Multiview Autostereoscopic Displays

Bruno Mercier, Kévin Boulanger, Christian Bouville, Kadi Bouatouch

I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N°1868

MULTIVIEW AUTOSTEREOSCOPIC DISPLAYS

BRUNO MERCIER, KÉVIN BOULANGER, CHRISTIAN BOUVILLE, KADI BOUATOUCH

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Multiview Autostereoscopic Displays

Bruno Mercier[*] , Kévin Boulanger[**] , Christian Bouville[***] , Kadi Bouatouch[****]

Systèmes numériques
Projet Bunraku

Publication interne n1868 — Octobre 2007 — 53 pages

Abstract: This report describes the different technologies used par the manufacturers of autostereoscopic displays which are also called 3D displays. It also presents some existing 3D displays as well as the problems of resampling inherent in this kind of display. Next, it introduces more particularly the NEWSIGHT 3D displays as well as the interleaving of 8 views used by this kind of 3D display. The implementation of this interleaving on the CPU and on the GPU (programmable graphic cards) is also described.

Key-words: stereoscopy, autostereoscopy, multiscopy, automultiscopy, 3D display, immersive visualization, multiview

(Résumé : tsvp)

[*] Post-Doctorant, Equipe Bunraku, IRISA
[**] Doctorant, Equipe Bunraku, IRISA
[***] , Equipe Bunraku, IRISA
[****] Professeur d'Université, Equipe Bunraku, IRISA

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Les écrans multi-vues autostéréoscopiques

Résumé : Ce rapport décrit les différentes technologies utilisées par les constructeurs d'écrans autostéréoscopiques, appelés aussi écrans 3D. Il présente aussi certains systèmes de visualisation 3D ainsi que les problèmes de rééchantillonnage inhérents à ce genre de système. Ensuite, il introduit plus particulièrement les écrans 3D NEWSIGHT et le mode d'entrelacement de 8 vues utilisé par ce type d'écran ainsi que sa mise en oeuvre logicielle et matérielle utilisant les cartes graphiques programmables (GPU).

Mots clés : stéréoscopie, autostéréoscopie, multiscopie, automultiscopie, écran 3D, visualisation en relief, multi-vues

Table des matières

Première partie
Présentation générale

1 Introduction

Depuis une dizaine d'année, une nouvelle génération d'écrans est apparue : *les écrans 3D*. La construction de tels écrans est encore en phase expérimentale et la principale caractéristique des *écrans 3D*, les distinguant d'*écrans classiques*, est leur capacité à afficher des images en relief de manière autonome. Actuellement, aucune norme ne décrit les spécificités exactes de ces écrans, chaque constructeur définit son propre modèle avec ses propres caractéristiques.

Mais avant d'aller plus loin, posons-nous la question de la définition d'un *écran multi-vues autostéréoscopique*, également appelé *écran automultiscopique* ou plus familièrement *écran 3D*. Cette appellation regroupe trois notions différentes, à savoir :

- la stéréoscopie,
- les multi-vues,
- l'autonomie.

En effet, ces écrans permettent la diffusion d'images stéréo, i.e. la diffusion d'un point de vue différent pour chaque oeil de l'observateur. L'application du principe de la stéréoscopie permet à l'observateur de mieux évaluer la profondeur des éléments de la scène décrite sur l'écran et ainsi d'obtenir une meilleure perception de l'univers 3D projeté sur l'écran.

Habituellement, l'affichage d'une vue différente pour chaque oeil s'effectue à l'aide d'un dispositif spécial appliqué sur l'observateur comme par exemple des lunettes polarisantes. L'intérêt d'un écran autostéréoscopique, c'est qu'aucun matériel supplémentaire n'est nécessaire, l'observateur voit directement les images diffusées par l'écran en stéréoscopie. Ceci est rendu possible par un système de séparation des vues positionné devant l'écran : nous verrons plus en détail quel est ce dispositif dans la section 3.

Enfin ces écrans, appelés familièrement écrans 3D, sont également multi-vues car ils permettent non seulement d'afficher une vue différente pour chaque oeil (deux vues) mais d'afficher simultanément un ensemble de 4 à 9 vues différentes selon les modèles. Cette série de vues permet ainsi à l'observateur de se déplacer autour de l'écran pour observer la scène

affichée avec une immersion plus importante que la stéréoscopie classique : voir la vidéo d'exemple fournie.

2 Objectif des écrans 3D

Un écran 3D a pour but principal d'augmenter l'immersion de l'observateur dans la scène projetée sur l'écran. Dans la vie courante, lorsque nous observons une scène réelle composée de plusieurs objets, chaque oeil voit une image différente de la scène : c'est le principe de la stéréoscopie. Ceci permet au cerveau humain d'interpréter les informations reçues afin d'évaluer les formes géométriques des objets environnants et leur distance. Lorsque nous nous déplaçons autour de ces objets, une infinité de points de vue différents de cette scène est transmise au cerveau et cet ensemble dense d'informations permet au cerveau de reconstruire le modèle 3D de la scène (figure 1.a). Il en va de même lorsque nous observons cette scène retransmise sur un écran 3D. La stéréoscopie nous aide à évaluer la distance des objets, et le fait de pouvoir tourner autour des objets (autour de l'écran) permet de mieux comprendre la disposition des éléments dans la scène. Evidemment, le caractère discret de l'écran interdit l'affichage d'une infinité de vues simultanées mais le choix d'un nombre de vues minimal est essentiel pour conserver une vision stéréoscopique. L'angle θ_v sous lequel un oeil voit une vue inchangée doit être inférieur ou égal à l'angle θ_y formé par les deux yeux de l'observateur afin que les deux yeux voient toujours une vue différente de la scène projetée sur l'écran (figure 1.b). Pour un observateur placé à 3m de l'écran, $\theta_y = 1.2^o$ (la distance interoculaire est généralement de 6.25cm).

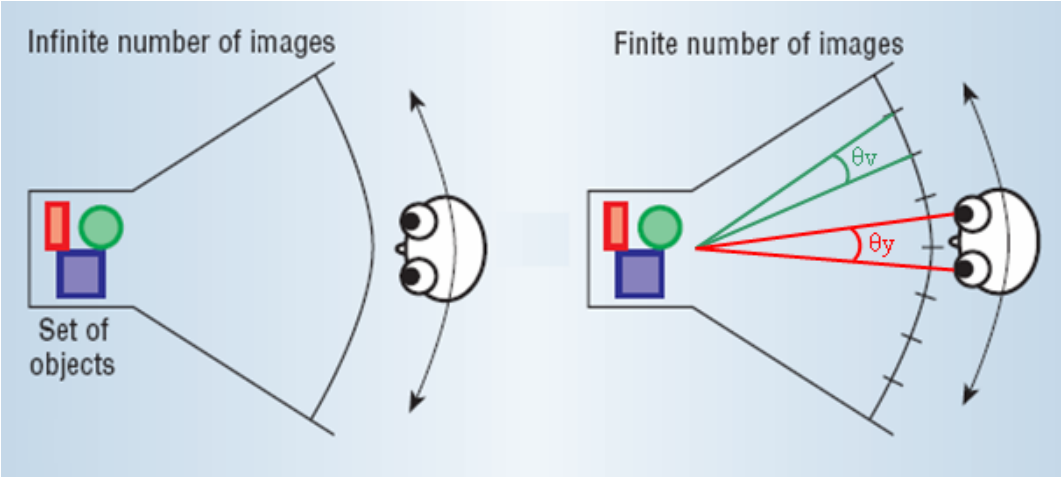

FIG. 1 – a. A gauche, un observateur se déplaçant autour d'une scène voit un nombre infini de points de vue ; b. A droite, un observateur se déplaçant autour d'un écran projetant de multiples vues de cette même scène percevra la scène en relief si et seulement si $\theta_v \leq \theta_y$.

Pour afficher une scène réelle sur ce type d'écran, il faut au préalable avoir enregistré la scène réelle à partir d'un ensemble de caméras placées au centre de chaque zone de vue réaffichée (figure 2). Il est donc nécessaire de capturer autant de vues que le nombre de vues affichables par le dispositif final de visualisation.

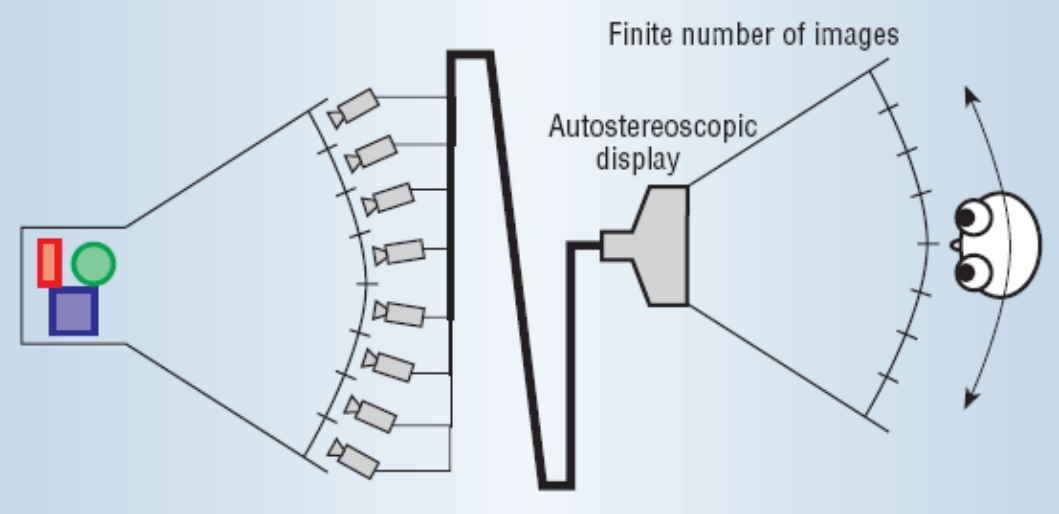

FIG. 2 – Schéma complet illustrant l'enregistrement d'une scène réelle par de multiples caméras et le réaffichage de cette scène sur un écran multi-vues autostéréoscopique.

Nous verrons dans la partie II (section 2) que le nombre de vues en entrée (caméras) n'est pas toujours égal au nombre de vue en sortie (écran 3D). Dans ce cas, des algorithmes de reprojection de vues sont utilisés pour reconstruire de nouveaux points de vue virtuels (à partir de points de vue réels) correspondant aux différents points de vue devant être visualisés par l'utilisateur situé en face de l'écran 3D.

3 Technologie des écrans 3D

Il existe deux grands types de technologie pour construire un écran autostéréoscopique : soit par application d'un panneau lenticulaire, soit par application d'une barrière de parallaxe. Dans les deux cas, la technologie appliquée nécessite de positionner des matériaux devant un écran LCD avec une précision nettement inférieure au pixel (de l'ordre de la dizaine de microns).

3.1 Avec panneau lenticulaire

L'application d'un panneau lenticulaire devant un écran LCD permet de dévier les rayons issus des pixels de l'écran afin de reproduire un affichage stéréoscopique. Le panneau lenticulaire est composé d'une grille régulière de petites lentilles sphériques. Chaque lentille recouvre une zone de plusieurs pixels de l'écran LCD et va dévier la direction de la lumière émise par les pixels couverts.

Chaque lentille peut être vue par l'utilisateur comme un pixel 3D, où la lumière émise par une lentille dans une direction donnée provient toujours d'un seul pixel, mais elle provient d'un pixel différent selon la direction d'observation choisie. Par conséquent, un observateur regardant une lentille de l'écran 3D ne voit pas le même pixel de l'écran LCD entre son oeil gauche et son oeil droit, bien que ses deux yeux fixent la même lentille. Sur la figure 3, l'oeil gauche voit une lentille émettant une couleur rouge (provenant d'un pixel rouge) tandis que l'oeil droit voit simultanément cette même lentille de couleur bleue.

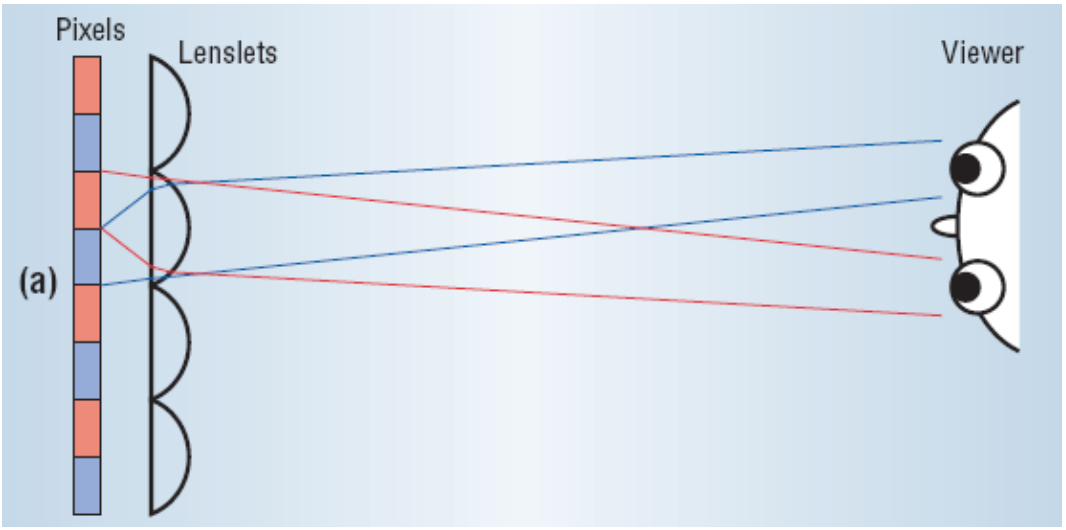

FIG. 3 – Construction d'un écran autostéréoscopique : un réseau de lentilles est appliqué devant un écran LCD afin de dévier les rayons lumineux émis par l'écran en direction de l'observateur.

L'observateur voit la résolution de l'écran 3D diminuée par rapport à l'écran LCD, elle correspond à la résolution du panneau lenticulaire (nombre de lentilles sphériques) puisque chaque lentille va émettre une seule couleur provenant d'un pixel de l'écran LCD. Comme les deux yeux de l'observateur sont supposés être sur un plan horizontal (l'axe horizontal de l'écran 3D), la résolution de cet écran peut être étendue en ne déviant que les rayons selon l'axe horizontal. Pour se faire, les constructeurs utilisent généralement des lentilles cylindriques (et non sphériques). Ainsi la résolution selon l'axe vertical de l'écran LCD est

conservée pour l'écran 3D (une lentille cylindrique émet la lumière provenant d'une ligne verticale de pixels de l'écran LCD).

3.2 Avec barrière de parallaxe

L'application d'une barrière de parallaxe devant un écran LCD permet de bloquer certains rayons issus des pixels de l'écran afin de reproduire un affichage stéréoscopique. La barrière de parallaxe est composée d'un filtre noir opaque et de petits trous en tête d'épingle laissant passer certains rayons lumineux issus des pixels de l'écran LCD.

En comparaison avec le réseau lenticulaire, nous pouvons voir chaque lentille sphérique remplacée par une surface opaque et un petit trou en son centre. Par conséquent, comme dans le cas du réseau lenticulaire, un observateur regardant un petit trou de l'écran 3D ne voit pas le même pixel de l'écran LCD entre son oeil gauche et son oeil droit : ceci est dû au fait que la barrière de parallaxe n'est pas accollée à l'écran LCD, une distance de quelques millimètres sépare les deux panneaux (des détails plus techniques sont donnés dans la partie II). Sur la figure 4, l'oeil gauche voit au travers d'un trou une couleur rouge (provenant d'un pixel rouge) tandis que l'oeil droit voit simultanément une couleur bleue au travers de ce même trou.

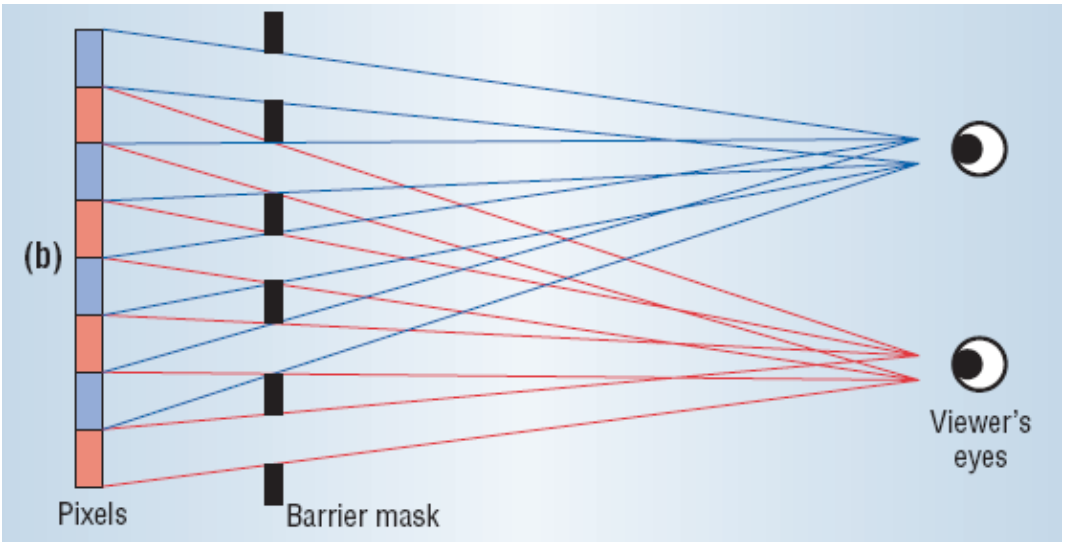

FIG. 4 – Construction d'un écran autostéréoscopique : une barrière de parallaxe est appliquée devant un écran LCD afin de stopper certains rayons lumineux émis par l'écran en direction de l'observateur.

La résolution de l'écran 3D correspond à la résolution de la barrière de parallaxe (nombre de trous). Comme dans le cas de l'application d'un réseau lenticulaire, la résolution de cet

écran peut être étendue en stoppant des rayons seulement selon l'axe horizontal. Pour se faire, les constructeurs remplacent les trous en tête d'épingle par des fentes verticales. Ainsi la résolution selon l'axe vertical de l'écran LCD est conservée pour l'écran 3D (une fente verticale émet la lumière provenant d'une ligne verticale de pixels de l'écran LCD).

Ces deux technologies (panneau lenticulaire / barrière de parallaxe) sont différentes mais produisent le même effet 3D. L'impression de relief est identique puisque les pixels vus par chaque oeil sont indépendants de la technologie employée. La connaissance des caractéristiques exactes de l'écran 3D permettent de produire des images en relief en projetant les données issues des caméras sur l'écran 3D : voir la partie II pour des détails techniques.

Le choix du réseau lenticulaire est particulièrement intéressant pour pouvoir continuer à utiliser l'écran comme un écran 2D classique : en effet, les lentilles sont noyées dans un substrat qui, par polarisation électrique, peut prendre l'indice de réfraction des lentilles et ainsi désactiver l'effet du panneau lenticulaire (plus aucune déviation des rayons lumineux). En revanche, cette technologie est moins utilisée lorsque l'écran doit être regardé sur des angles de vision très larges : les lentilles déforment les directions des rayons pour des angles rasants.

La barrière de parallaxe peut sembler plus judicieuse puisqu'elle n'admet pas de déformation sur les angles de visualisation rasants. En contrepartie, elle absorbe une partie non négligeable de la lumière émise par l'écran LCD, ce qui implique d'utiliser un écran LCD de forte luminosité et d'avoir tout de même une visualisation 3D avec une plus faible luminosité (les images affichées paraissent toujours un peu sombres).

4 Intérêt des écrans multi-vues

Jusqu'à présent, nous avons essentiellement abordé l'effet d'autostéréoscopie permettant d'afficher des images en relief sans utiliser d'équipement spécifique. Mais généralement, ces écrans ne sont pas seulement stéréo (2 vues affichées simultanément) mais multi-vues (entre 4 et 9 vues sont affichées simultanément suivant les constructeurs). Ces écrans sont parfois désignés sous le nom d'écrans automultiscopiques.

4.1 Ecran autostéréoscopique

Dans le cas d'un écran 3D à deux vues, l'observateur doit être correctement placé devant l'écran afin de percevoir le relief affiché sur l'écran.

En effet, l'observateur doit être à une distance bien précise de l'écran 3D. Cette distance dépend des caractéristiques de l'écran 3D, i.e. résolution, taille réelle de l'écran LCD et focale entre l'écran LCD et le dispositif placé devant l'écran pour dévier / bloquer certains rayons lumineux / pixels. Sur la figure 5, les zones de vues permettant la visualisation du relief sont affichées en couleur. Elles sont en forme de losange et inter-connectées en un point. Pour visualiser correctement le relief, l'observateur doit avoir son oeil gauche placé dans une

zone de vue rouge et l'oeil droit placé dans une zone de vue bleue. Nous remarquons que plus l'observateur s'éloigne de la distance de vue optimale (représentée par un trait noir vertical), que ce soit en s'approchant ou en s'éloignant de l'écran 3D, plus les zones de vues s'affinent (bord des losanges) et plus l'utilisateur est inconfortable devant l'écran : il doit constamment chercher à se repositionner afin de visualiser le relief... le moindre mouvement latéral va positionner l'observateur à l'extérieur d'une zone de vue et l'effet 3D n'est plus perçu, l'image affichée devient floue et désagréable à regarder.

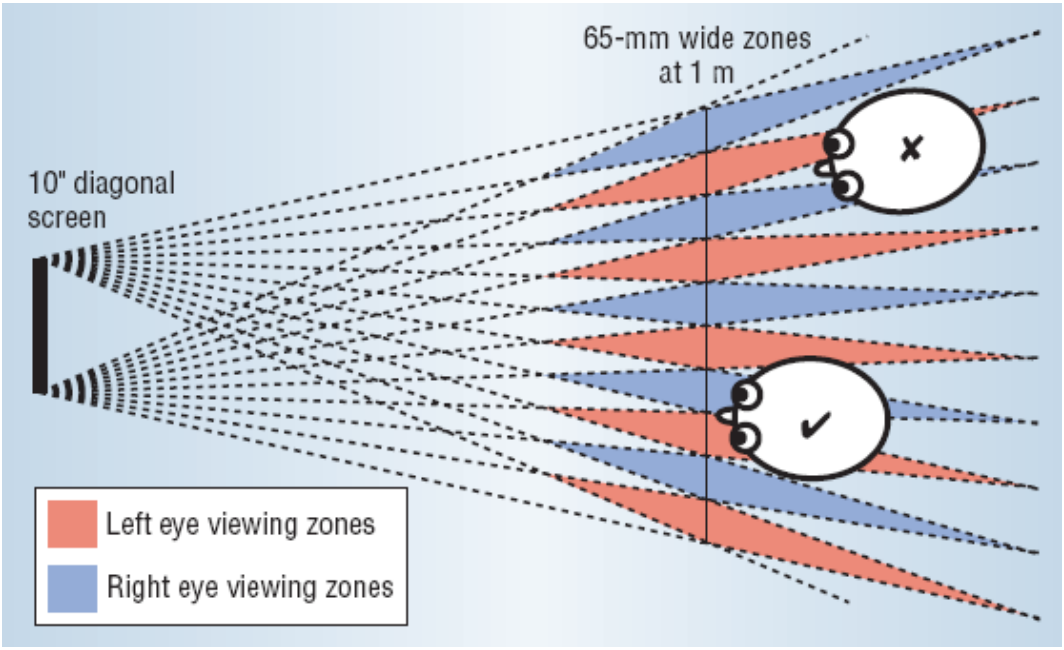

FIG. 5 – Affichage d'une image en relief sur un écran autostéréoscopique à deux vues. L'observateur est contraint de se placer à une certaine distance de l'écran 3D et à une certaine position devant l'écran pour une visualisation correcte en relief.

Lorsque l'utilisateur est placé près de la distance de vue optimale (ligne noire sur la figure 6), la zone de vue est plus large et par conséquent l'observateur est moins contraint à rester statique devant l'écran 3D. Néanmoins, il ne peut se déplacer latéralement sans voir l'effet stéréo s'inverser et ainsi perdre la visualisation en relief. Pour pallier ce problème, la solution serait d'utiliser un détecteur de mouvement positionné sur la tête afin d'afficher les deux vues en fonction de la position de l'observateur face à l'écran. Ce système est gênant puisqu'il force l'observateur à s'équiper d'un dispositif alors que justement, l'intérêt d'utiliser un écran autostéréoscopique est de se dispenser de tout matériel complémentaire afin d'augmenter l'immersion de l'observateur. De plus, il est très difficile de gérer les transitions entre les zones de vues : un saut visuel va apparaître lors du passage d'un affichage en stéréo à un

affichage en stéréo inversée. L'observateur ne va généralement pas se déplacer exactement sur la ligne de vue optimale : ses yeux se retrouveront quelques courts instants à l'extérieur des zones de vue car ces zones ne sont raccrochées entre-elles que par un point situé sur la ligne de vue optimale (figure 6).

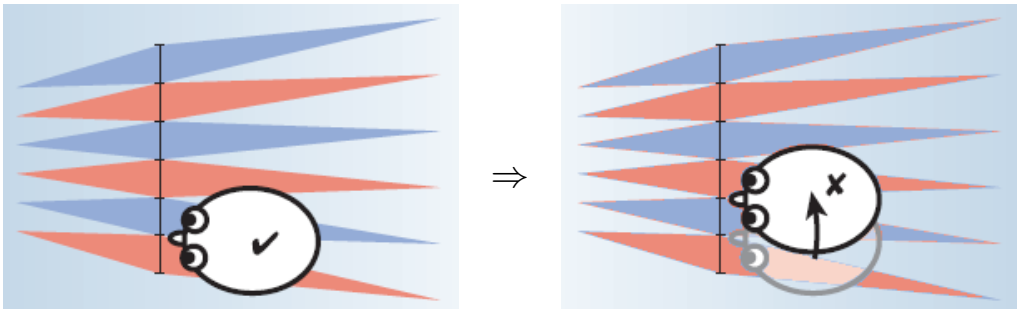

FIG. 6 – L'observateur ne peut se déplacer latéralement devant un écran 3D à deux vues, sinon il perçoit le relief en mode stéréo inversée.

4.2 Ecran automultiscopique

Dans le cas de l'utilisation d'un écran 3D multi-vues, la zone de visualisation augmente en fonction du nombre de vues affichables par l'écran 3D : pour un écran à n vues, un observateur placé à la distance de vue optimale est correctement positionné *n-1 fois sur n* : l'augmentation du nombre de vues augmente considérablement le confort de positionnement de l'observateur devant l'écran. En effet, dans l'exemple de l'utilisation d'un écran à 16 vues sur la figure 7, il suffit que les deux yeux de l'utilisateur se situent dans la zone de vue colorée pour visualiser correctement le relief. Par conséquent, l'observateur peut se déplacer latéralement devant l'écran sans perdre la vision en relief : l'angle de vision proposé par les écrans multi-vues actuels est autour de 10^o. L'observateur est également libre de s'éloigner de la ligne de vue optimale tout en conservant une visualisation correcte du relief à l'écran. Nous verrons dans la partie technique (partie II_, section 3) pourquoi de tels déplacements sont autorisés.

Les systèmes de visualisation multi-vues permettent ainsi à plusieurs utilisateurs de se positionner simultanément devant un tel écran pour visualiser correctement le relief affiché, en se positionnant les uns devant les autres dans une même zone de vue mais également les uns à côté des autres dans différentes zones de vue, et ceci sans matériel additionnel. La figure 8 montre l'exemple de personnes situées dans trois zones de vue de l'écran. Toutes les zones de vue (angle d'environ 10^o par zone de vue) affichent le même relief, il y a donc environ 18 zones de vues possibles autour de l'écran (pour un écran visualisable sur 180^o).

La plus forte restriction de déplacement, pour un observateur, c'est l'angle de vue restituable par l'écran (10^o sur les écrans multi-vues actuels). Pour augmenter cet angle de

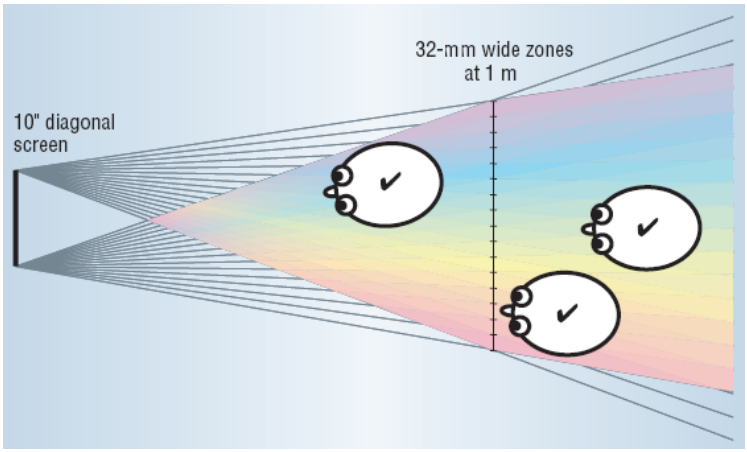

FIG. 7 – Exemple d'un écran à 16 vues où une seule zone de vue est représentée (zone colorée). L'observateur peut se déplacer latéralement devant l'écran 3D : il suffit que ses deux yeux restent dans la zone de vue pour percevoir correctement le relief. Ceci autorise donc plusieurs observateurs à se placer simultanément devant l'écran pour une visualisation en relief.

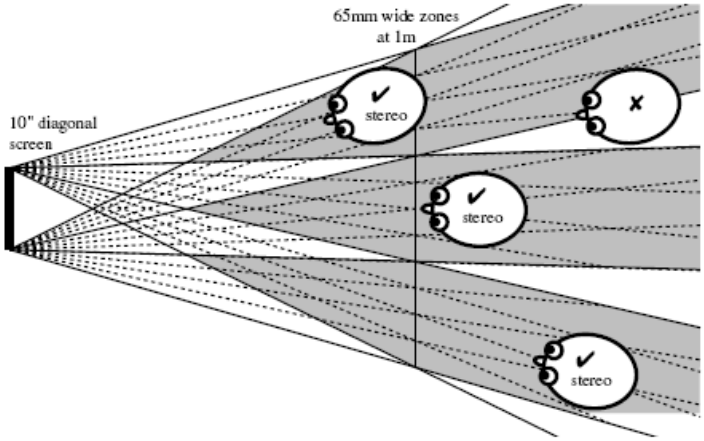

FIG. 8 – Exemple d'un écran à 4 vues où trois zones de vue (grises) sont représentées. Plusieurs observateurs peuvent donc visualiser des images en relief simultanément en se répartissant dans les différentes zones de vue.

vue, il est possible d'équiper l'utilisateur d'un capteur de mouvement positionné sur la tête afin d'afficher des images en relief correspondant à la position de l'observateur. Cette technique à l'avantage de permettre à l'utilisateur de se déplacer sur 180^o autour de l'écran et d'avoir la sensation de tourner autour d'un objet visualisé en relief sur l'écran. Néanmoins, même si cette approche permet d'afficher des images différentes pour les deux yeux (images stéréo) en limitant les artefacts visuels lors des déplacements (la zone de vue reste centrée dans la direction de l'observateur), elle contraint l'écran à une mono-utilisation : le relief est correctement perçu par un seul observateur.

Deuxième partie
Présentation technique

1 Les systèmes de visualisation 3D

De nombreux prototypes d'écran 3D ont été réalisés durant ces dix dernières années avec pour objectifs principaux :

- Un large écran de visualisation ;
- Un large angle de rotation autour de l'écran ;
- Une forte qualité d'image restituée.

Ces objectifs sont difficilement atteignables simultanément et des compromis doivent être faits sur certains critères.

1.1 Ensemble de vidéo-projecteurs

Dans [?], un ensemble de 16 vidéo-projecteurs est utilisé pour afficher simultanément 16 images sur un écran de projection : cette technique permet de s'abstraire de la résolution de l'écran pour diffuser des contenus multimédias en relief.

Deux procédés différents existent : soit les images sont projetées sur un écran diffuseur et sont observées de l'autre côté de l'écran (figure 9), soit les images sont projetées sur un écran miroir et sont observées dans la direction miroir des projecteurs par rapport à l'écran (figure 10).

Comme toutes les images sont projetées sur le même écran, il faut placer un panneau lenticulaire entre les projecteurs et l'écran afin de séparer les rayons provenant des différentes images. Il en va de même du côté de l'observateur : un second panneau lenticulaire doit être placé devant l'écran afin que chaque oeil de l'observateur puisse voir une image provenant d'un projecteur différent et ainsi obtenir une vision stéréoscopique (une image différente pour chaque oeil).

Chaque projecteur peut être relié à une caméra différente afin de fournir un système de TV 3D temps-réel. Pour cela, chaque caméra doit être correctement positionnée dans

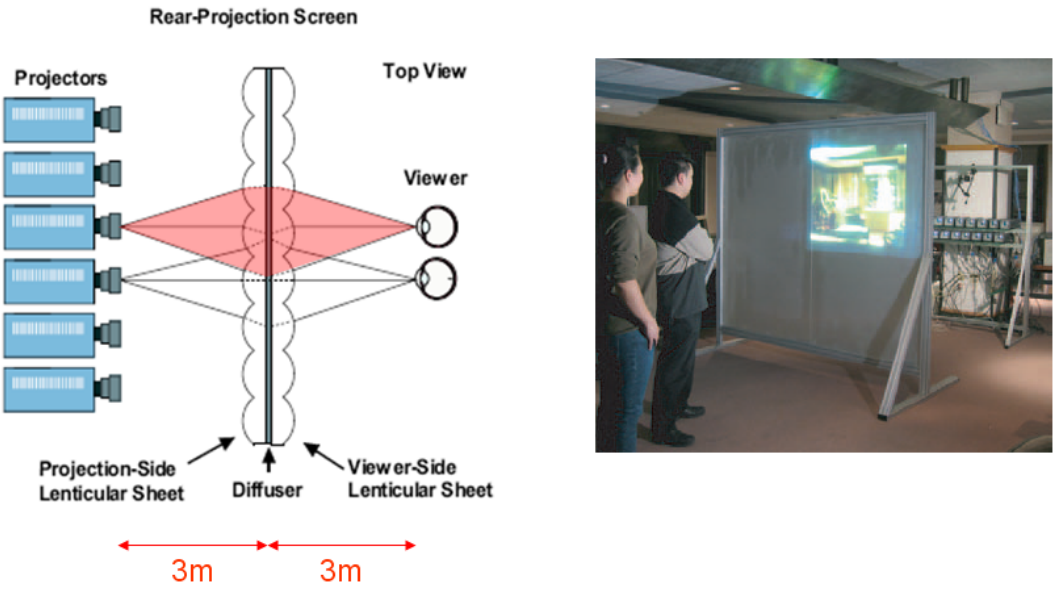

FIG. 9 – Système de retransmission TV 3D composé de 16 vidéo-projecteurs et d'un écran de projection diffuseur équipé d'un double panneau lenticulaire (de chaque côté de l'écran).

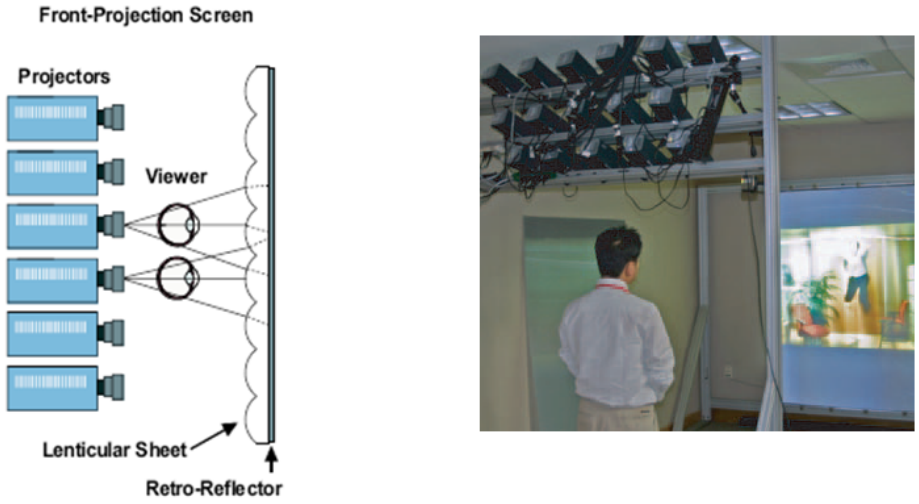

FIG. 10 – Système de retransmission TV 3D composé de 16 vidéo-projecteurs et d'un écran de projection miroir équipé d'un simple panneau lenticulaire.

la scène à visualiser : nous verrons les différentes techniques de positionnement de caméras dans la sous-section 1.4.

1.2 Lentilles convergentes

Un panneau lenticulaire est constitué d'une grille régulière de lentilles sphériques, convergentes. Tout rayon, issu d'un projecteur, traverse le centre d'une lentille sphérique sans aucune déviation (principe de la lentille sphérique, centrée) pour venir toucher l'écran en un point : figure 11. De plus, tous les rayons issus d'un même projecteur et traversant une même lentille sont supposés parallèles car le projecteur est distant de l'écran de 3 mètres et la focale des lentilles est de l'ordre du millimètre. Par conséquent, tous ces rayons vont converger en un même point sur le plan focal (écran), c'est le principe même des lentilles convergentes : voir figure 11.

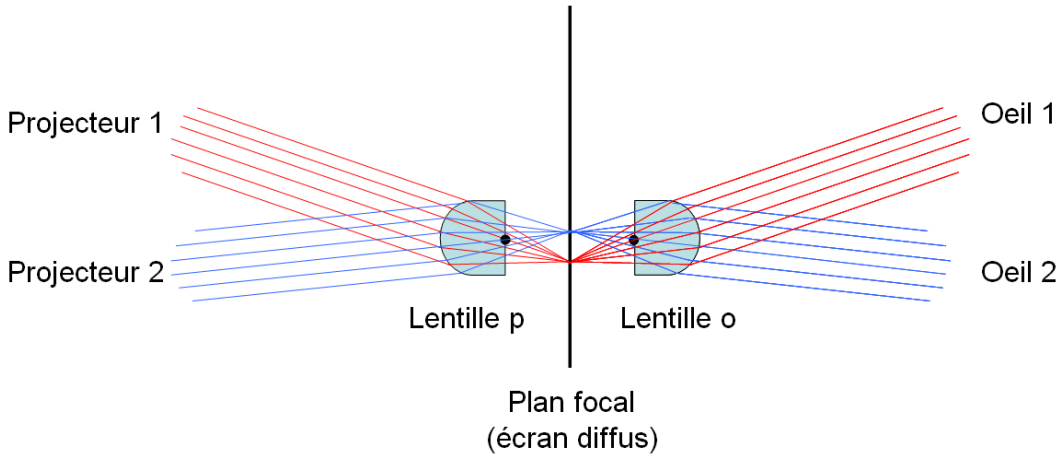

FIG. 11 – Tous les rayons issus d'un projecteur et traversant une même lentille convergente, vont converger en un point sur le plan focal. De plus, par réciprocité, tous les rayons traversant l'écran diffuseur en un point, vont être parallèles après la traversée d'une nouvelle lentille convergente. Par conséquent, l'observateur voit des informations en stéréo à travers la lentille o.

Pour que le système de visualisation du relief reste cohérent, il ne faut ni observer l'écran sur un angle de plus de 30^o, ni placer des projecteurs sur un angle de plus de 30^o : voir figure 12. Ces restrictions sur les conditions d'observation sont déduites des propriétés physiques des lentilles sphériques généralement utilisées (diamètre de $1mm$, focale de $1.86mm$). Le champ de vue horizontal fov est calculé par la relation :

$$fov = 2 \times \arctan\left(\frac{1}{2 \times 1.86}\right) = 30^o$$

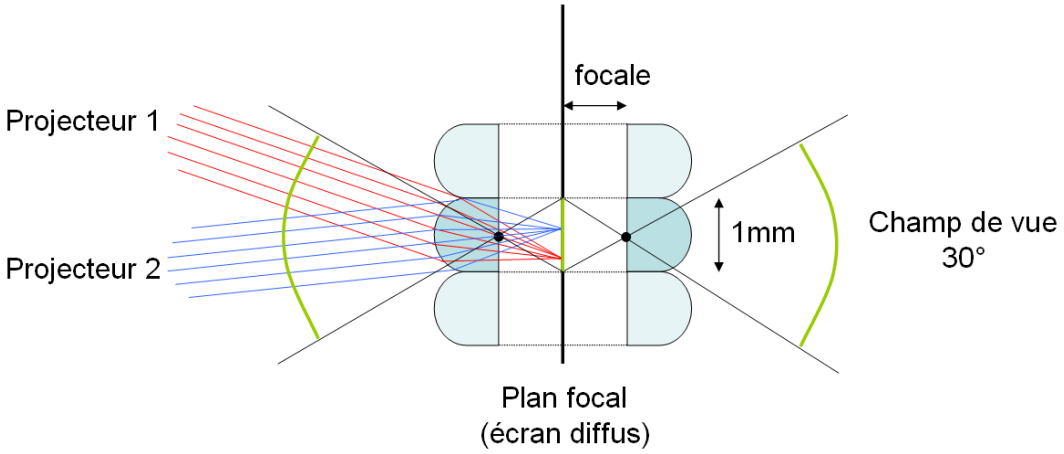

FIG. 12 – L'angle de vue ne peut excéder 30^o, sinon l'observateur peut apercevoir à travers une lentille des rayons provenant de l'écran qui étaient destinés à être visualisés par une autre lentille, ce qui provoquerait des ruptures dans l'affichage du relief et ainsi une dégradation de la vision stéréoscopique.

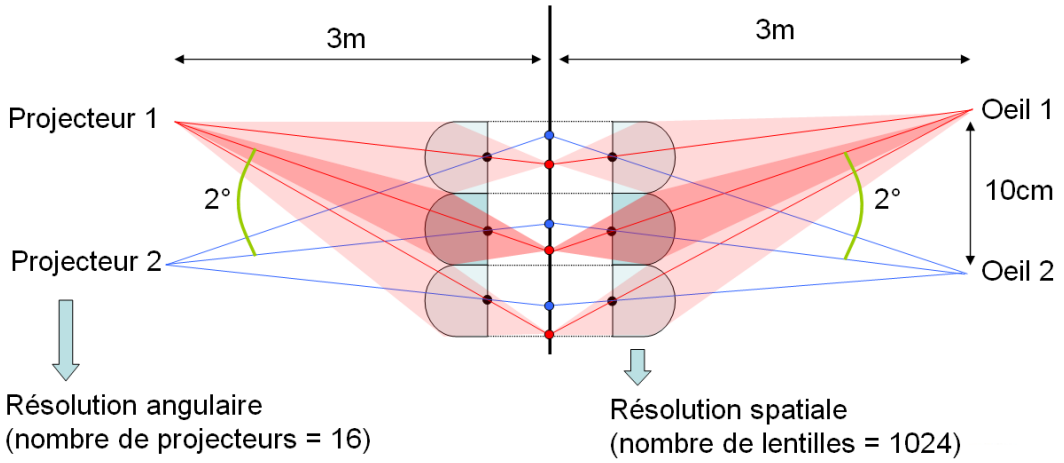

FIG. 13 – Pour une vision pleinement stéréoscopique, l'angle entre les directions d'observation des deux yeux ne doit pas excéder 2^o, ce qui contraint le système de visualisation à utiliser un minimum de 16 projecteurs pour balayer le champ de vue complet : c'est la résolution angulaire de l'écran 3D. Nous pouvons également remarquer qu'une image issue d'un projecteur est entièrement retransmise vers l'oeil avec une résolution spatiale dépendant du nombre de lentilles sur le panneau lenticulaire.

De plus, pour que la vision reste bien en stéréoscopie, il faut s'assurer que les deux yeux voient des images provenant de projecteurs différents. L'écartement interoculaire étant généralement proche des $7cm$, une distance d'observation de $2m$ induit une déviation de 2^o sur la direction d'observation. Par conséquent, pour pouvoir balayer l'intégralité du champ de vue (30^o), au minimum 16 projecteurs sont nécessaires : c'est ce qui caractérise la résolution angulaire de l'écran 3D, voir figure 13. De plus, nous pouvons remarquer que l'image issue du projecteur 1 est intégralement retransmise à l'oeil 1 sans aucune perte de luminosité ; en revanche, la résolution de l'image perçue dépend du nombre de lentilles : cela correspond à la résolution spatiale de l'écran 3D. Nous verrons l'importance de ces résolutions (spatiale et angulaire) dans la section 2 traitant des problèmes d'échantillonnage.

Maintenant, étudions le cas plus général de l'observation de l'écran 3D par un observateur placé au hasard devant l'écran 3D (mais positionné à la bonne distance de visualisation) : voir figure 14.

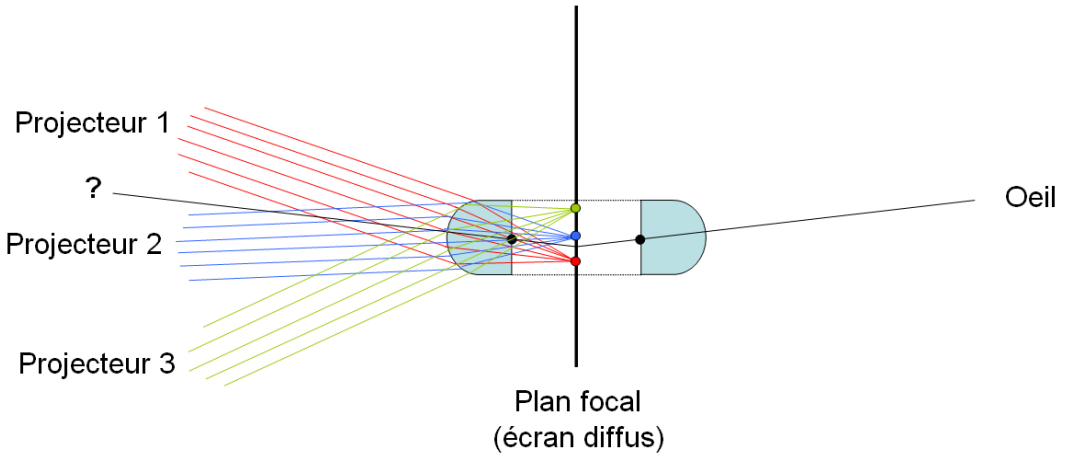

FIG. 14 – Pour un utilisateur placé de manière quelconque devant l'écran 3D, quelle image voit-il ? A priori, il peut se situer dans une zone où aucune image n'est visible.

Il serait préférable que cet utilisateur voit une image provenant d'un projecteur sans qu'il ait à se déplacer pour trouver une zone de vue confortable. Or actuellement, il voit théoriquement une image qui serait issue d'un projecteur placé devant le point d'interrogation de la figure. Comme nous ne pouvons pas placer un nombre infini de projecteurs pour satisfaire l'infinité de positions d'observation devant l'écran, nous allons déplacer l'écran de projection par rapport au plan focal afin que l'ensemble des images projectées couvrent la superficie complète de l'écran de projection : voir figure 15.

Ce principe va permettre à l'utilisateur de se déplacer latéralement devant l'écran tout en conservant un affichage en relief. Et, pour éviter une perte de luminosité lors de la réception de l'image par l'oeil, nous déplaçons également l'écran de projection par rapport au plan

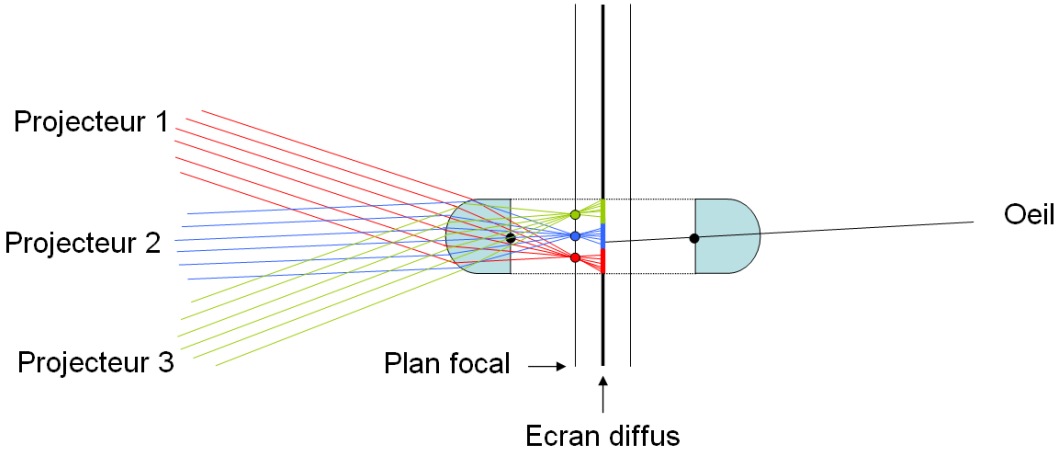

FIG. 15 – Séparation du plan focal et de l'écran de projection afin que l'ensemble des projecteurs recouvrent la superficie totale de l'écran. Séparation identique du côté observateur afin de conserver la luminosité totale de l'écran.

focal du second réseau lenticulaire, placé entre l'observateur et l'écran. Il va tout de même y avoir quelques imprécisions sur la reprojection des images sur l'écran, notamment pour les lentilles situées sur la périphérie de l'écran : l'écran est de grande dimension, il entraîne donc une déviation des rayons traversant les lentilles loin de la zone centrale de l'écran et un léger chevauchement des images va se produire dans ces régions, entraînant du flou lors de la visualisation : voir figure 16.

1.3 Ecran LCD de forte résolution

Pour remédier à ces problèmes de chevauchement entre rayons issus de projecteurs voisins, l'idée est de remplacer le système projecteurs + écran de projection par un écran à cristaux liquides. Evidemment, pour conserver une bonne qualité d'image visualisée, cet écran LCD doit être de forte résolution : il doit englober à la fois la résolution spatiale et la résolution angulaire de l'écran 3D : voir figure 17.

Sur cet exemple, 8 pixels sont placés derrière chaque lentille / chaque trou pour pouvoir stocker des informations provenant de 8 images différentes. Malgré une plus faible résolution angulaire (8), l'écran LCD nécessite une résolution impressionnante (5120 pixels de large) pour ne conserver au final qu'une résolution spatiale assez rudimentaire (640 pixels horizontaux visibles par chaque oeil). Il est donc difficile d'utiliser un écran LCD pour pouvoir effectuer un affichage multi-vues avec une bonne résolution spatiale qui correspond à la résolution réellement perçue par l'oeil. Nous verrons dans la partie III comment Newsight contourne cette nécessité de très forte résolution de l'écran LCD sans compromettre la qualité visuelle du relief 3D.

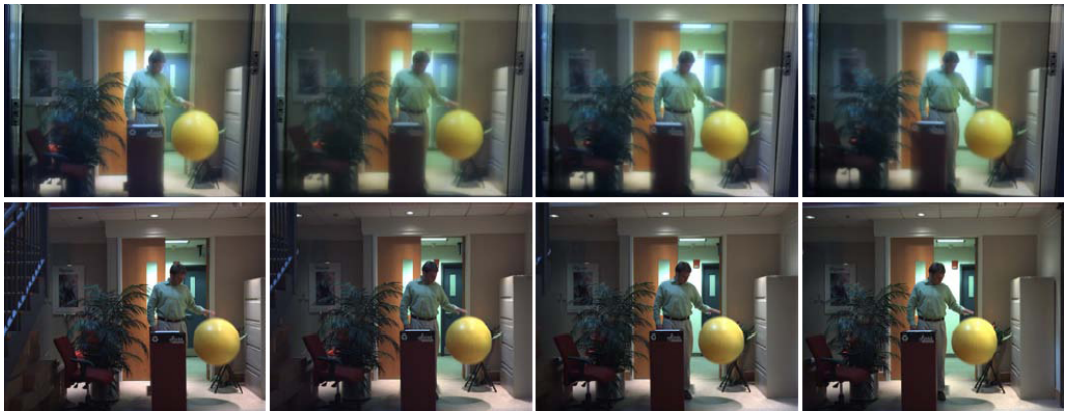

FIG. 16 – En bas, images issues des projecteurs ; en haut, images issues de l'écran de projection. Nous pouvons remarquer que les images observées sur l'écran sont plus floues et que quelques problèmes de chevauchement apparaissent (en haut de la sphère jaune, par exemple).

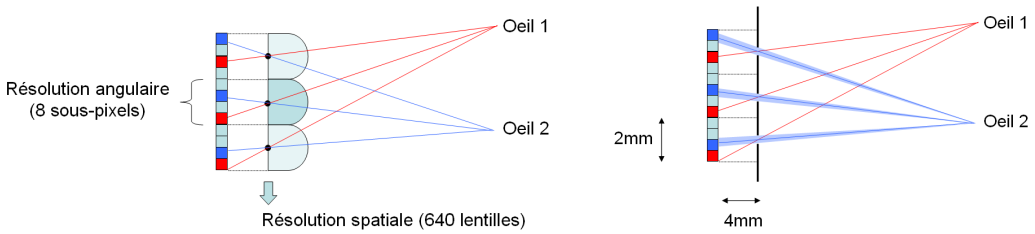

FIG. 17 – L'écran LCD doit être de très forte résolution pour pouvoir construire un écran autostéréoscopique multi-vues. Dans cet exemple, la résolution horizontale est de 5120 pixels.

1.4 Configuration de caméras

Lors de l'enregistrement de scènes réelles pour construire des images multi-vues, il faut positionner correctement les caméras les unes par rapport aux autres afin de pouvoir reproduire un affichage stéréoscopique sur un écran multi-vues. L'idée, la plus intuitive et la plus facile à mettre en place dans le cadre de la capture multi-caméras de scènes réelles, est d'enregistrer des séquences d'images simultanément depuis plusieurs caméras et d'utiliser directement les images issues des différentes caméras comme l'ensemble des vues en entrée de l'écran 3D : ceci permet l'affichage d'une vidéo en relief sans aucun traitement sur les images, sans aucune projection sur un plan fixe représentant l'écran 3D dans le repère de la scène à visualiser. Ce principe est utilisé dans les systèmes d'affichage avec vidéo-projecteurs, chaque caméra étant reliée à un vidéo-projecteur différent.

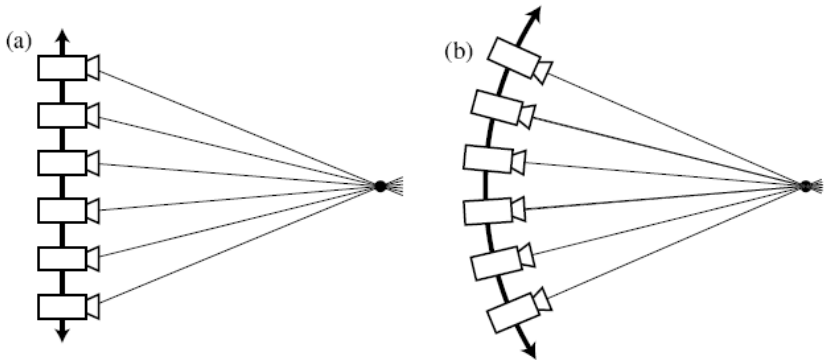

FIG. 18 – a. Configuration parallèle ; b. Configuration radiale.

Il existe plusieurs configurations concernant le placement des caméras devant la scène à enregistrer, soit une configuration parallèle, soit une configuration radiale : figure 18. Pour plus de détails, consulter l'article [**?**]. La configuration radiale est très facile à mettre en place pour l'enregistrement d'objets statiques (en faisant tourner l'objet devant une seule caméra par exemple), toutes les caméras sont orientées vers un point de convergence commun (le centre de l'objet) mais cette configuration ne permet pas d'utiliser directement les images enregistrées pour être visualisées sur un écran 3D. En effet, le plan de projection est différent d'une caméra à une autre, ce qui entraîne des distorsions très désagréables dans l'affichage de la stéréoscopie. Dans l'exemple de la figure 19, il n'y a pas conservation de la hauteur du cube entre les vues visualisées par les deux yeux de l'observateur : cet effet est très désagréable et nuit à la qualité de la perception du relief.

En revanche, la configuration parallèle est mieux adaptée pour un affichage direct sur écran 3D car le plan de projection des caméras est identique : figure 20. En effet, toutes les caméras doivent être positionnées sur une même ligne, avec un écartement régulier entre cha-

cune d'elles et des directions d'enregistrement communes pour pouvoir effectuer un affichage stéréoscopique correct.

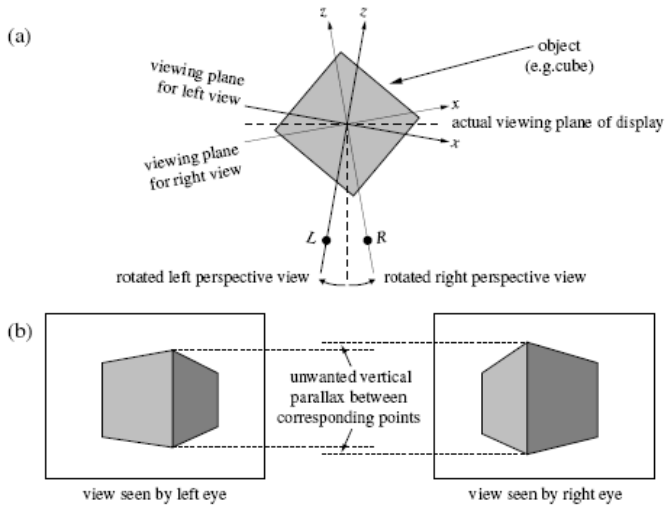

FIG. 19 – Configuration radiale : distorsion sur la hauteur des éléments perçus par les deux yeux.

En fixant une direction d'enregistrement commune pour l'ensemble des caméras, les caméras situées le plus à l'extérieur ne peuvent enregistrer la scène à visualiser que sur une partie de leur capteur CDD : partie gauche de la figure 21.

Pour pouvoir utiliser toute la surface d'enregistrement du capteur, il faut déplacer la lentille parallèlement au capteur (sans changer la direction de l'axe optique de la lentille) afin que l'axe de la caméra passe par le point de convergence (centre de la scène) : partie droite de la figure 21. Une autre solution, plus simple à mettre en place et très souvent utilisée, est de tourner intégralement la caméra pour l'orienter dans la direction du point de convergence tout en conservant l'alignement vertical de l'ensemble des caméras. Cette solution nécessite de traiter chacune des images enregistrées en les reprojetant sur un même plan, parallèle à l'écran 3D (afin de simuler la configuration parallèle) mais elle ne traite pas le problème de la lentille mal orientée : en fait, il faudrait utiliser une *pinhole* caméra pour anéantir l'effet de la lentille (caméra avec une ouverture très petite, de la taille d'une tête d'épingle). Cette technique est très souvent utilisée pour enregistrer le champ de lumière émis par des objets réels : ces données sont ensuite enregistrées dans une structure de données dédiée appelée *lumigraphe / lightfield* [?, ?].

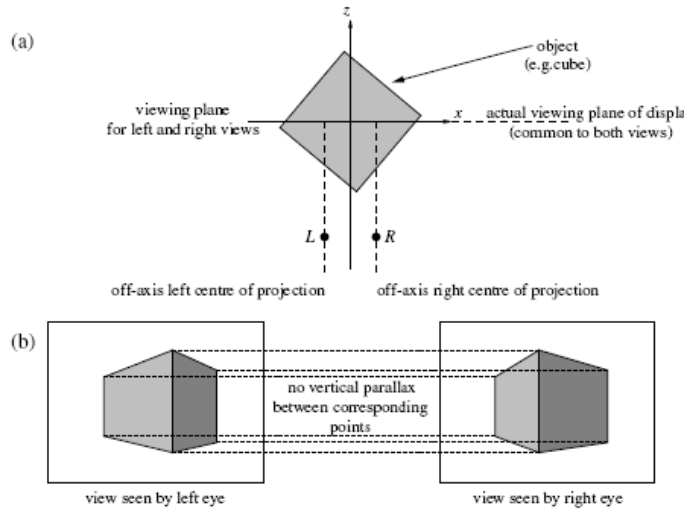

FIG. 20 – Configuration parallèle : aucune distorsion sur la hauteur des éléments perçus.

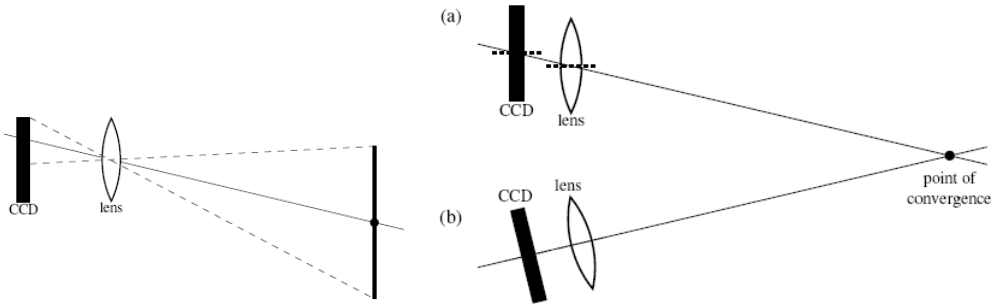

FIG. 21 – A gauche : Seulement une partie de la pellicule est utilisée lors de l'enregistrement par configuration parallèle. A droite : a. Pour éviter de n'utiliser qu'une partie de la pellicule dans la configuration parallèle, la lentille est déplacée orthogonalement à l'axe optique ; b. Configuration radiale avec une caméra toujours positionnée sur un même plan.

2 Les problèmes de rééchantillonnage

2.1 Analogie avec les lumigraphes

Lorsque nous ne disposons pas de caméras parfaitement calibrées pour construire des séquences visualisables sur un écran 3D, il est possible de travailler avec des séquences d'images multi-vues précédemment acquises. Même si ces séquences d'images ont été produites avec précision et rigueur (par exemple en construisant un lumigraphe [?, ?] où les prises de vue d'un objet sont calibrées et acquises régulièrement tout autour de lui), il n'est pas évident d'afficher ces images sur un écran 3D qui possède ses propres caractéristiques. En effet, même si un écran 3D avec barrière de parallaxe peut être vu comme un lumigraphe, il n'est pas évidement de faire le lien entre les images issues d'un lumigraphe pré-acquis et les images nécessaires à l'affichage du relief sur un écran 3D (figure 22). Sur la figure de gauche, les barrières de l'écran 3D sont vues comme des caméras et les sous-pixels en face de chaque barrière représentent la résolution de ces barrières/caméras. Pour plus de détails sur ce mode de représentation des rayons lumineux, consulter l'article [?].

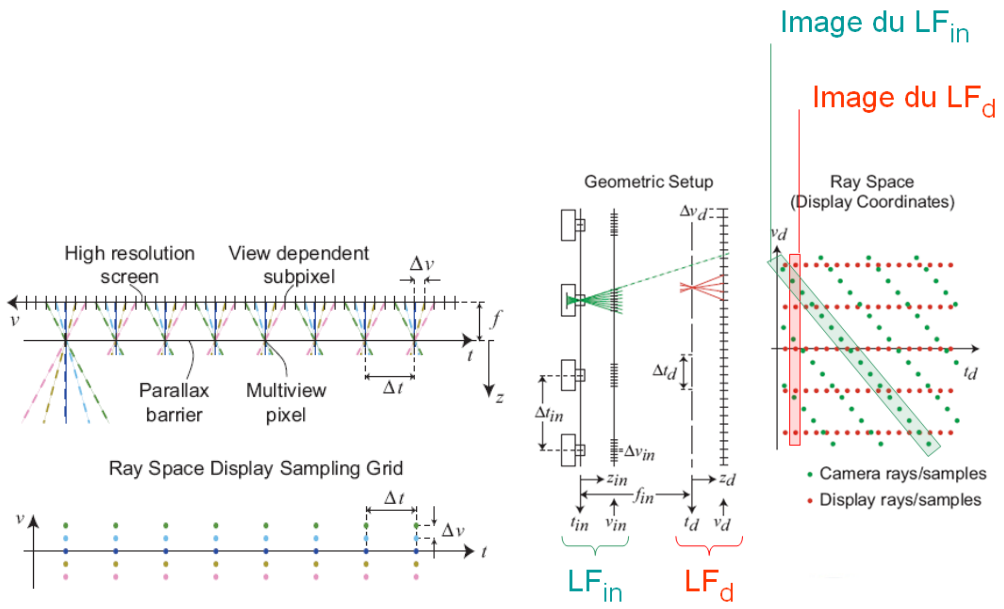

FIG. 22 – A gauche, analogie entre un écran 3D à barrière de parallaxe et un lumigraphe ; A droite : reprojection d'un lumigraphe pré-acquis LF_{in} pour construire le lumigraphe réellement affiché LF_d.

Sur la figure de droite, nous remarquons que l'acquisition d'un lumigraphe avec plusieurs caméras permet de construire un lumigraphe LF_{in} avec peu de points de vue (faible réso-

lution spatiale) mais des images de forte résolution (forte résolution angulaire) alors que le lumigraphe réellement affiché LF_d dispose de résolutions inversées (forte résolution spatiale correspondant au nombre de barrières de parallaxe, et faible résolution angulaire correspondant au nombre de sous-pixels en face de chaque barrière). Ceci implique une très faible concordance des rayons lumineux issus des deux lumigraphes : seuls les rayons/points en rouge et en vert qui se superposent sur la figure de droite sont en concordance ; un rééchantillonnage judicieux est donc nécessaire pour construire les autres rayons du lumigraphe à afficher.

2.2 Rééchantillonnage nécessaire

Un rééchantillonnage complet est nécessaire car une simple interpolation entre les rayons les plus proches ne permet pas d'obtenir une qualité d'image suffisante pour visualiser une scène en relief sur un écran 3D. En effet, une interpolation quadrilinéaire entre les rayons issus des plans UV et ST du lumigraphe LF_{in} suppose que les données à visualiser sont situées sur un plan commun : le plan de l'écran LCD dans la figure 23 gauche.

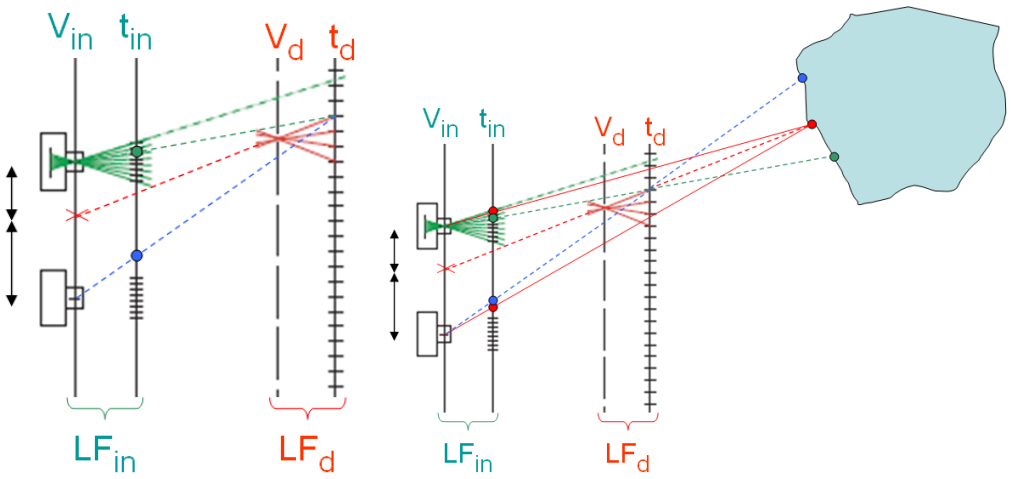

FIG. 23 – A gauche : le rayon en pointillés rouge est construit par interpolation des rayons en pointillés bleu et vert ; A droite, prise en compte de la profondeur de la scène : le rayon en pointillés rouge est construit par interpolation des deux rayons rouges.

Or, généralement, la profondeur de la scène n'est pas constante, il faut donc prendre en compte l'estimation de la profondeur de la scène pour reprojeter correctement les rayons et ainsi obtenir une interpolation de bien meilleure qualité (figure 23 droite). Evidemment, cette nouvelle interpolation dépendant de la profondeur, nécessite de connaître la géométrie de la scène à visualiser, ce qui est difficilement obtensible pour des scènes réelles ou animées.

Ce rééchantillonnage va construire de nouvelles images par interpolation entre plusieurs images déjà existantes. Des artefacts visuels peuvent donc apparaître, auquel cas un filtrage peut s'avérer utile et nécessaire afin l'adoucir les images produites. Dans certains cas (peu d'images dans le LF_{in}, mauvaise évaluation de la profondeur), des effets de flou ou de mosaïque (*ghosting*) peuvent apparaître et doivent être traités. Plus de détails sur les méthodes de rééchantillonnage de lumigraphes sont disponibles dans [?, ?, ?, ?].

Dans le cas de l'utilisation d'un nombre très insuffisant de vues issues du lumigraphe LF_{in}, la géométrie des objets observés peut être considérablement modifiée : dans l'exemple de la figure 24, le carré bleu est enregistré depuis une caméra placée en t_{in} et est revisualisé sur l'écran pour un point de vue situé en $s.t_{in}$. Pour générer cette nouvelle vue, seule l'image issue de t_{in} est utilisée (la plus proche), ce qui conduit à ne pouvoir visualiser qu'une seule arête du carré en position $s.t_{in}$: d'où l'impression de visualiser un trapèze plutôt qu'un carré.

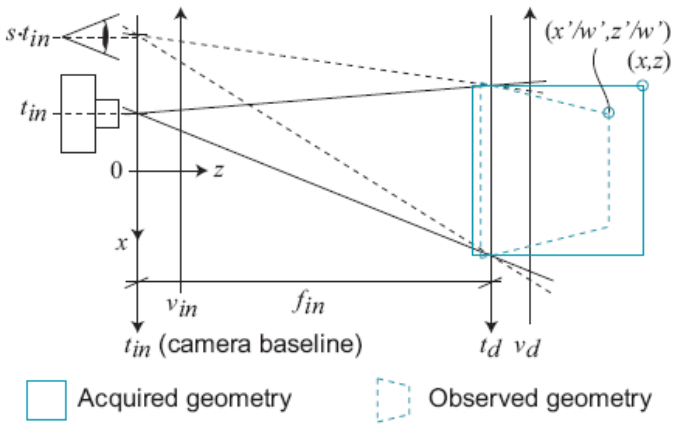

FIG. 24 – Déformation de la perspective lors de la visualisation à partir de nouveaux points de vue très différents des points de vue pré-enregistrés.

3 L'entrelacement des multi-vues

Nous avions vu dans la section 4.2 de la partie I que l'utilisation d'un écran multi-vues (supérieur à deux vues) permettait à l'utilisateur de se déplacer dans une large zone de visualisation. En effet, même s'il existe une distance de visualisation optimale pour les écrans 3D, l'observateur a tout de même la possibilité de se déplacer latéralement, et d'avancer ou reculer devant l'écran par rapport à la distance de visualisation optimale tout en conservant une forte qualité sur la visualisation en stéréoscopie.

La figure 25 montre un exemple de visualisation devant un écran 3D à 6 vues. La distance de vue optimale est située dans les losanges marqués par les chiffres 1, 2, ..., 6 représentant les zones de vues où un oeil verra une image provenant respectivement des caméras 1, 2, ..., 6. Ainsi si l'observateur place un oeil dans la zone 1 et l'autre dans la zone 2, il verra l'écran avec une visualisation stéréoscopique parfaite (une image différente pour chaque oeil). Néanmoins, si l'observateur se déplace et, par exemple, positionne un oeil dans la zone 12 et l'autre dans la zone 23, il observera toujours une visualisation en stéréoscopie car en position 12, l'oeil verra sur la partie gauche de l'écran une image provenant de la caméra 1 et l'autre oeil (position 23) verra sur la même partie gauche de l'écran une image provenant de la caméra 2 et sur la partie droite une image provenant de la caméra 3. Par conséquent, la stéréscopie s'applique sur chaque partie de l'écran, soit entre les images 1 et 2, soit entre les images 2 et 3 ; accessoirement, un petit artefact visuel peut être visible dans la zone de transition au centre de l'écran.

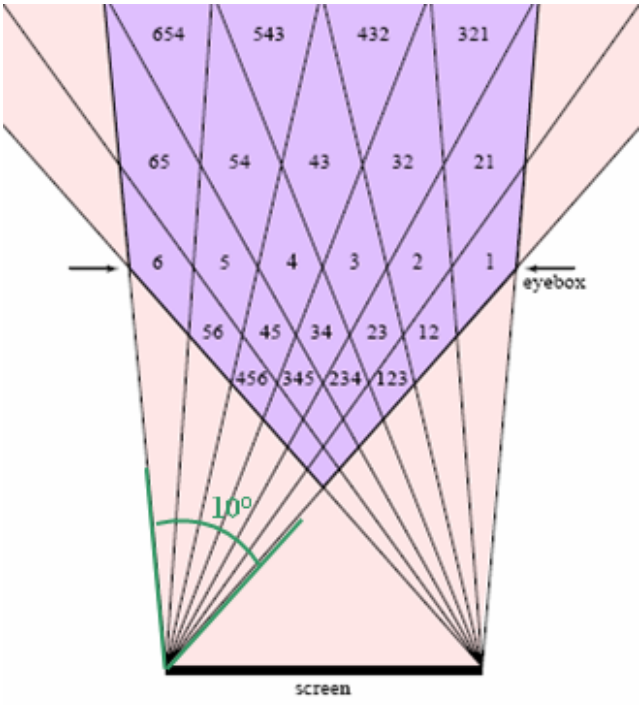

FIG. 25 – Zones de visualisation stéréoscopique sur un écran 3D à 6 vues : l'observateur peut se déplacer latéralement et en avant-plan dans la zone de vue violette sans perdre la vision stéréo.

Toute la zone de vue apparaissant en violet sur la figure représente des positions valides pour les deux yeux d'un observateur : celui-ci peut donc se déplacer latéralement devant

l'écran (sur un angle de visualisation proche des 10^o pour les écrans 3D commercialisés actuellement) mais également s'avancer ou s'éloigner de l'écran 3D ; cela permet ainsi à plusieurs personnes de visualiser simultanément le relief sur ce type d'écran.

Si l'observateur se rapproche à nouveau, les zones de vues se rétrécissent, donc l'utilisateur peut très bien avoir un oeil dans la zone 234 et l'autre non pas dans la zone voisine 345 mais 456 : ceci n'empêchera pas l'utilisateur de voir en stéréo avec trois zones différentes à l'écran. La figure 26 montre un exemple de la visualisation monoculaire depuis la zone 234 : dans le a., la caméra 2 envoie une image constituée uniquement de chiffres 2, et caetera pour les caméras 3 et 4. On peut remarquer les deux zones de transition à chaque tiers de l'écran malgré un mélange d'images provenant de plusieurs caméras. Quelques ruptures géométriques apparaissent dans ces zones de transition lors de l'affichage d'une vraie image dans la partie b. En fait, ces ruptures ne sont visibles que pour les objets de la scène qui ont du relief, c'est à dire qui sont situées en avant-plan ou en arrière-plan par rapport au plan de l'écran : en effet, la position du plan de l'écran dans la scène à visualiser a son importance : elle permet de sélectionner les objets que l'on souhaite visualiser en relief (voir [**?**] pour plus de détails).

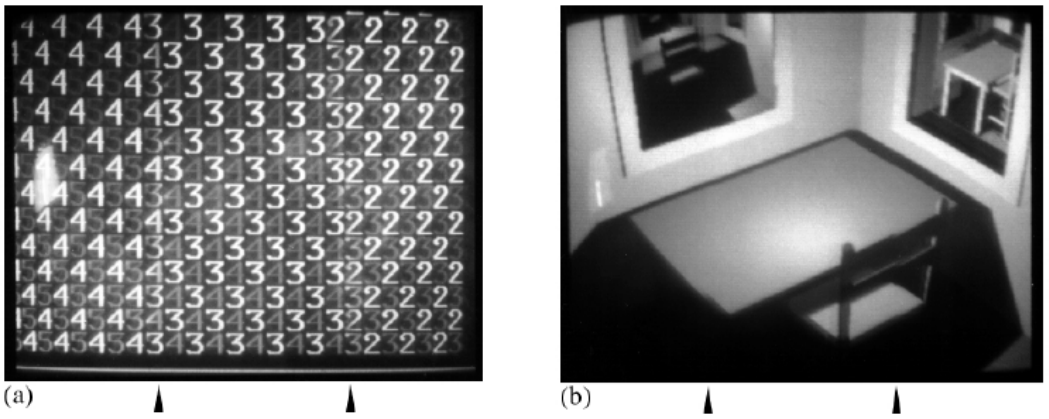

FIG. 26 – Visualisation monoculaire d'une image sur écran 3D depuis la zone de vue 234, proche de l'écran ; a. Pour chaque caméra, transmission du numéro de cette caméra ; b. Pour chaque caméra, transmission d'une image provenant d'une scène réelle.

Troisième partie

Les écrans 3D Newsight

Après une vaste étude bibliographique sur l'intérêt des écrans autostéréscopiques multivues et leur utilisation dans le domaine de la recherche scientifique, nous avons décidé d'investir dans un écran 3D qui soit pleinement configurable afin que nous puissions exploiter au maximum les capacités de ce type d'écran. Deux grandes marques sont disponibles sur le marché des écrans 3D : Philips et Newsight. Nous allons voir plus en détail quelles sont les principales caractéristiques de ces écrans.

1 Spécificités des écrans Philips

Philips a une stratégie de commercialisation de leurs écrans 3D plus orientée vers le grand public et vers le développement innovant de la TV 3D. Ils proposent une gamme de produits où neufs vues sont affichées simultanément avec un panneau lenticulaire comme système d'autostéréoscopie. Le panneau lenticulaire est un bon choix puisqu'il permet de conserver l'intensité lumineuse émise par l'écran LCD mais leur système de visualisation est trop contraint pour pouvoir être utilisé dans le cadre de travaux scientifiques.

En effet, l'écran accepte en entrée une seule vue avec la carte de profondeur associée, puis un traitement matériel (processeur intégré dans l'écran 3D) permet de reconstruire en temps réel les huit autres vues en utilisant les données de la carte de profondeur. Les valeurs de profondeur sont traduites en valeurs de disparité (décalage en pixels entre deux vues succcessives pour chaque pixel de l'écran), elles sont utilisées pour reprojeter les pixels de l'image originale dans les huit autres vues puis un algorithme de remplissage permet de compléter chaque vue affichée à l'écran : voir schéma de la figure 27.

A partir de ce schéma, nous pouvons en déduire les relations suivantes :

$$\frac{v_0}{f} = \frac{-p_x}{p_z}, \frac{v_1}{f} = \frac{-p_x + b_1}{p_z}$$

avec C_0 le centre optique de la caméra d'origine, C_1 le centre optique de la caméra suivante virtuelle (première vue à recalculer), f la distance focale des caméras, (p_x, p_y, p_z) les co-

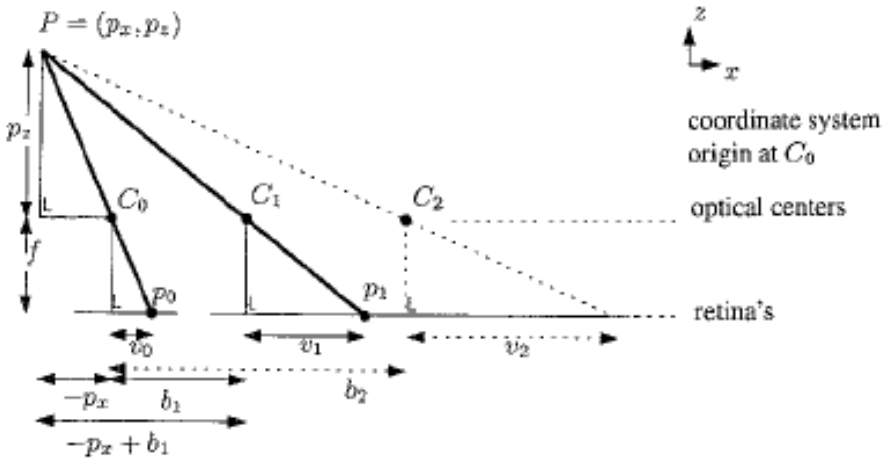

FIG. 27 – Projection du point P sur les pellicules de trois caméras parallèles.

ordonnées du point P dans la scène 3D enregistrée et p_z la valeur extraite de la carte de profondeur pour le pixel P_0.

La disparité peut être exprimée à partir des relations précédentes :

$$d_1 = v_1 - v_0 = \frac{f.b_1}{p_z}$$

La disparité d_1 est donc inversement proportionnelle à la valeur de profondeur p_z et proportionnelle à la fois à la distance focale f et à la distance inter-caméras b_1.

Le calcul de la disparité des autres caméras peut être ramené au calcul de la disparité de la caméra 1 :

$$d_i = v_i - v_0 = \frac{f.b_i}{p_z} = \frac{f.c_i.b_1}{p_z} = c_i.d_1$$

avec d_i le calcul de disparité de la caméra i et c_i une constante qui vaut généralement i pour avoir un écartement inter-caméras fixe, comme présenté dans la partie technique II (section 1.4).

Le calcul des positions des pixels dans les nouvelles vues peut donc être automatisé et réduit à une addition et une multiplication par pixel traité :

$$v_i = d_i + v_0 = c_i.d_1 + v_0$$

en stockant pour chaque image non pas la valeur de profondeur p_z mais la valeur de disparité d_1 en chaque pixel de l'image. Plus de détails, notamment sur le traitement des problèmes d'occlusion, sont disponibles dans l'article [**?**].

Nous ne voulons pas utiliser ce type d'écran car nous préférons envoyer nous même les différentes vues à l'écran afin d'obtenir un effet stéréoscopique de meilleure qualité. En

effet, la politique de Philips est la simplicité d'utilisation par l'observateur : ils visent plutôt l'affichage stéréoscopique de DVD 2.5D (où la valeur de disparité est disponible en chaque pixel de chaque trame de la vidéo) et d'autres contenus numériques diffusés par de futures chaînes TV 3D : le principal intérêt de ce format est son faible coût de stockage pour des séquences vidéos (pour chaque pixel, un canal de disparité est ajouté en complément des canaux R, G et B usuels).

Notre choix s'est donc naturellement porté sur les écrans 3D fabriqués par la société Newsight où les différentes vues à afficher ne sont pas *calculées à la volée* par le matériel mais peuvent être *précalculées* puis envoyées à l'écran (nous verrons comment dans la section suivante).

2 Spécificités des écrans Newsight

La société Newsight ne fabrique pas réellement les écrans, elle utilise des écrans LCD *classiques* de marque Fujitsu/Siemens qu'elle habille d'une barrière de parallaxe (appliquée devant l'écran LCD) afin d'obtenir des écrans 3D à huit vues.

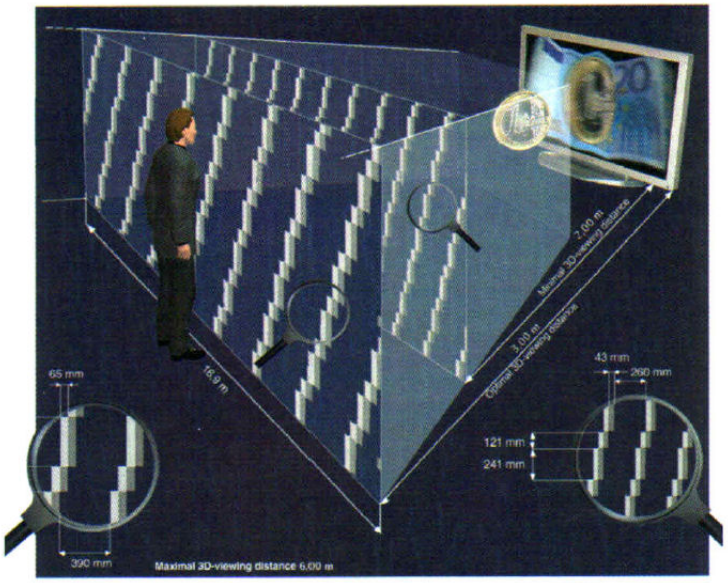

FIG. 28 – Une barrière de parallaxe, composée de fentes obliques, est apposée devant un écran LCD pour séparer les images affichées et ainsi permettre une visualisation en stéréoscopie par Newsight.

La barrière de parallaxe n'est pas constituée de trous mais de fentes afin de conserver au maximum la résolution verticale de l'écran LCD, comme expliqué dans la section 3.2 de la

partie I. La particularité des écrans 3D fabriqués par Newsight est l'orientation diagonale (et non verticale) de ces fentes afin de conserver au maximum la résolution horizontale de l'image 3D : voir figure 28.

Sur le modèle 30 pouces disponible chez France Télécom R&D, l'écran LCD est en haute résolution (HD, 1280×768 pixels), avec 480 barrières (fentes) sur la largeur, ce qui représente une fente tous les $\frac{8}{3}$ de pixels (nous verrons dans la section suivante pourquoi un tel chiffre apparaît). L'écran 3D propose donc une résolution spatiale de 480×288 pixels par oeil. En effet, la résolution finale de l'écran 3D est diminuée $\frac{8}{3}$ selon la largeur et la hauteur de l'écran LCD. Ce système permet ainsi de ne pas avoir de forte disproportion entre la résolution horizontale et la résolution verticale de l'écran 3D : avec des fentes verticales, la résolution spatiale de l'écran 3D aurait été de 160×768 pixels par oeil (très faible horizontalement donc images stéréo de piètre qualité).

Sur le modèle 23 pouces disponible dans l'équipe Bunraku de l'IRISA, l'écran LCD est en très haute résolution (Full HD, 1920×1200 pixels), avec 720 barrières sur la largeur de l'écran. La résolution spatiale de l'écran 3D est donc assez importante, permettant ainsi une visualisation stéréoscopique de bonne qualité : 720×450 pixels par oeil.

Sachant que la distance de visualisation optimale est de 3 mètres et que la distance interoculaire moyenne est de 6.25 centimètres, nous pouvons calculer une déviation angulaire entre les directions des deux yeux de 1.2^o, soit un angle de visualisation totale (sur les 8 vues) d'environ 10^o. De plus, sachant que l'écran de 23 pouces mesure 51.5 centimètres de largeur et que les deux yeux voient un même point sur l'écran 3D (sur la barrière de parallaxe) qui est en fait projeté sur l'écran LCD avec une différence de $\frac{1}{3}$ de pixel, nous pouvons en déduire la distance focale entre l'écran LCD et la barrière de parallaxe : 4.3 millimètres. Aucune information sur ces données techniques n'ont pu être obtenues du fabricant (Newsight), il a fallu effectuer des tests sur ces écrans 3D afin de déduire des informations pertinentes et ainsi pouvoir exploiter au mieux les caractéristiques des écrans.

2.1 Outils proposés par Newsight

Les drivers et outils proposés par Newsight sont exclusivement réservés au système d'exploitation Windows. Le logiciel *Newsight Media Player* est proposé afin de visualiser des séquences d'images ou des vidéos sur l'écran 3D. Aucune information n'est disponible sur la manière dont sont entrelacées les 8 vues pour pouvoir visualiser correctement la stéréoscopie. Le logiciel demande :

- soit 8 images en entrée et se charge de faire l'entrelacement de manière logicielle juste avant d'afficher le résultat à l'écran ;
- soit une seule grande image multi-vues de résolution 5760×3200 constituée d'une mosaïque des 8 images de résolution 1920×1200 (nous verrons dans la section suivante comment sont organisées les 8 images) et fait l'entrelacement de manière logicielle juste avant d'afficher le résultat à l'écran ;
- soit une image au format propriétaire *x3dim* contenant déjà les 8 images entrelacées, stockée sous forme compressée.

Il en va de même pour les vidéos, ces trois modes de représentations sont également acceptés (avec un format propriétaire *x3dmv*.

La bibliothèque *opengl32.dll* est également proposée pour pouvoir visualiser des applications openGL en stéréoscopie sur l'écran 3D. Il suffit de mettre le fichier dans le répertoire de l'exécutable pour qu'il soit utilisé à la place de celui disponible dans le répertoire *System32* de Windows.

Des plugins pour une visualisation en stéréoscopie sont proposés pour les logiciels *3d studio max* et *maya*.

Il existe également des outils de création de contenus aux formats propriétaires *x3dim* (pour les images) et *x3dmv* (pour les vidéos).

Nous avons également utilisé l'outil *X3D zWarper* ou plutôt la version logciel en ligne de commande *zWarpCmd.exe* fournie pour pouvoir construire 8 vues à partir d'une seule vue + la carte de profondeur associée. Le warping est effectué par extrusion de la géométrie à partir de la carte de profondeur puis reprojection en chaque centre optique des caméras virtuelles : la figure 29 illustre le processus de génération des 8 vues.

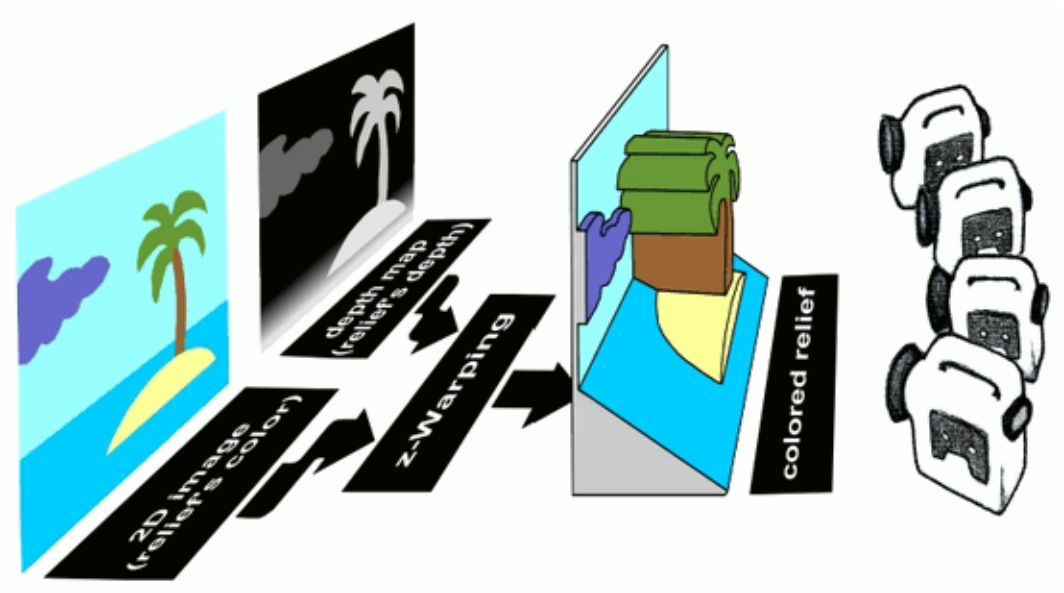

FIG. 29 – Construction de 8 vues pour un écran 3D Newsight à partir d'une vue et sa carte de profondeur associée.

Le résultat n'est pas très convaincant pour des images fixes mais est intéressant pour effectuer des séquences vidéos. Nous l'avons appliqué au filmounet *Wood Side Story* réalisé par des membres de l'équipe Bunraku de l'IRISA qui ont fourni une séquence d'images plus

des cartes de profondeurs extraites depuis *3d studio max*. Voici la ligne de commande utilisée :

```
zWarpCmd.exe -numviews 8 -numframes 2 -startframe 1150 -width 853
 -height 480 -angle 16.0 -fixpoint 0.0 -zoom 1.05
 -irgb Scene_II_1150_2350_%04#.bmp -iz Scene_II_1150_2350_Zdepth_%04#.bmp
 -o vues_finales_%#_%04#.bmp
```

L'outil nécessite de générer des images de 480 pixels de hauteur. Pour pouvoir compresser les données, nous utilisons le format de compression DivX. Seulement, cette compression ne peut pas être utilisée pour des images entrelacées, c'est pour quoi nous transformons les 8 images en une seule grande image (mosaïque des 8 vues), comme demandée par le logiciel *newsight media player*. Ces grandes images multi-vues ont une résolution trop importante pour pouvoir être compressées directement en DivX (une vidéo en DivX est limitée à du full HD, i.e. 1920 pixels de large), donc nous redimensionnons au préalable les 8 vues en 640×360 avec *virtual dub* puis nous construisons la mosaïque des 8 vues en 1920×960. Pour construire cette mosaïque, nous avons développé un petit outil en C appelé *generer_ multivues.exe*. Il faut le lancer avec une syntaxe du style (une aide est disponible dans l'outil et plus de détails seront donnés dans la section 3) :

```
generer_multivues.exe vues_finales_1151_1.ppm 1
```

Enfin, nous utilisons *virtual dub* pour compresser au format DivX la séquence vidéo constituée de l'ensemble des mosaïques. Le logiciel *newsight media player* est ensuite capable de faire la décompression DivX et l'entrelacement des 8 vues en temps réel sans difficulté sur un Pentium IV 3.6 GHz (à 24 images par seconde).

2.2 Limitations

Les outils proposés par Newsight ne permettent pas d'utiliser pleinement l'écran 3D pour plusieurs raisons. La première, c'est qu'ils ne permettent pas d'utiliser l'écran sur des systèmes d'exploitation autres que *Windows*.

De plus, il n'est pas possible de connaître la manière dont les 8 images sont entrelacées avant d'être affichées à l'écran. La technique est brevetée et non réutilisable. Nous avons donc procédé à des séries de tests pour pouvoir déduire les motifs d'entrelacement après expérimentation.

La bibliothèque *opengl32.dll* est très lente, elle divise le nombre d'images par seconde par un facteur 15 à 20 sur des scènes openGL de petite taille (avec peu de polygones), ce qui devient rapidement problématique pour espérer pouvoir se déplacer interactivement. De plus, elle n'accepte pas certaines fonctionnalités récentes des cartes graphiques, comme l'utilisation de *shaders* successifs, de *frame buffer object*, de rendus multi-passes, etc.

Le plugin pour *3dstudiomax* ne fonctionne pas très bien, il y a de grosses erreurs de visualisation qui apparaissent spontanément et les calculs d'illumination ne sont pas gérés correctement.

3 Développements réalisés sur l'écran Newsight 23 pouces

3.1 Motif d'entrelacement simplifié

L'entrelacement des 8 vues n'est pas effectué horizontalement sur 8 pixels comme nous avions pu le constater dans la partie II, mais en suivant la direction des fentes obliques de la barrière de parallaxe. De plus, l'entrelacement est effectué non pas sur les pixels successifs de l'écran LCD mais directement au niveau des canaux RGB de l'écran LCD. C'est pour cette raison que les barrières de parallaxe sont aussi rapprochées les unes des autres (tous les $\frac{8}{3}$ de pixels contre 8 pixels habituellement) : sur la figure 30, la partie à gauche montre le motif d'entrelacement au niveau des canaux RGB ; pour obtenir le motif complet, il suffit d'appliquer ce motif comme mosaïque, et c'est déjà ce qui a été fait pour la dernière colonne apparaissant en jaune. On peut remarquer en fushia la répartition des pixels provenant de la première vue : ces pixels sont bien répartis selon la direction des fentes obliques mais seule une partie des valeurs de ces pixels est utilisée (un seul des trois canaux RGB est conservé). De plus, tous les pixels ne sont pas utilisées, c'est ce que l'on peut constater sur la partie de droite : elle représente l'image entrelacée pour la première vue uniquement.

R	G	B	R	G	B	R	G	B
1	2	3	4	5	6	7	8	1
2	3	4	5	6	7	8	1	2
2	3	4	5	6	7	8	1	2
3	4	5	6	7	8	1	2	3
4	5	6	7	8	1	2	3	4
4	5	6	7	8	1	2	3	4
5	6	7	8	1	2	3	4	5
6	7	8	1	2	3	4	5	6
6	7	8	1	2	3	4	5	6
7	8	1	2	3	4	5	6	7
8	1	2	3	4	5	6	7	8
8	1	2	3	4	5	6	7	8

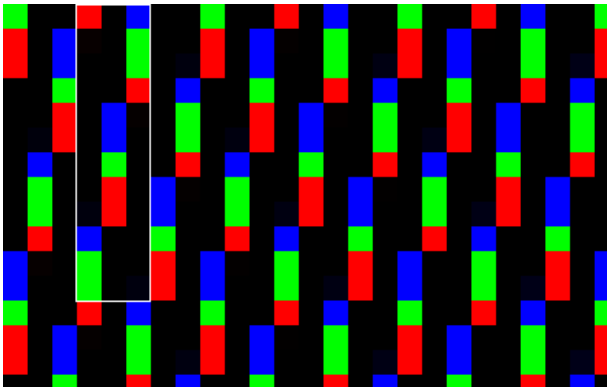

FIG. 30 – A gauche, schéma permettant de construire l'image entrelacée, ce schéma doit être reproduit en mosaïque pour couvrir la totalité de la résolution de l'écran LCD. A droite, image entrelacée représentant la vue 1 blanche et les autres vues noires en réutilisant le motif décrit sur la gauche : la partie encadrée en blanc représente exactement le motif de gauche avec sa colonne jaune incluse (largeur de 3 pixels).

Dans cette configuration, les 8 vues sont supposées être de la même résolution que l'image finale entrelacée, affichée sur l'écran LCD, i.e. 1920×1200 pixels. Nous remarquons tout de suite que les vues de départ sont très largement sur-échantillonnées car beaucoup de pixels sont inutilisés dans l'image finale (beaucoup de pixels noirs sur la partie droite de l'image).

Dans le cas de la visualisation 3D d'une première vue entièrement constituée de pixels rouges et des autres vues complètement noires, nous obtiendrions une image entrelacée qui ressemblerait à celles de la figure 31.

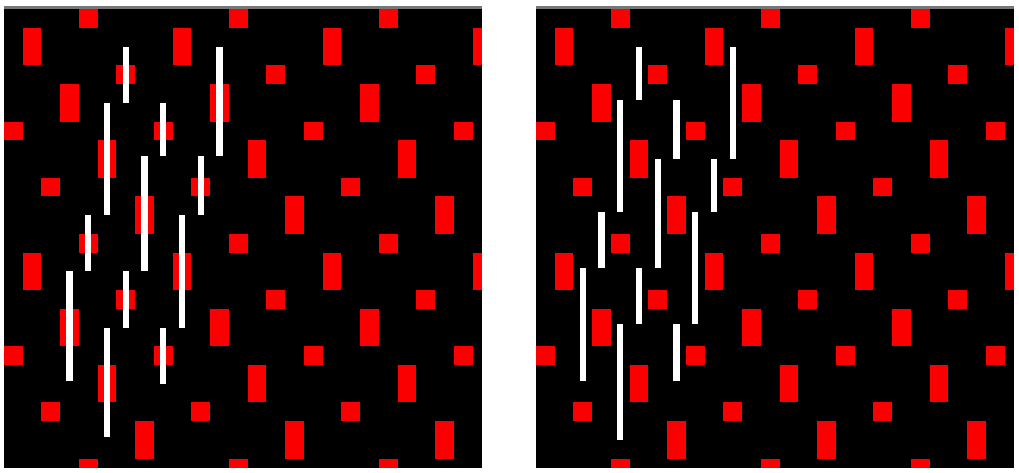

FIG. 31 – Image entrelacée représentant la visualisation 3D de la première vue composée de pixels rouges et des autres vues composées uniquement de pixels noirs. A gauche, visualisation d'une image rouge sombre depuis l'oeil gauche ; A droite, visualisation d'une image complètement noire depuis l'oeil droit.

Sur cette figure, nous avons ajouté en blanc les fentes de la barrière de parallaxe afin de simuler ce que peuvent voir les deux yeux d'un observateur positionné devant l'écran 3D : soit une image rouge sombre, soit une image complètement noire. Evidemment, ce n'est pas un contenu stéréoscopique *valide* dans le sens où ça ne représente aucun relief concret mais ça permet de comprendre comment fonctionne l'écran 3D. Nous avons dessiné les barres obliques en alternant par des segments de barre verticale courte ou longue comme sur le schéma de la figure 28 provenant du manuel de l'écran de 30 pouces, mais nous avons plutôt l'impression que ces barres sont réellement obliques. Si nous avions apposé ces barres sur le schéma de droite de la figure 30, nous aurions vu par le même principe une vue blanche pour un oeil et une vue noire pour l'autre.

3.2 Entrelacement des 8 vues par le CPU

En réalité, le motif d'entrelacement des 8 vues ne peut pas être restreint à la répétition du schéma de la figure 30, il est plus compliqué que cela. Nous avions vu précédemment que chaque pixel de l'image finale entrelacée était construit à partir du canal R d'une vue, du canal G d'une seconde vue et du canal B d'une troisième vue : ce principe ne change pas, mais à cela il faut ajouter une pondération sur chacun des canaux, c'est à dire le canal R

de l'image finale est majoritairement construit à partir du canal R d'une vue (et le reste de la pondération provient du canal R de la vue suivante), le canal G de l'image finale est majoritairement construit à partir du canal G d'une autre vue (et le reste de la pondération provient du canal G de la vue suivante) et de même pour le canal B. Et pour compléter le tout, les pondérations sont différentes d'un canal à un autre et sont également différentes d'un pixel de l'écran à un autre afin de tenir compte de la déviation angulaire entre les directions de vue provenant d'un même centre optique (par exemple l'oeil) vers les différents pixels de l'écran. En effet, pour un observateur placé au centre de l'écran, le rayon traversant la fente centrale va atteindre le pixel central de l'écran (orthogonal à la direction de la fente) alors qu'un rayon traversant une fente sur le bord de l'écran ne sera pas exactement orthogonal à l'écran et atteindra ainsi un pixel légèrement distant du pixel orthogonal à la fente traversée. Voici sur la figure 32 une partie du motif d'entrelacement utilisé pour les vues 1 et 2. On peut noter, par exemple, que la composante rouge du pixel de coordonnées $(1, 3)$ est constitué de $\frac{230}{255}$ème de la valeur rouge du pixel de mêmes coordonnées provenant de la vue 1 et de $\frac{25}{255}$ème de la valeur rouge du pixel de mêmes coordonnées provenant de la vue 2 (même si ce pixel paraît vert dans la vue 2, il contient tout de même une petite composante de rouge).

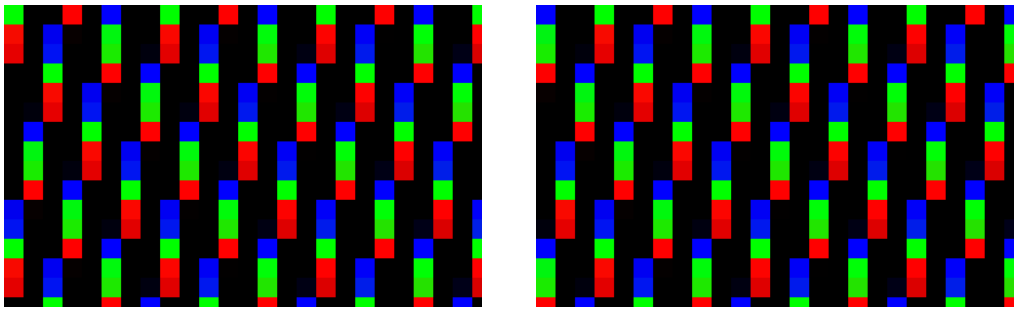

FIG. 32 – Utilisation du *vrai* motif d'entrelacement. A gauche, image entrelacée représentant la vue 1 blanche et les autres vues noires. A droite, image entrelacée représentant la vue 2 blanche et les autres vues noires.

L'utilisation de barres obliques et un entrelacement au niveau des canaux RGB des pixels est une technique propre à la société *newsight* qui lui permet d'afficher 8 vues sur l'écran 3D sur un angle restreint de quelques degrés (7.7^o) en donnant l'impression que l'on visualise un flot continu de vues sur ce cône de visualisation de 7.7^o. Lorsque ces paramètres de placement de caméra sont respectés, l'observateur n'est pas capable de compter le nombre de vues affichées devant lui, même s'il se déplace sur toute la zone de visualisation des 8 vues. Le mélange des pixels se fait au niveau des canaux RGB, avec 3 vues mélangées par pixel et des vues différentes pour les mêmes canaux d'un pixel à son voisin, ce qui casse subtilement la transition présente habituellement entre deux vues consécutives.

Etant donné la complexité des motifs d'entrelacement (jusqu'à trois vues consécutives peuvent être utilisées pour construire un seul canal couleur d'un pixel), nous avons préféré

stocker les motifs sous forme d'une image couleur pour chaque vue (comme dans la figure 32) plutôt que de stocker les valeurs de pondération et de numéros de vue dans un seul tableau comme l'exemple de la figure de gauche 30. Nous stockons donc 8 filtres de la résolution de l'image finale 1920×1200 pixels et faisons l'interpolation linéaire des 8 images en pondérant chaque canal de chaque pixel par la valeur stockée dans les filtres.

Pour cela, nous avons développé un outil appelé *generer_multivues.exe* qui permet, à partir de 8 vues de n'importe quelle résolution, de reconstruire soit l'image entrelacée directement affichable sur l'écran 3D (sans avoir à utiliser *Windows*), soit l'image mosaïque des 8 vues affichable par le logiciel *newsight media player*. Pour l'image mosaïque, elle est constituée comme sur la figure 33, i.e. sa largeur est 3 fois la largeur d'une vue et sa hauteur est $\frac{8}{3}$ de fois la hauteur d'une vue.

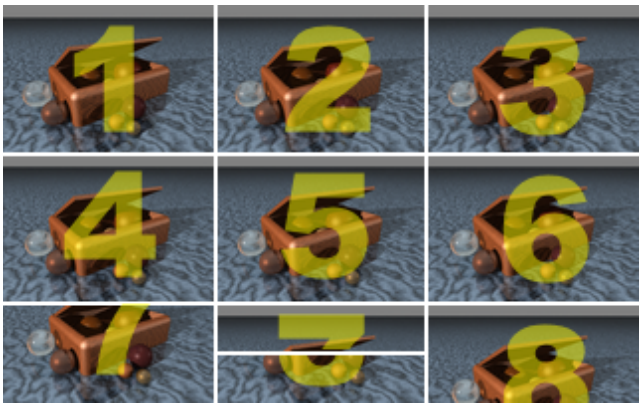

FIG. 33 – Représentation de l'image mosaïque constituée de 8 vues non entrelacées : une compression par bloc (type jpeg) peut donc être appliquée sur ce type d'image : les numéros indiquent la position relative de chaque vue dans l'image complète.

Pour l'image entrelacée, elle peut être définie avec/sans interpolation bilinéaire (anti-aliassage) lors du redimensionnement des 8 vues pour que l'image finale soit à la résolution maximale de l'écran LCD, avec/sans conservation du ratio largeur/hauteur, avec/sans redimensionnement des 8 vues pour générer l'image entrelacée (toutes ces fonctionnalités sont décrites dans l'aide fournie avec l'outil). Les 8 filtres spécifiques à l'écran newsight 23 pouces sont chargés lors de la génération de l'image entrelacée mais ils peuvent être remplacés par d'autres filtres spécifiques à un autre modèle d'écran 3D.

Concernant les performances de calcul de l'image entrelacée, le CPU nécessite 150ms pour effectuer l'entrelacement de 8 images en pleine résolution (1920×1200) avec des pondérations stockées en *int* et 180ms avec des pondérations stockées en *float* (en langage C/C++), ce qui restreint l'entrelacement CPU à 6 images par seconde en pleine résolution (tests effectués sur un Pentium IV 3.6GHz). Nous verrons pourquoi l'entrelacement en pleine résolution est important dans la section suivante.

3.3 Génération des 8 vues dans le RayTracer PovRay

Afin de tester l'algorithme d'entrelacement des 8 vues par le CPU, nous avons généré des images de scènes virtuelles par lancer de rayon en utilisant le logiciel PovRay. La connaissance de la géométrie exacte des scènes virtuelles nous ont permis de positionner correctement l'ensemble des 8 caméras pour générer 8 images correspondant exactement aux différentes positions de l'oeil face à l'écran 3D et dans les conditions d'observation optimale (caméras virtuelles situées à 3m de l'écran virtuel, avec un écartement inter-caméra de 6.25cm).

Nous avons modifié les valeurs des constantes *Initial_Clock* et *Final_Clock* dans le fichier de définition des options de rendu (*.ini*) afin de pouvoir automatiquement générer les 8 vues, puis nous avons utilisé la variable *clock* pour pouvoir paramétrer la caméra de chaque vue directement dans le fichier de description de scène (*.pov*). Nous avons tout de même dû modifier le code source du logiciel (logiciel en open source) pour pouvoir supprimer les tests d'orthogonalité entre la direction de vue passant par le centre de chaque caméra et le centre de l'écran virtuel, et la direction de l'axe optique qui est invariante, quelle que soit la position de la caméra (les 8 caméras sont positionnées avec une configuration parallèle : voir section 1.4 de la partie II).

La nouvelle version de *povray*, intitulée *POV-Ray for generating multi-views image*, a été obtenue en modifiant le code source uniquement des fichiers *parse.cpp*, *render.cpp* et *vbuffer.cpp*. La majeure partie des modifications est de la maintenance de la cohérence des variables définies dans le fichier de description de scènes (*.pov*). Voici le fichier de description de scène minimal, contenant les informations nécessaires à la description des caméras :

```
// Version : A utiliser avec la version modifiée de POV-Ray,
//   "POV-Ray for generating multi-views image"

// à mettre dans le fichier .pov de votre scène
// remplace la précédente déclaration de "camera"
// permet de paramétrer très simplement la caméra pour générer les 8 vues
// attention : il faut également vérifier les options du fichier .ini

// ====================================================
// les seuls points à modifier dans ce fichier !!!
// ====================================================
// position de la camera "centrale" (entre la vue 4 et la vue 5)
#declare point_location = <0, 6, -12>;
// position du centre de l'image
#declare point_look_at = <0,3,0>;

// angle séparant deux caméras (1.1ř est l'écartement théorique pour l'écran NewSight)
#declare angle_entre_deux_vues = 1.1;

// ====================================================
// ne pas toucher la définition de la caméra, ce sera toujours la même, quelle que soit la scène
// on interdit seulement de placer une caméra regardant dans la direction y (ou -y)
// ====================================================
#declare vecteur_right = vnormalize(vcross(<point_location.x - point_look_at.x, 0,
                                            point_location.z - point_look_at.z>, y));
#declare vecteur_up = vnormalize(vcross(vecteur_right, point_location - point_look_at));
```

```
camera {
//rotation de la caméra autour de l'axe des Y pour générer les 8 vues
//   location point_location
   location vaxis_rotate(point_location - point_look_at, vecteur_up, clock) + point_look_at
//direction et right permettent de définir l'angle de vue (seulement ||direction|| est utile ici)
   direction z
//right et up définissent le plan image : fixes durant la rotation de la caméra
//   right x*image_width/image_height
   right vecteur_right*image_width/image_height
//   up y
   up vecteur_up
//look_at modifie la direction + la distance camera-plan_image
//  (ni l'angle de vue, ni la direction des vecteurs right et up -- seulement leur longueur)
// -> le centre des images est fixe
   look_at point_look_at
}
```

Dans ce fichier, l'utilisateur a simplement 3 nouvelles variables à renseigner pour décrire son système constitué de 8 caméras : la position de la caméra centrale *point_location*, la position du centre de l'écran dans la scène *point_look_at* et l'angle inter-caméras *angle_entre_deux_vues*. En fait, seul l'angle représente une réelle donnée supplémentaire permettant de passer d'une à huit caméras. La figure 34 permet de visualiser le système de caméras représenté dans ce fichier.

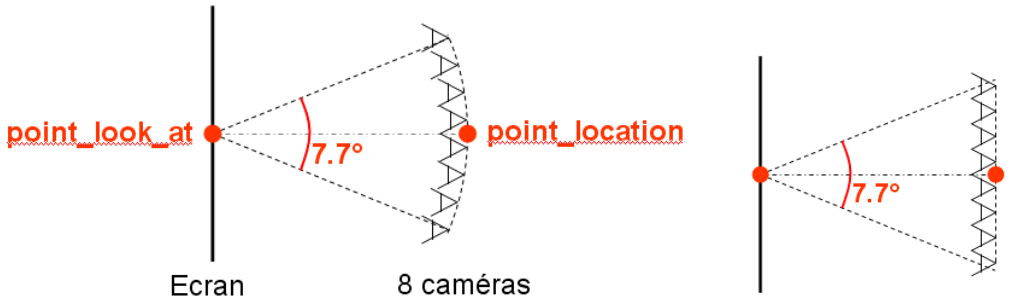

FIG. 34 – A gauche, système de caméra redéfini dans PovRay pour gérer le rendu avec 8 vues. Les 8 caméras sont parallèles à gauche comme à droite (axe optique orthogonal au plan de l'écran). La seule différence entre ces deux systèmes, c'est la position des caméras : radiale à gauche, sur une même ligne à droite.

Même si les 8 caméras sont placées sur un cercle autour du centre de l'écran, elles n'en restent pas moins définies comme parallèles (axe optique commun) et non radiales grâce à une redéfinition des vecteurs *right* et *up* représentant les deux axes de l'écran 3D, indépendants de la position de la caméra. Nous avons défini les 8 caméras par rotation autour de l'écran car nous n'étions pas encore très sûr du meilleur choix pour le placement des caméras lorsque nous avons effectué les séries de tests avec povray. Mais comme l'écartement entre les caméras les plus éloignées est seulement de 7.7^o, la différence entre les deux configurations de la figure

34 est négligeable, que ce soit en terme de position de caméra ou d'angle inter-caméras (sur la figure, la valeur angulaire n'est pas représentative, elle a été exagérée pour donner de l'espace entre les caméras). Rien n'empêche tout de même de modifier le fichier de configuration des caméras (la partie interne de *camera {...}* dans le fichier *.pov*) mais il faut être extrêmement prudent sur les modifications apportées : elles peuvent avoir des conséquences facheuses sur le comportement du logiciel : il ne faut pas inverser l'ordre de déclaration des variables dans cette partie, sous peine de voir le contenu de certaines variables affecté par la modification d'une autre variable lors du chargement de ce fichier dans povray.

Enfin, il faut également penser à modifier le fichier de définition des options (*.ini*) pour paramétrer le nombre de caméras. Voici un exemple de son contenu :

```
# Version : A utiliser avec la version modifiée de POV-Ray,
#           "POV-Ray for generating multi-views image"

# à mettre dans le fichier .ini lu lors du rendu de votre scène
# options minimales pour générer les 8 vues de l'écran NewSight de résolution 1920x1200
# attention : il faut également vérifier les paramètres de la caméra dans le fichier .pov

Output_File_Type=P

[960x600, 8 views]
Width=960
Height=600
Initial_Clock=-3.5
Initial_Frame=1
Final_Clock=3.5
Final_Frame=8

[1920x1200, 8 views]
Width=1920
Height=1200
Initial_Clock=-3.5
Initial_Frame=1
Final_Clock=3.5
Final_Frame=8
```

Dans ce fichier, nous pouvons également configurer plusieurs rendus différents avec notamment des résolutions d'images différentes.

3.4 Batterie de tests de l'écran 3D avec PovRay

Nous avons effectué une série de tests avec une même scène, la *woodbox*[1], afin de vérifier que l'outil d'entrelacement développé produit des images valides et que les paramètres de configuration de caméras ont été correctement estimés. La figure 35 présente les 8 vues de cette scène soit sous la forme d'une mosaïque, soit sous la forme entrelacée. Bien évidemment, l'image entrelacée est directement visualisable sur l'écran 3D et n'est pas destinée à être visualisée sur un écran 2D classique. Néanmoins, son affichage sur un écran 2D permet très

[1] dont les images référencées ci-après se trouvent dans http ://www.irisa.fr/prive/kadi/Ecran3D

facilement de repérer où a été positionné l'écran virtuel dans la scène (région du sol qui n'est pas floue).

Nous avons choisi de construire cette scène (inspirée d'un exemple fourni dans *povray*) car les matériaux des objets constituant cette scène sont représentés par des textures procédurales 3D (bois sur la boîte, marbre sur le sol) et ont des propriétés de réflectance assez variées (de la diffusion, de la spécularité et même de la transparence).

3.4.1 Position du plan image (écran)

Le plan image doit être placé au centre de la scène pour bénéficier au maximum du relief : à la fois celui d'avant plan et celui d'arrière plan. A titre d'exemple, nous pouvons visualiser l'image `woodbox_1920x1200_Ang1.1_Dist13_all_3D.ppm` qui place le plan image (écran) au centre de la boîte en bois (distance 13 entre le plan image et la caméra dans la scène *.pov*). Lorsque le plan image est placé en avant plan, la scène semble derrière l'écran mais l'arrière plan peut paraître flou. C'est ce qui se passe dans l'image `woodbox_1920x1200_Ang1.1_Dist6.5_all_3D.ppm` où le plan image est à la distance 6.5 de la caméra. Inversement, si le plan image est placé en arrière plan, la scène semble sortir de l'écran mais l'avant plan peut à son tour devenir flou : voir l'image `woodbox_1920x1200_Ang1.1_Dist19.5_all_3D.ppm` où le plan image est à la distance 19.5 de la caméra.

Pour réduire le flou tout en conservant une scène entièrement en avant plan (ou arrière plan), il faut réduire l'angle entre deux vues consécutives.

3.4.2 Angle inter-caméras

D'après nos mesures, l'angle entre deux vues consécutives doit être de 1.1^o pour un affichage optimal du relief sur l'écran 3D newsight. Si l'angle est réduit, la perception du relief est moins importante. A titre de comparaison, il suffit de regarder l'image `woodbox_1920x1200_Ang0.5_Dist13_all_3D.ppm` où l'angle inter-caméra a été réduit à 0.5^o. Sur cette image, le relief de la boîte est moins important mais il permet de visualiser le sol en avant plan avec plus de facilité (il est bien moins flou). Si l'angle est trop important, le relief est accentué et l'affichage peut devenir flou en avant ou arrière plan : voir l'image `woodbox_1920x1200_Ang2.0_Dist13_all_3D.ppm` où l'angle inter-caméra de 2.0^o est beaucoup trop important (avant plan complètement flou).

La réduction de l'angle théorique est donc intéressante dans le cadre de la visualisation d'un environnement complet et son agrandissement s'appliquera plutôt à la visualisation d'objets afin d'accentuer leur relief.

3.4.3 Orientation du plan image (écran)

Le plan image doit être fixe dans le repère global (de la scène) pour pouvoir visualiser cette scène en stéréoscopie. La version modifiée de PovRay (*PovRay for generating multi-*

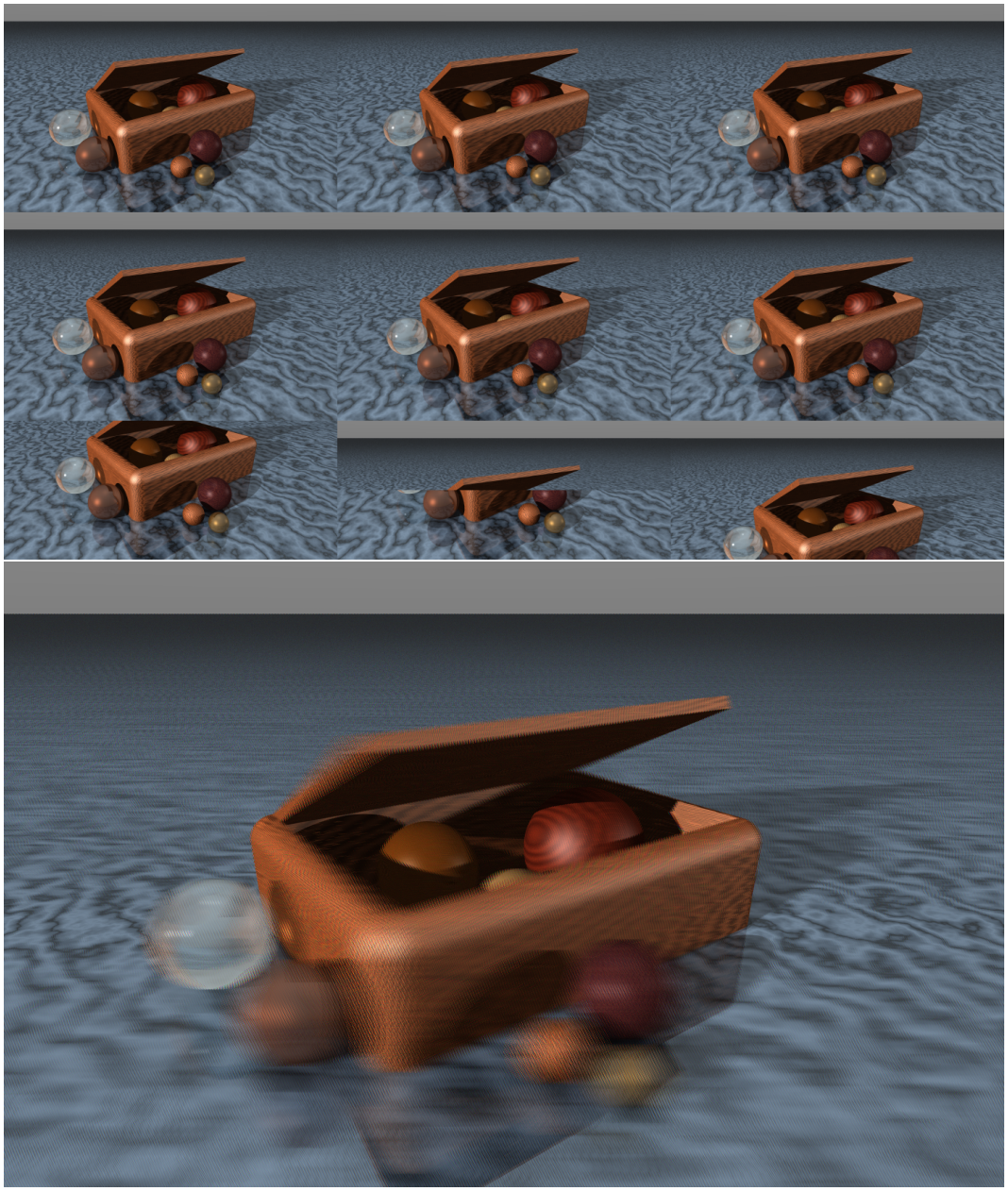

FIG. 35 – En haut, mosaïque des 8 vues visualisable en relief avec l'outil *newsight media player*. En bas, image entrelacée avec notre propre outil, directement visualisable.

views image) permet justement de fixer l'orientation de ce plan image dans la direction de la caméra centrale. Si *PovRay* est utilisé dans sa version standard, au lieu d'obtenir l'image `woodbox_1920x1200_Ang1.1_Dist13_all_3D.ppm`, nous obtiendrons l'image `woodbox_1920x1200_Ang1.1_Dist13_ImageOrthoCamera(PovRay)_all_3D.ppm`. L'effet obtenu est très désagréable, nous avons l'impression de tourner dans la scène, l'image entrelacée est bien moins nette. Le sol en avant plan tourne très fortement, le résultat est visible même sur un écran 2D : sur la figure 36, le sol est flou sur toute sa surface ; nous pouvons retrouver la position du plan image dans la scène grace au centre de rotation du sol.

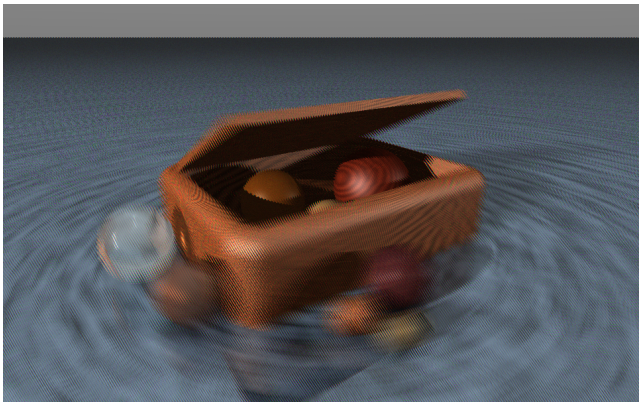

FIG. 36 – En haut, mosaïque des 8 vues visualisables en relief avec l'outil *newsight media player*. En bas, image entrelacée avec notre propre outil, directement visualisable.

La technique de réorientation du plan image dans la direction de la caméra est utilisée pour fabriquer les vidéos appelées *Pulfrich* où l'utilisateur, après avoir filmé une scène avec une seule caméra en tournant autour d'un point fixe, va simplement reprojeter 8 trames successives de la vidéo sur l'écran 3D puis décaler toutes les vues (la vue 2 devient la vue 1, la vue 3 devient la vue 2, etc) pour insérer la trame suivante en huitième vue et ainsi de suite jusqu'à diffuser toute la vidéo sur l'écran 3D. Toutefois, il faut noter que ce type de contenu ne fonctionne que pour filmer des objets loin de la caméra, sans aucun élément en avant plan de la scène : ce n'est pas de la *vraie* stéréoscopie.

3.4.4 Profondeur de champ

La profondeur de champ peut être réduite pour provoquer du flou, il suffit d'augmenter l'ouverture (comme un appareil photographique). Pour cela, chaque pixel de chaque vue n'est pas calculé en lançant un rayon au centre de ce pixel mais en faisant la moyenne sur un certain nombre d'échantillons (ce nombre est paramétrable et influe sur la qualité de l'image finale) en lançant plusieurs rayons par pixel avec une petite déviation aléatoire de la direction visée (la taille maximale de la déviation simule l'ouverture).

Dans le fichier de description de la scène (*.pov*), il suffit de rajouter quelques paramètres relatifs à la configuration de la caméra :

```
// --------------------------------
// gestion de la profondeur de champ
// --------------------------------
// point de netteté (sur la direction point_location-point_look_at)
//   focal_point <0,1,0> // centre de la boîte net (indépendamment de la position du plan image)
//   focal_point <0,3.5,-6> // avant plan net
//   focal_point <0,-1.5,6> // arriere plan net
   focal_point point_sur_focale
// ouverture (0.0 pour une profondeur de champ infinie, i.e. pas de flou)
   aperture 1.0
   blur_samples 200
   confidence 0.9
   variance 1/10000
```

On peut noter que le focus (plan de netteté) n'est pas obligatoirement défini sur le plan image (écran). L'image `woodbox_1920x1200_Ang1.1_Dist13_all_3D.ppm` est définie pour une profondeur de champ infinie (ouverture réduite au minimum), il n'y a donc pas de flou, toute la scène est nette sur chacune des 8 vues. Néanmoins, l'image finale entrelacée paraît toujours floue pour les éléments trop en avant plan ou trop en arrière plan du plan image (ce plan est indépendant du plan de focus de la caméra d'enregistrement mais dans le cas d'un oeil humain, il y a effectivement concordance entre le plan de focus de la caméra et le plan image puisque le plan image contient le point de convergence des 8 caméras). Plus simplement, si les deux yeux d'un observateur regardent un objet, le point de convergence des deux yeux est fixé sur cet objet et comme l'observateur voit cet objet net, cela signifie que chaque oeil fait une mise au point sur l'objet donc le plan de focus de chaque oeil est confondu avec le plan contenant le point de convergence.

Nous pouvons accentuer l'effet de flou de profondeur déjà présent dans l'entrelacement des 8 vues en réduisant la profondeur de champ des caméras : dans l'image `woodbox_1920x1200_Ang1.1_Dist13_FocusSurBoite_all_3D.ppm`, le paramètre *aperture* est fixé à 0.5 afin de réduire la profondeur de champ à la taille de la boîte en bois et le point de convergence des 8 caméras est laissé au centre de la boîte : cela donne une impression d'immensité de la scène, avec un sol qui se prolonge à l'infini.

Les images qui suivent ne sont pas réalistes puisque le plan de focus est déplacé soit sur la sphère la plus proche (image `woodbox_1920x1200_Ang1.1_Dist13_FocusSurSpherePres_all_3D.ppm`), soit sur la sphère la plus loin (image `woodbox_1920x1200_Ang1.1_Dist13_FocusSurSphereLoin_all_3D.ppm`), avec une ouverture très large (*aperture* est fixée à 1) mais le point de convergence des 8 caméras est resté fixé au centre de la boîte en bois : par conséquent, le reste de la boîte devient très vite flou.

3.4.5 Résolution du plan image (écran)

Nous rappelons que l'écran LCD 23 pouces que nous avons utilisé pour effectuer ces tests est en full HD, i.e. qu'il est défini par une résolution de 1920×1200 pixels. Nous avons vu

que l'entrelacement est effectué différemment d'un canal RGB à un autre mais que, lors de l'entrelacement des huits vues définies à la résolution maximale de l'écran, il y avait une perte d'information : voir figure 32 pour plus de détails sur les pixels noirs. En effet, cette perte d'information est due au fait que l'entrelacement ne va utiliser qu'une information sur huit horizontalement pour chaque vue (voir figure 30) : nous pouvons donc raisonnablement réduire la résolution horizontale par $\frac{8}{3}$, soit 2.66 (il faut diviser par 3 car il y a trois canaux dans un pixel, R, G et B). En revanche, il ne faudrait pas toucher à la résolution verticale.

Seulement, ce type de réduction impose de générer des vues avec des pixels non carrés (2.66 fois plus hauts que larges) et l'affichage de telles vues sur un écran 2D n'est plus envisageable sans un rééchantillonnage de l'image (sinon, l'image est complètement déformée). Par souci de simplicité, nous avons effectué des tests avec une réduction de la résolution par 2 selon les deux axes (horizontal et vertical) soit une réduction de 4 au total. L'image résultante entrelacée
`woodbox_960x600_Ang1.1_Dist13_SansAA_all_3D.ppm` laisse apparaître des défauts sur les textures, même si elle est visuellement satisfaisante. En effet, il y a perte de la texture de bois sur le côté gauche de la boîte. Pour adoucir les contours dans l'image, nous pouvons également utiliser une interpolation bilinéaire afin de générer les pixels manquants dans chaque vue lors de l'entrelacement : le résultat visuel (image
`woodbox_960x600_Ang1.1_Dist13_AvecAA_all_3D.ppm`) est plus agréable (suppression du crénelage sur le contour des ombres) même si cela génère un peu plus de flou.

En utilisant huits vues en pleine résolution, l'image finale entrelacée ne va conserver qu'$\frac{1}{8}$ de l'information originale, et ce qui peut paraître surprenant, c'est qu'en réduisant la quantité d'information originale seulement par 4, nous avons une perte sensible de détails dans l'image. Evidemment, ceci vient du principe que l'entrelacement ne se fait pas entre pixels mais entre canaux RGB des pixels donc la réduction ne peut excéder 2.66.

En conclusion de cette série de tests, nous pouvons admettre que générer les 8 vues en pleine résolution donne toujours de meilleurs résultats mais qu'une réduction par 2 selon la hauteur et selon la largeur permet de réduire les temps de calcul tout en conservant une qualité d'image très acceptable : en fait, cette réduction peut être appliquée lorsque nous voulons générer des vidéos, par exemple (lorsqu'une image est animée, la quantité de détails affichés peut être réduite sans être détectée visuellement).

3.5 Entrelacement des 8 vues par le GPU

Etant donné le format de stockage des filtres/masques de chaque vue de l'écran (sous forme d'image contenant les pondérations pour chaque canal couleur de chaque pixel), l'idée sous-jacente est de stocker ces 8 filtres sous forme de textures en mémoire vidéo, de stocker également chacune des 8 vues sous forme de textures et ensuite d'appliquer un programme par GPU (*shader*) pour effectuer des opérations entre ces 16 textures et ainsi générer l'image finale entrelacée. Le principal intérêt de cette technique est la parallélisation massive des calculs effectués par le GPU puisque pour tous les pixels d'une vue (on pourrait dire *pour tous les canaux de tous les pixels d'une vue*), le GPU va effectuer la même opération : produit

de la valeur du canal du pixel par la pondération, stockée au même endroit dans le filtre, puis stockage dans l'image finale, toujours au même endroit.

Voici donc le code, très succint, du *fragment shader* permettant d'entrelacer les 8 vues (en GLSL) :

```
// Fragment shader for compositing of the 8 rendered images
static const GLchar compositingFragmentShaderSource[] =
"uniform sampler2D images[8];                "
"uniform sampler2D masks[8];                "
"void main()                        "
"{                          "
" vec4 color0 = texture2D(images[0], gl_TexCoord[0].st); "
" vec4 color1 = texture2D(images[1], gl_TexCoord[0].st); "
" vec4 color2 = texture2D(images[2], gl_TexCoord[0].st); "
" vec4 color3 = texture2D(images[3], gl_TexCoord[0].st); "
" vec4 color4 = texture2D(images[4], gl_TexCoord[0].st); "
" vec4 color5 = texture2D(images[5], gl_TexCoord[0].st); "
" vec4 color6 = texture2D(images[6], gl_TexCoord[0].st); "
" vec4 color7 = texture2D(images[7], gl_TexCoord[0].st); "
" vec4 mask0 = texture2D(masks[0], gl_TexCoord[0].st);   "
" vec4 mask1 = texture2D(masks[1], gl_TexCoord[0].st);   "
" vec4 mask2 = texture2D(masks[2], gl_TexCoord[0].st);   "
" vec4 mask3 = texture2D(masks[3], gl_TexCoord[0].st);   "
" vec4 mask4 = texture2D(masks[4], gl_TexCoord[0].st);   "
" vec4 mask5 = texture2D(masks[5], gl_TexCoord[0].st);    "
" vec4 mask6 = texture2D(masks[6], gl_TexCoord[0].st);   "
" vec4 mask7 = texture2D(masks[7], gl_TexCoord[0].st);   "
" gl_FragColor = color0 * mask0 + color1 * mask1       "
"              + color2 * mask2 + color3 * mask3     "
"              + color4 * mask4 + color5 * mask5     "
"              + color6 * mask6 + color7 * mask7;     "
"}                          ";
```

Cette technique suppose donc d'utiliser une carte graphique récente, dotée de 16 unités de texture afin de pouvoir laisser les 16 textures chargées en mémoire vidéo pendant l'application du *fragment shader*.

Etant donné que ce code est intégré à une application OpenGL, nous avons pensé qu'il serait judicieux de développer une application interactive OpenGL qui entrelacerait les 8 vues provenant de 8 rendus successifs et ainsi afficherait un contenu OpenGL en relief et en temps-réel sur écran 3D.

3.6 Rendu temps réel et en relief sous OpenGL

Evidemment, effectuer 8 rendus successifs diminue déjà par 8 le taux de rafraichissement d'une application OpenGL standard, il est donc nécessaire de minimiser les opérations supplémentaires afin de conserver un rendu final sur écran 3D qui soit interactif. Nous avons développé une bibliothèque de rendu OpenGL interactif sur écran 3D utilisant les GPU.

Nous avons utilisé des *framebuffer objects (FBO)* afin de stocker chaque rendu de caméra dans un *framebuffer* qui puisse être directement lu comme une texture en mémoire vidéo. Nous évitons ainsi les multiples copies coûteuses du *framebuffer* principal vers une texture.

Cela évite également de devoir changer de contexte OpenGL lorsque l'on passe du rendu d'une caméra au rendu de la caméra suivante (le changement de contexte OpenGL est coûteux en temps et obligatoire lorsqu'on utilise les *pixel buffers*).

Nous avons développé le code OpenGL en C++ et avons utilisé la bibliothèque GLUT pour définir l'interface afin que le code soit portable, contrairement à la bibliothèque fournie par newsight (compatible Windows, Linux, etc).

La principale difficulté est la paramétrisation des caméras afin de générer 8 rendus en effectuant le moins d'opérations possible sur chaque redéfinition des plans de clipping de chaque caméra. Comme l'écran est fixe d'une caméra à l'autre et que les caméras sont placées selon une configuration parallèle (section 18 de la partie II), le déplacement des plans de clipping d'une caméra à l'autre est contraint à suivre l'axe de déplacement des caméras (1 seul degré de liberté). Nous pouvons donc nous ramener de simples calculs de rapports de distance en utilisant des relations de Thalès pour évaluer les nouvelles valeurs de clipping fournies à la fonction *glFrustrum*.

Nous avons ensuite effectué des tests de performance sur une scène de faible complexité géométrique où le taux de rafraîchissement est de 160 images par seconde en résolution maximale 1920×1200 pour un affichage classique (sans relief). Lorsque nous utilisons notre bibliothèque de rendu OpenGL pour visualiser le relief sur l'écran 3D, nous obtenons un rendu interactif de 13 images par secondes, soit des performances divisées par 12 en moyenne. Ces tests ont été effectués avec un processeur Pentium IV 3.6GHz et une carte graphique Quadro FX 256Mo. Nous avons également testé les performances en diminuant la résolution des 8 rendus de caméra à 960×600 pixels et nous obtenons ainsi un rendu final temps réel de 32 images par seconde, soit des performances divisées seulement par 5 et une qualité d'image en relief très correcte. En effet, les filtres ainsi que le rendu final restent en pleine résolution et nous activons l'interpolation bilinéaire sur les textures provenant des 8 rendus.

Nous pouvons donc constater que la parallélisation des calculs permet un entrelacement GPU beaucoup plus rapide qu'avec le CPU (sans compter qu'avec le CPU, nous avions uniquement entrelacé les 8 vues, la phase de rendu de chaque vue n'était pas incluse dans les performances).

Nous avons amélioré notre bibliothèque de rendu multi-caméras sous OpenGL pour qu'elle puisse supporter n'importe quel type de rendu pour chaque caméra. Actuellement, le rendu de chaque vue peut être complexe, effectué en plusieurs passes, faire appels à ses propres *shaders programs* et utiliser ses propres *framebuffer objects* pour stocker les résultats intermédiaires du rendu multi-passes.

3.7 Automatisation du paramétrage de l'écran 3D sous OpenGL

L'interactivité obtenue sous OpenGL, contrairement à la construction d'images fixes, pose le problème de la redéfinition en temps réel de la meilleure position de l'écran dans la scène virtuelle et du meilleur angle inter-caméras pour pouvoir effectuer une navigation interactive confortable.

En effet, l'affichage d'une scène en relief sur un écran 3D nécessite de placer correctement l'écran dans la scène à visualiser en se posant, à chaque rendu multi-caméras, les questions suivantes :

- quels objets doivent être en avant plan pour donner l'impression de sortir de l'écran ?
- quels éléments doivent rester en profondeur de l'écran ?
- quel est l'élément le plus important affiché à l'écran ?
- doit on le mettre en valeur en plaçant le focus sur lui ?
- si oui, doit on augmenter l'écartement inter-caméras pour qu'il soit plus en relief ?
- doit on prendre en compte le fait que toute modification de l'écartement inter-caméras risque de déstabiliser le relief des autres éléments de la scène visibles sur l'écran ?
- quel compromis doit-on faire lorsqu'il y a un choix à faire ?
- etc.

Nous nous rendons vite compte que le paramétrage d'un écran 3D dans une application interactive n'est pas trivial à réaliser, qu'une quantité astronomique de questions peuvent être soulevées et ceci, de manière chronique. En réalité, il n'est pas possible de répondre à ces questions de manière tranchée sans la connaissance a priori du type d'application utilisée.

Dans la bibliothèque OpenGL décrite dans la section 3.6, nous avions paramétré les touches *haut* et *bas* du clavier pour pouvoir déplacer dynamiquement la position de l'écran dans la scène virtuelle, et les touches *droite* et *gauche* pour pouvoir modifier l'écartement inter-caméras. C'est une solution qui permet de s'adapter à la configuration de la scène visualisée et de changer interactivement l'objet que l'on veut observer en relief.

Sébastien Hillaire[2] a travaillé pendant son stage de master dans l'équipe Bunraku et a développé une application OpenGL de navigation interactive dans un environnement virtuel où il tient compte de la sémantique des objets qu'il affiche pour changer l'aspect visuel de son appplication. Un poids est affecté à chaque polygone de la scène et lors du rendu, il met l'accent sur l'objet de plus forte sémantique proche de la zone pointée par la souris. Pour cela, il effectue un rendu en deux passes et applique un effet de flou dans sa scène en définissant une profondeur de champ et en définissant la zone de focus proche de l'objet pointé.

Nous avons intégré la bibliothèque OpenGL définie dans la section 3.6 aux travaux de Sébastien afin de visualiser son application sur l'écran 3D. Nous avons ainsi pu redéfinir la position de l'écran dans la scène en le positionnant automatiquement au centre de l'objet 3D pointé par la souris.

Néanmoins, la navigation reste inconfortable si l'on ne modifie jamais l'écartement inter-caméras. En effet, lorsque l'utilisateur pointe la souris sur un objet en arrière plan et qu'un objet se situe en premier plan, celui-ci se retrouve à être très désagréable à regarder car il donne l'impression de trop sortir de l'écran (l'objet se dédouble, il faudrait réduire l'écartement inter-caméras).

Nous avons mis en place une solution simple qui consiste à utiliser le tampon de profondeur du rendu d'une caméra afin de déterminer, en fonction des valeurs minimale et maximale de profondeur, quelle sera la disparité maximale des polygones devant être affichés à l'écran

[2] actuellement doctorant de l'équipe Bunraku de l'IRISA

(voir la section 1 pour plus de détails sur la disparité). Ensuite, à partir d'une valeur seuil de disparité maximale (par exemple, un pixel entre deux vues consécutives), nous pouvons modifier la valeur d'écartement inter-caméras afin de toujours positionner le polygone le plus loin (en avant ou arrière plan) à exprimer un relief prononcé mais jamais trop excessif à visualiser. Ceci évite d'avoir des situations où des objets en premier plan se dédoublent complètement (réduction automatique de la valeur d'écartement inter-caméras) mais ceci évite également d'afficher une scène avec un relief trop faible (augmentation automatique de l'écartement inter-caméras pour redonner du relief à la scène visualisée).

Comme nous disposons également d'une sémantique sur chaque polygone affiché à l'écran, nous avons optimisé le calcul de l'écartement automatique inter-caméra pour qu'il soit adapté aux objets de très forte sémantique. Concrètement, ceci permet de visualiser les objets intéressants avec un maximum de relief, même s'ils se situent tous très loin dans la scène. La contrepartie, c'est que le sol en avant plan peut devenir très flou si sa sémantique indique qu'il est peu important.

L'ajout de la profondeur de champ à l'entrelacement des 8 vues donne un effet visuel intéressant car il est complémentaire au flou introduit par l'entrelacement : il permet par exemple d'atténuer le crénelage existant sur les contours des objets hors de la zone de focus en lui ajoutant un peu de flou.