés pour identifier les éléments sonores.
- Identifier les représentation physique des sons. Chaque élément sonore étant représenter par un ou plusieurs éléments physique, l'application devait être capable, à partir des résultats de la détection, d'identifier et de différencier des éléments sonores entre eux.
- Modifier un élément sonore. L'application devait pouvoir contrôler certains paramètre défini a l'avance de chaque élément sonore généré. Ces paramètres ont pour but d'apporter a l'utilisateur a un niveau de contrôle supérieur lors de la création de musique en direct.
- Déetecter des événements liés au toucher. Dans le cas du contrôle d'un son, l'utilisateur peut être amener a toucher des zones interactives pour déclencher divers effets.
- Crée une visualisation basique d'un son. L'application devait proposer une visualisation du son généré pour guider l'utilisateur dans son expérimentation.

Choix et implémentation

L'application a donc été développé avec Processing en utilisant PapARt pour la partie visualisation, détection et projection et Sonic Pi[?] pour la génération de musique en direct. Sonic Pi est un synthétiseur temps réel qui permet très facilement de générer des sons de manière cohérente. Le gros avantage de Sonic Pi est qu'il résout tout seul énormément de problèmes posé par la génération dynamique de musique comme par exemple la synchronisation des boucles, les effets d'entrée et de sortie des instruments et bien d'autre ce qui dé-complexifie énormément le processus.

Comme on peut le voir sur le schéma explicatif (fig 4.3), les éléments tangibles représentants des sons se présentent sous forme de regroupement d'éléments rond de petite taille (des aimants dans notre cas). L'idée derrière ce choix est d'encourager la manipulation d'élément physique pour garder le contenu

2. Source : Reactable

numérique en contexte et favoriser la création. On peut différencier deux sons en fonction du contenu du regroupement (nombre, position et couleur des éléments regroupés).

FIGURE 4.3 – ReARtable : Schéma général

Une fois les éléments détectés, regroupés et identifiés, l'élément sonore associé peut être créé. La création d'élément sonore se fait simplement via la transmission d'un message OSC³ à un serveur Sonic Pi préalablement démarré. Ce message contient l'identifiant unique de la boucle que Sonic Pi doit démarrer. Pour chaque éléments sonores que l'application peut créer Sonic Pi possède une fonction à exécuter que nous avons préalablement créé. Toutes les communications entre l'application et Sonic Pi utilisent ce protocole ce qui permet de démarrer/arrêter/modifier certaines parties du son en directe.

Pour ce qui est du contrôle du son, une zone autour du composant est défini dans laquelle soit un élément tangible, soit une interaction physique (avec le doigt) vont être détecté et converti en interaction avec le contenu numérique (fig. 4.4).

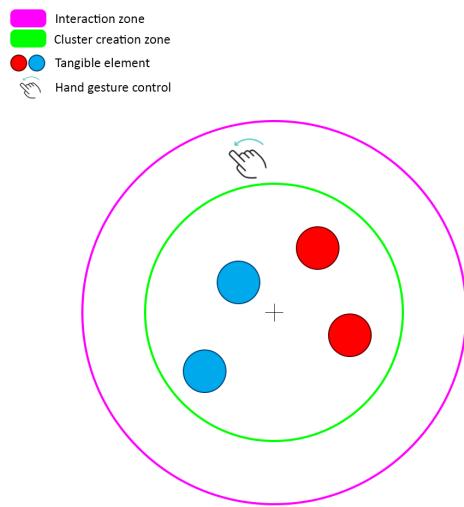


FIGURE 4.4 – Schéma représentant la création d'un son avec zone d'interaction

La dernière étape du développement de l'application était la visualisation de la musique générée (section 4.1.1). Cette étape n'a finalement pas été aboutie par manque de temps. L'idée était d'utiliser le spectre du son et les différentes fréquences qui le compose récupérable à l'aide d'une transformée de Fourier⁴ pour créer une visualisation globale basée sur les fréquences avec des variations visuelles en fonction de la hauteur, du tempo du son et tous les autres paramètres du son qu'on peut extraire.

Extraction de document

Plus tard au cours de mon stage des cas d'utilisation où l'extraction et la numérisation de document ont été abordés comme par exemple et ainsi il était judicieux de développer une preuve de concept de cette fonctionnalité sous la forme d'une application de SAR.

Le but de ce développement était d'expérimenter diverses techniques de détection de document en temps réel se basant ou non sur des connaissances à priori comme la taille du document, sa couleur, la couleur du fond (duquel il faut extraire le document), la présence d'éléments distinctifs (comme des marqueurs fiduciaires ou des ronds colorés de petite taille).

Besoins de l'application

- Accéder au flux vidéo d'une caméra. L'application devait avoir accès au flux vidéo d'une caméra filmant le document à détecter.

3. Le protocole OSC où OpenSoundControl est un format de transmission de données conçu pour le contrôle en temps réel

4. Opération mathématique permettant de décomposer un signal en la somme des signaux qui le compose Wikipédia - Transformation de Fourier.

FIGURE 4.5 – Document à détecter muni de marqueurs colorés

- Déetecter un document dans une image. Des images extraites du flux vidéo, l'application devait être capable, avec ou sans connaissance a priori, de détecter un document se trouvant dans cette image.
- Extraire un document d'une image. Grâce au résultat de la détection, l'application devait être capable d'extraire ce document de l'image afin d'obtenir une image ne contenant que le dit document.

Choix et implémentation

La détection de document est un problème connu en traitement d'image sur lequel j'avais déjà eu l'occasion de travailler lors de mon projet de fin d'études durant le deuxième semestre de mon année de Master 2.

Dans cette application nous avons mis en place plusieurs détection différentes pour essayer de trouver une solution à ce problème.

Détection de document basé sur des marqueurs colorés La première détection utilise une connaissance a priori sur le document : Le document cible est muni de lignes d'éléments ronds colorés de petite taille dans un ou plusieurs de ses coins (fig. 4.6a). Ainsi tout l'enjeu de cette détection se base dans la détection de ces éléments colorés dont il est question plus tard dans ce rapport (voir section ??).

Une fois les éléments détectés, ils sont regroupés en différentes lignes (fig. 4.6b). Une ligne est défini par un regroupement d'éléments dont l'écart entre chaque éléments ne dépasse pas une certaine distance verticale ou horizontale. L'angle de la ligne est défini par les deux premiers éléments qui la compose. Si un autre élément "dérive" il est rejeté et la ligne est créée. Cette ligne est ensuite utilisé pour calculer deux vecteurs, dont un est confondu avec la ligne et le deuxième est perpendiculaire au premier (fig 4.6c). La détection finale du document se fait en calculant l'intersection des différents vecteurs verticaux et horizontaux (fig. 4.6d)

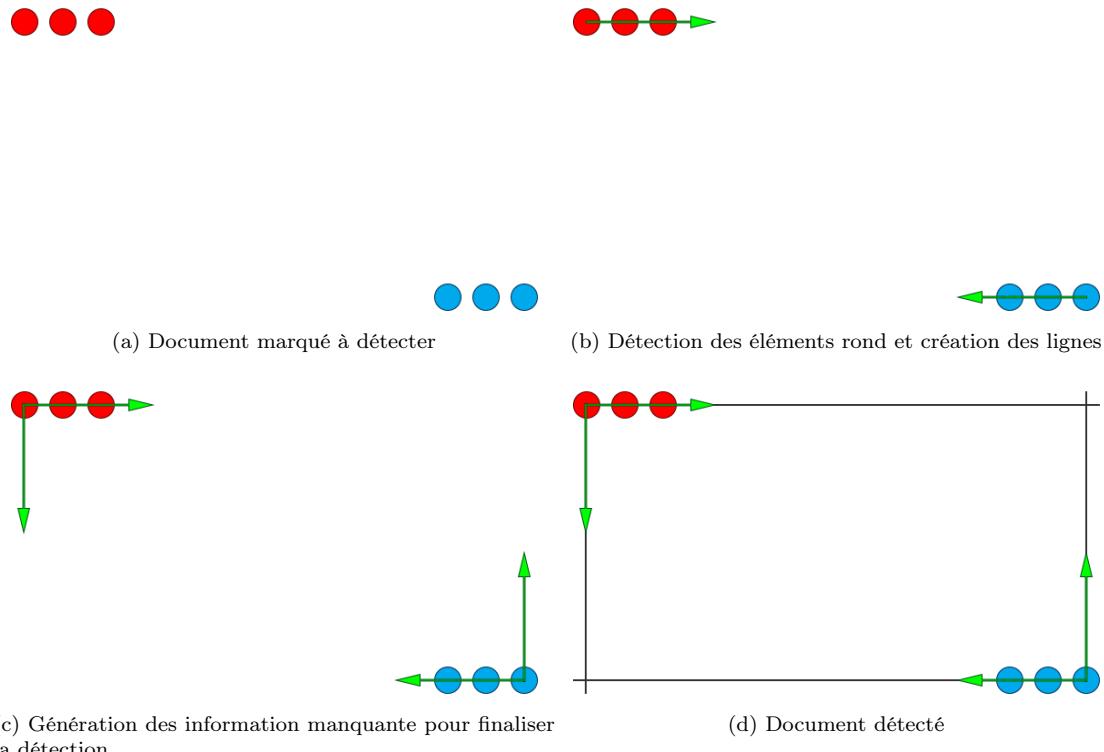


FIGURE 4.6 – Détection de document étape par étape

Comme on peut le voir figure 4.6d, lors de cette détection les bords du document sont rognés. Généralement ces zones ne contiennent aucune information car elles correspondent généralement aux

marges verticales et horizontales que chaque document possède. Cependant comme évoqué dans les cas d'utilisation, cette détection ne sera pas utiliser seulement pour des documents de type A4, on pourra s'en servir comme outil pour suivre une feuille de papier à la manière des marqueurs ARToolKit, ou pour détecter des post-it par exemple. Ainsi ces marges ne peuvent pas être ignoré dans notre cas.

Une simple connaissance a priori de la distance des éléments rond par rapport au coin permet de résoudre ce problème ou alors pour avoir une détection bien plus précise et robuste, il est possible d'utiliser des algorithmes de détection de contour couplé à des algorithmes d'extraction de lignes pour retrouver le vrai coin du document. C'est ce dont nous allons parler dans la deuxième partie.

Détection de document - Canny et transformée de Hough Sans connaissance a priori la détection de document devient un problème compliqué et bien connu du monde de l'informatique surtout lorsqu'il y a un besoin de temps réel vient s'ajouter la tâche, comme par exemple, dans une application de scan de document.

L'algorithme de Canny[?] est un filtre de détection de contour permettant d'extraire d'une image des contours (fig 4.7b) très précis respectant trois critères : La bonne détection, la bonne localisation et la clarté de la réponse. Ces trois critères en font un très bon choix dans le cadre de la détection de document ou la qualité et surtout la précision de la détection est importante pour ne pas rogner des bouts de document par exemple. Cet algorithme a cependant tendance à laisser beaucoup de bruit issue de faible contour tout de même détectés c'est pourquoi une étape de floutage visant à lisser les zones de faible contour est fortement conseillée.

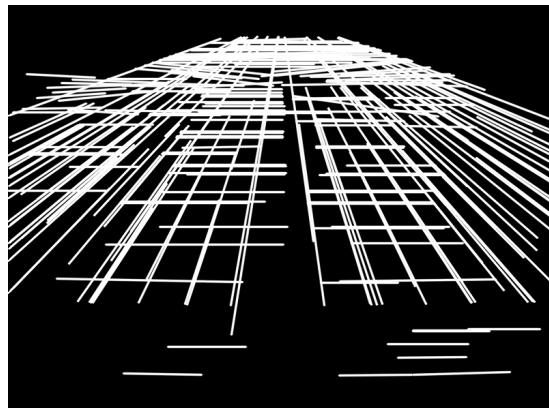
Une fois les contours détectés, il est possible d'essayer d'extraire directement le document mais c'est une tâche plus complexe, car elle requiert d'analyser les contours, qu'il est possible de faciliter en utilisant la transformée de Hough[?]. La transformée de Hough permet d'extraire n'importe quelle forme à partir d'une image contour en utilisant les propriétés mathématiques de la forme. Dans notre cas où nous souhaitons extraire des lignes droites les propriétés mathématiques utilisées correspondent aux coordonnées polaires considérés comme plus robuste que l'équation de la droite (fig 4.7c)



(a) Image originale⁵



(b) Canny - Détection de contours⁶



(c) Transformée de Hough - Détection de lignes⁷

FIGURE 4.7 – Détection de lignes : Canny + Hough

7. Source : <http://funvision.blogspot.com/2016/01/hough-lines-and-canny-edge-sobel.html>

7. Source : <http://funvision.blogspot.com/2016/01/hough-lines-and-canny-edge-sobel.html>

7. Source : <http://funvision.blogspot.com/2016/01/hough-lines-and-canny-edge-sobel.html>

Il ne reste qu'à filtrer les lignes pour trouver des potentiels documents dans une image.

Cette succession de traitement est cependant lourde et peut difficilement être effectué en temps réel sur des images de haute résolution. Dans notre cas, nous nous sommes servis de ces algorithmes seulement sur des parties d'image (de petite résolution) de façon à améliorer une première détection grossière effectuée par exemple, à l'aide de marqueurs colorés. Une fois la première détection effectuée, nous obtenons une position plus ou moins précise des quatre coins nécessaires à l'extraction du document. Nous utilisons cette information sur la position potentiel des coins pour extraire dans des sous images centrées sur ces coins (fig. 4.8a). Ensuite nous appliquons les deux algorithmes mentionnés plus tôt, à savoir, la détection de contour (fig. 4.8b) puis la détection de ligne (fig. 4.8c). Nous filtrons ensuite le résultat de la détection de ligne pour obtenir exactement une ligne verticale et une ligne horizontale. Une fois ces deux lignes trouvées, nous calculons les équations de droite (pente, et constante) associées pour pouvoir en calculer l'intersection et finalement trouver le coin dans cette sous image (fig. 4.8d).

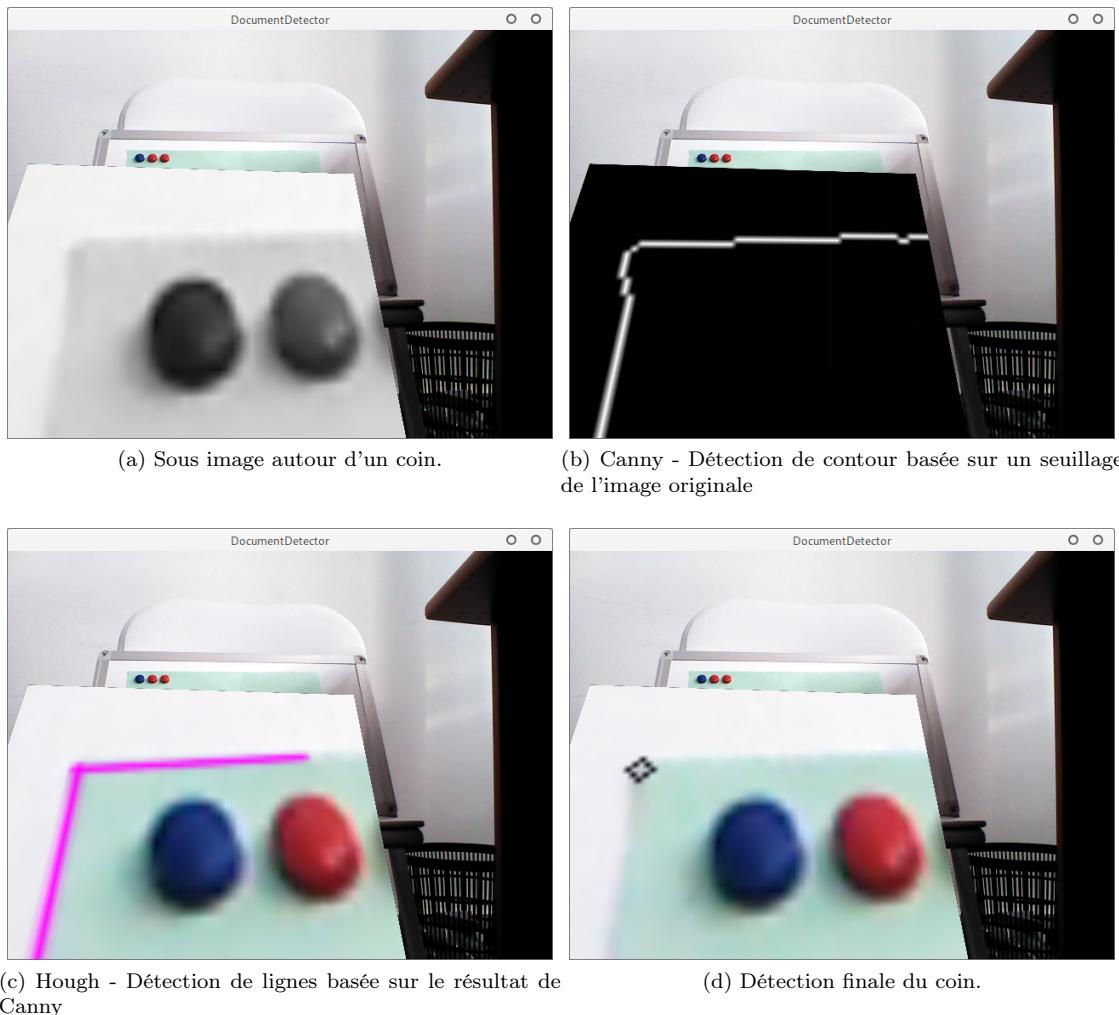


FIGURE 4.8 – Affinage de la détection d'un coin du document

Bilan Pour finir, vous trouverez Figure un rapide comparatif des différents résultats obtenus. On peut observer quelques points notables : En utilisant Hough et Canny pour améliorer la détection d'un coin, l'extraction est plus fidèle au document réel. En effet, les coins détectés sont plus précis que des hypothèses effectuées à priori. Cette méthode est cependant moins rapide car elle requiert de nombreux calculs en plus. Lorsqu'il est utilisé en temps réel, on peut cependant observer que la détection avec connaissance à priori du modèle est bien plus robuste, car elle ne dépend pas d'une deuxième détection qui a beaucoup de chance d'échouer (seuillage, détection de contour, détection de ligne, intersection entre deux droites puis obtention finale du coin) ainsi la méthode à utiliser variera avec les cas d'utilisations. Par exemple dans le cadre d'une application de scan de document, une méthode d'extraction de document plus précise sera envisagée, mais dans le cadre d'une estimation de pose (3D)⁸ on pourra préférer une détection rapide et robuste.

8. Pose : Wikipédia

Prototype haute performance

Dans un cadre d'application comme celui de la réalité augmentée spatiale ou les interactions jouent un rôle majeure dans l'expérience de l'utilisateur la réactivité, la fluidité de l'expérience et la latence générale du système sont des points cruciaux qu'il est impossible de négliger. Pour pouvoir atteindre ces objectifs et pouvoir pousser les applications encore plus loin aussi bien dans l'interaction, dans le rendu ou dans le contenu, sans avoir besoin d'une puissance de calcul dépassant l'entendement, une optimisation aussi bien logicielle, matérielle ou architecturale est nécessaire. Cette optimisation a fait l'objet d'une grande partie de mon stage qui m'a amener a développer un prototype dit "haute performance" des outils que propose RealityTech. L'optimisation logicielle c'est surtout porté sur l'amélioration des performances des algorithmes de traitement d'image bien connus pour être extrêmement consommateur des ressources. Pour l'optimisation matérielle, la tâche a était un peu différentes et nous nous sommes attelé à effectuer des mesures et des calculs sur la puissance théorique du matériel, la latence réel des caméra ou encore la rapidité de l'encodage des flux vidéos pour arriver à établir les performances réelles qu'il nous était possible d'atteindre avec différentes combinaison de matériel. Le développement du prototype s'est achevé avec la création d'une nouvelle architecture logiciel en micro services dans le but de créer un environnement modulaire réactif ou les services peuvent mourir sans mettre en péril tout le système et ainsi améliorer grandement la qualité général des outils fournis.

Amélioration logiciel

De nos jour, les optimisations font l'objet de développements ciblés et très spécifique, se concentrant la plupart du temps sur l'amélioration d'un unique point cruciale d'un algorithme ou d'une application. Dans notre cas l'optimisation logiciel a surtout été effectué au niveau des algorithmes de traitement d'images omniprésent et indispensable a la technologie. La réalité augmentée spatiale a besoin du monde réel pour exister c'est pourquoi le matériel dispose de nombreux capteurs (caméras) pour l'analyser et que de nombreux algorithmes de traitement des données captées (images) sont mis en place. Après une rapide analyse du logiciel, il est indéniable que traitement le plus utilisé est la convolution d'une image par un filtre qui possède un nombre incalculable d'application et c'est pourquoi nous avons choisi de concentrer nos efforts sur l'optimisation de ce dernier.

Convolution - Théorie

*En mathématiques, le produit de convolution est un opérateur bilinéaire et un produit commutatif, généralement noté *, qui, à deux fonctions f et g sur un même domaine infini, fait correspondre une autre fonction $f * g$ sur ce domaine, qui en tout point de celui-ci est égale à l'intégrale sur l'entiereté du domaine (ou la somme si celui-ci est discret) d'une des deux fonctions autour de ce point, pondérée par l'autre fonction autour de l'origine — les deux fonctions étant parcourues en sens contraire l'une de l'autre (nécessaire pour garantir la commutativité).¹*

Dans le cadre du traitement d'image, le produit de convolution représente une technique de filtrage d'image visant à accentuer ou atténuer certaines caractéristiques ce celle ci comme la netteté, le flou ou les zones de fort gradient (les contours) par exemple (fig 5.1). Étant donné que nous travaillons avec des images définie par un nombre fini de pixels, la convolution d'une image est réalisée dans le domaine discret où f et g dans la définition mathématique représentent respectivement une image et le filtre qu'on souhaite lui appliquer. Le résultat de cette convolution est une nouvelle image.

1. Source : Produit de convolution - Wikipedia

On appelle filtre, ou noyau de convolution, une image (ou une matrice) généralement de petite taille définie en amont qui va être utilisée pour calculer la nouvelle valeur de chacun des pixels de l'image résultat. C'est la définition de ce dernier qui va décider du traitement appliquer à l'image.

Le calcul de la valeur d'un pixel dans l'image résultat se fait de la manière suivante : Le voisinage autour du pixel dont on souhaite calculé la valeur est pondéré par le filtre de convolution que l'on aura préalablement centré sur ce pixel. La nouvelle valeur du pixel représente la somme de toutes les valeurs précédemment calculées (voir algorithme 1).

Sur la figure 5.2 on souhaite calculer la nouvelle valeur du pixel positionné en 3,3 dans l'image d'origine (I). On sélectionne donc un voisinage de même taille que le filtre (K) centré sur ce pixel dont chaque élément va être multiplié par la valeur du filtre pour calculée la valeur du pixel 3,3 dans la nouvelle image soit :

$$I_{3,3} * K = 88 * 1/9 + 21 * 1/9 + 25 * 1/2 + 68 * 1/9 + 14 * 1/9 + 15 * 1/9 + 35 * 1/9 + 52 * 1/9 + 10 * 1/9 = 36$$

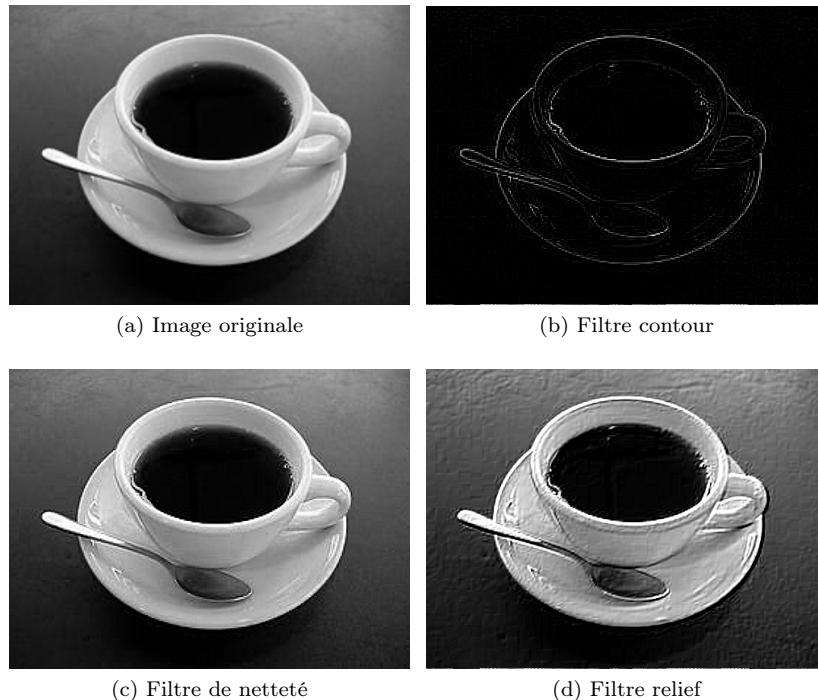


FIGURE 5.1 – Différentes filtres de convolution appliquée à une image.

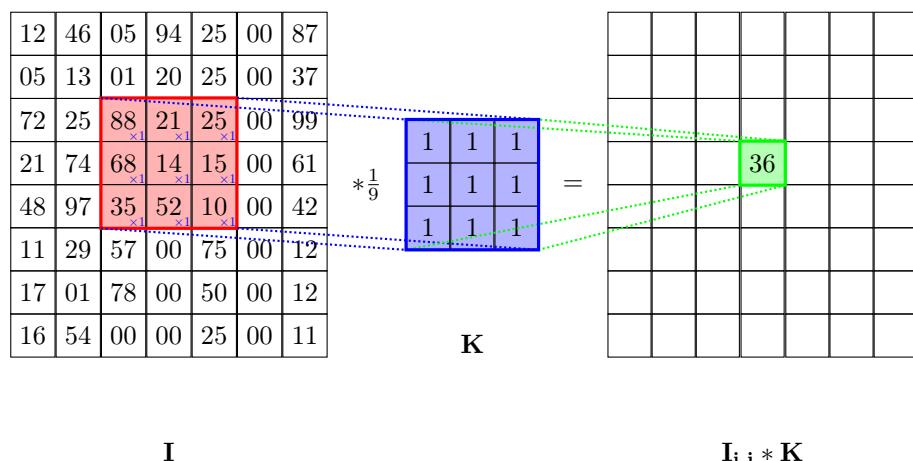


FIGURE 5.2 – Convolution d'une matrice (image) (I) par un filtre (K)

Algorithm 1 Convolution d'image image par un filtre

```
procedure CONVOLUTION(I, K, Iw, Ih, Ks)                                ▷ I : image, K : filtre
     $I_{conv} \leftarrow I$ 
     $Khs \leftarrow floor(Ks \div 2)$ 
     $x \leftarrow 0, y \leftarrow 0$ 
     $sum \leftarrow 0$ 
    for  $x \leq Iw; ++x$  do
        for  $y \leq Ih; ++y$  do
            for  $i \leq Ks; ++i$  do
                for  $j \leq Ks; ++j$  do
                     $pos_x \leftarrow x + i - Ksh$                                 ▷ Pour chaque éléments dans une fenêtre de taille  $Ks$ 
                     $pos_y \leftarrow y + j - Ksh$                                 ▷ pos = pos + position dans le voisinage
                    if  $outOf(I, pos_x, pos_y)$  then                                ▷ pos = pos + position dans le voisinage
                        continue
                    end if
                     $sum \leftarrow sum + I_{pos_x, pos_y} * Ki, j$                   ▷ Vérifie que les positions sont dans l'image (bords)
                     $maskSum \leftarrow maskSum + Ki, j$ 
                end for
            end for
             $I_{conv}, y \leftarrow sum \div maskSum$                                 ▷ Somme du voisinage par le filtre
        end for
    end for
    return  $I_{conv}$                                                         ▷ Valeur final = somme normalisée
end procedure
```

Comme on peut s'en rendre compte dans le pseudo code proposé (algo 1), l'image résultat est une nouvelle image, indépendante de l'image d'origine dont chaque pixel est calculé indépendamment de ces voisins dans cette nouvelle image. Cela signifie que n'importe quel pixel peut être calculé dans n'importe quel ordre. C'est précisément à cette propriété que nous allons nous intéresser car en théorie, avec une puissance de calcul suffisante il est possible de calculer en même temps tous les pixels de l'image résultat. Cet algorithme possède donc un très fort potentiel d'optimisation car il est très largement parallélisable.

Convolution - Optimisation

L'optimisation de cet algorithme peut se faire de deux façon bien distinctes. La première se fait en utilisant la puissance de la carte graphique de l'ordinateur pour effectuer énormément de calculs en même temps. C'est l'optimisation sur carte graphique dont nous avons évoqué le principe section 2. La seconde méthode d'optimisation consiste à légèrement changer l'algorithme de convolution : la convolution est séparé en deux filtres distincts[?], un horizontal et un vertical qui sont successivement appliqués à l'image origine. Nous avons choisi de n'effectuer que l'optimisation sur carte graphique car la deuxième méthode comporte de nombreux coûts et désavantages qui seront évoqués par la suite.

Pour pouvoir développer la dite optimisation, il a fallut utiliser un langage de programmation sur carte graphique. De nos jours, il en existe plusieurs et ils possèdent tous leurs spécificités cependant pendant la phase de recherche, 3 langages (ou sous langages) se sont démarqués : OpenCL[?], OpenGL ES[?] et CUDA[?]. Nous avons donc choisi d'implémenter 3 version de l'algorithme de convolution naïf (non séparé) utilisant chaque de ces langages et d'en évaluer les performances.

OpenCL OpenCL ou *Open Computing Language* est un langage de programmation basé sur le C créé par **Khronos Group** en 2009. Un programme OpenCL s'écrit en deux partie : La partie **code hôte** et la partie **noyau ou code périphérique** qui représentent respectivement la partie application se chargeant d'orchestrer les différentes tâches, la gestion mémoire, la gestion des périphériques s'exécutant sur l'hôte et la partie calcul permettant de compléter les dites tâches s'exécutant sur les périphériques. La partie hôte est écrite en C tandis que la partie noyau est écrit en OpenCL-C. Il faut donc aussi différencier hôte et périphérique (fig 5.3). Dans notre cas d'utilisation l'hôte représente le processeur et permet de transmettre les données au périphérique qui dans notre cas correspond à une ou plusieurs cartes graphiques.

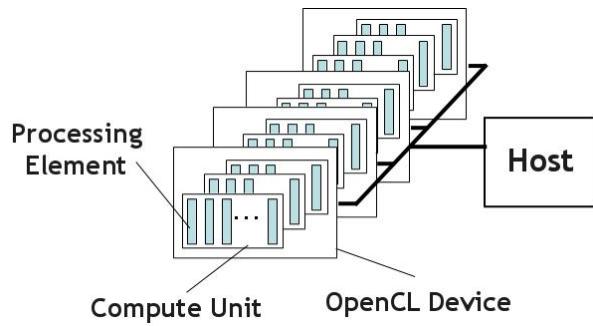


FIGURE 5.3 – Schéma OpenCL - Hôte et périphériques²

Nous nous sommes intéressé à OpenCL car il est compatible avec la plupart des systèmes et des architectures aujourd’hui présents sur le marché sans aucune modification de code nécessaire. Cet avantage est aussi l’un de ses plus gros inconvénients car il ne permet pas d’exploiter au mieux chaque architecture comme peut le faire CUDA avec NVIDIA, et les performances de ce dernier ne sont donc pas équivalentes sur chaque architecture.

OpenGL (ES) OpenGL est une interface de programmation multi plateforme et multi langage permettant faire le rendu de scène 2D/3D. En tant qu’interface il est possible de l’implémenter de façon logiciel mais elle a été conçue pour être implémentée de manière matérielle avec de profiter au mieux des accélérations matérielles disponibles. Ainsi c’est grâce à ces implémentations qu’OpenGL fournit un *pipeline* programmable de rendu ultra performant. C’est via ce pipeline programmable et plus spécifiquement via le code hôte et les shaders qu’il est possible de transmettre des instructions et des données à la carte graphique (fig ??)

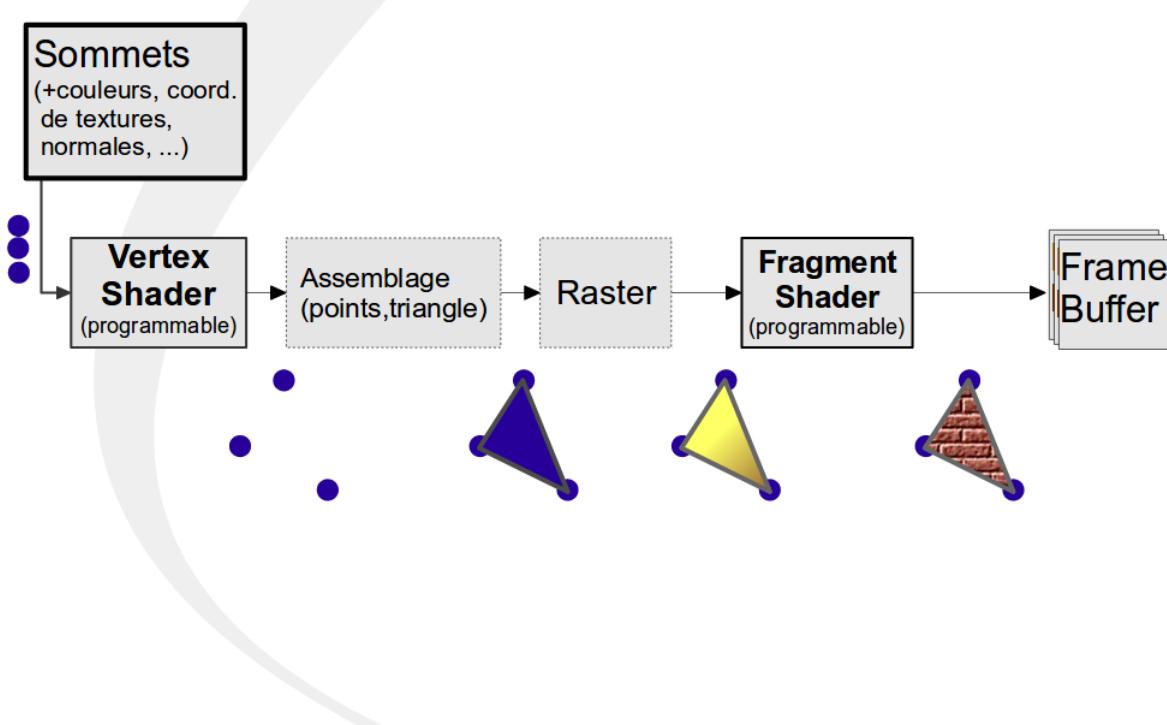


FIGURE 5.4 – Pipeline de rendu - OpenGL³

Le pipeline OpenGL reçoit en entrée :

- Des informations sur la géométrie de la scène.
- Des paramètres nécessaires pour effectuer le rendu de la scène. Point de vue de la caméra, lumières, textures, matériaux.

et donne en sortie une image de la scène.

2. Source : <https://www.anandtech.com/show/7334/a-look-at-alteras-opencl-sdk-for-fpgas/2>
 3. Source : Cours M1 Informatique - Mondes 3D - Pierre Benard

Pour pouvoir utiliser ce pipeline dans le but d'opérer des traitements sur des images 2D, il est nécessaire d'en détourner l'utilisation. Sans géométrie à fournir au vertex shader, le pipeline de rendu ne se déclenche pas. L'idée pour passer outre est de créer un bout de géométrie recouvrant l'écran, le plus souvent un quad, afin d'activer le pipeline. Une fois le pipeline activé, le vertex shader est programmé pour ne rien faire et ainsi les étapes d'assemblage et de rastérisation sont très rapidement passées pour arriver à l'étape du rendu par fragment. C'est dans ce shader que se compose l'image de sortie du rendu et c'est ici que nous avons accès à tous les pixels de l'image.

Matériel

Nectar - Architecture micro services