Conception d'outils avancés pour le codage de la vidéo 3D : fusion d'images couleur binoculaires perçues

Xavier Pineau

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL ASSOCIE DE PAYS DE LOIRE

MEMOIRE

présenté en vue d'obtenir
le DIPLÔME d'ingénieur CNAM

SPECIALITE : INFORMATIQUE

OPTION : systèmes d'information

par

Xavier Pineau

Conception d'outils avancés pour le codage de la vidéo 3D : Fusion d'images couleur binoculaires perçues

Soutenu le : 16 décembre 2011

JURY :

Présidente:	Isabelle Métais	(présidente du jury, professeur Cnam Paris)
Membres:	Henri Briand	(professeur Ecole polytechnique Nantes)
	Vincent Ricordel	(LUNAM Université, Université de Nantes)
	Bogdan Cramariuc	(IT Center for Science and Technology, Bucarest, Roumanie)
	Benoît Le Gal	(Directeur des systèmes d'information, Arjowiggins Graphic)
	Jacques Demontoux	(Attaché de direction et chef de projet, Girard Hervouet)

Remerciements

« Ce n'est pas la force, mais la persévérance, qui fait les grandes œuvres. »
Samuel Johnson.

Je tiens à remercier et saluer ma famille qui m'a soutenu et motivé pendant ces dernières années. Je remercie particulièrement ma femme Elodie pour m'avoir encouragé et supporté pendant ces six années de cours du soir malgré mes longues absences. Je remercie ma mère et mon père de m'avoir toujours encouragé dans mes projets personnels. Merci à mes deux filles Héloïse et Mélissa qui sont ma source de motivation, surtout dans les moments difficiles.

Merci à Vincent Ricordel d'avoir accepté de m'encadrer et de s'être toujours montré à l'écoute et très disponible durant ces neuf mois de stage, sans lui ce mémoire n'aurait jamais vu le jour. Je lui suis très reconnaissant pour l'intérêt qu'il a porté à mon travail, ses conseils scientifiques ainsi que pour son amabilité.

Merci à Marcus Barkowsky pour l'expertise et le suivi qu'il m'a apporté pour la réalisation du projet de ce mémoire. J'ai ainsi pu apprécier ses qualités scientifiques et la pertinence de ses remarques.

Par ailleurs, un merci tout particulier aux membres de l'équipe IVC du laboratoire de l'IRCCyN pour leur accueil et le temps qu'ils m'ont accordé pour les discussions liées à ce document.

Je remercie également Jacques Demontoux, Benoit Le Gal et Yann Quesnel pour m'avoir donné l'envie et les possibilités de réaliser des projets passionnants. Ce projet personnel de formation est la continuité du travail réalisé avec eux.

Liste des abréviations

3DTV 3D Télévision

3DVC 3D Vidéo Coding

ACSED Analyse et Commande des Systèmes à Evénements Discrets

ADTSI Analyse et Décision en Traitement du Signal et de l'Image

ANR Agence Nationale de la Recherche

CIE Commission Internationale de l'Eclairage

CNAM Conservatoire National des Arts et Métiers

CNRS Centre National de la Recherche Scientifique

EQM Erreur Quadratique Moyenne

FTV Free ViewPoint TV

HD Haute Définition

IETR L'Institut d'Electronique et de Télécommunications de Rennes

IMAX Image Maximum

IRCCyN L'Institut de recherche en communications et cybernétique de Nantes

IVC Images et Vidéocommunications

IVGI Ingénierie Virtuelle pour le Génie Industriel

INRIA L'Institut National de Recherche en Information et Automatique

LBG Algorithme de Linde, Buzo, Gray

LBG stéréo Algorithme de Linde, Buzo, Gray, amélioré dans le contexte du mémoire

LCD Liquid Cristal Display

LTCI Le laboratoire traitement et Communication de l'Information

MCM Méthode de Conception en Mécanique

MO2P Modélisation et Optimisation de Process de Production

MoVES Modélisation et Vérification des Systèmes embarqués

MVD Multiview, Video-plus Depth

QV Quantification Vectorielle

QVAr quantification vectorielle arborescente

PsyCoTec Psychologie, Cognition, Technologie

RENATER Réseau National de Télécommunications pour la Technologie, l'Enseignement et la Recherche

SVH Système Visuel Humain

TEMICS Traitement, modélisation et communication d'images numériques

UCS Uniform Chromaticity Scale

UMR Unite Mixte de Recherche

SLP Systèmes Logistiques et de Production

Table des matières

Chapitre 1

Introduction et contexte du mémoire

Le contexte global de ce mémoire est la compression vidéo 3D. Ce travail de mémoire est réalisé dans le cadre du projet ANR PERSEE visant à contribuer à la conception d'outils et méthodes avancés pour le codage de la vidéo 2D et 3D [38]. La vidéo 3D existe depuis le début du XX^{e} siècle, mais elle n'a pas encore séduit le grand public. Depuis la télévision noir et blanc, plus tard la télévision couleur et maintenant la télévision numérique, les technologies 3D permettent d'apporter au téléspectateur des impressions subjectives plus réelles concernant la scène visualisée.

Si cette technologie ne s'est pas imposée dans les foyers, c'est que la qualité proposée n'était pas à la hauteur des attentes du grand public. C'est avec l'apparition récente des télévisions HD que le domaine de la 3D connaît un regain d'intérêt et est considéré comme la future révolution pour nos téléviseurs. Comme pour la vidéo HD, la vidéo 3D nécessite de l'équipement spécial pour la captation et la diffusion. Chacune de ces étapes nécessite des savoir-faire et des équipements particuliers qui commencent tout juste à être déployés aujourd'hui. Le monde de l'audiovisuel doit pour cela s'adapter pour produire et réaliser des contenus en 3D. Cette captation nécessite en premier lieu l'utilisation de caméras spécifiques dites "stéréoscopiques" qui permettent de restituer la richesse de la vision humaine. Pour profiter de la TV 3D, il faut bien entendu que le programme soit à l'origine filmé (ou synthétisé) en relief, mais aussi que les équipements de restitution soient compatibles.

Plusieurs acteurs du marché travaillent actuellement pour mieux maîtriser les complexes aspects de qualité pour l'affichage et de reconstruction 3D. Il existe plusieurs technologies de restitution comme l'anaglyphe, l'alisoscopie, la stéréoscopie ou même l'auto-stéréoscopie mais actuellement la technologie la plus simple et la plus mature est la stéréoscopie. Cette technologie copie la façon dont le SVH voit l'environnement en trois dimensions grâce au cerveau qui associe simultanément les informations captées par l'œil droit et par l'œil gauche. Les deux yeux sont séparés d'une distance de 6.5 cm environ et le cerveau interprète le décalage des deux images perçues pour restituer les informations de luminosité, de couleur, mais aussi de longueur de largeur et de profondeur. Basiquement, les télévisions 3D ne sont

pas si différentes des télévisions standard, la plus grosse différence réside dans le fait qu'au lieu d'envoyer une image pour une scène, on envoie deux images. Le téléspectateur doit être équipé de lunettes capables de cacher alternativement chaque œil de façon synchrone avec l'écran.

L'inconvénient principal de la diffusion en 3D réside dans le poids croissant des flux numériques. Quand une scène 2D est captée, une seule caméra enregistre une seule information. Une scène 3D nécessite au mieux une caméra bifocale (une pour chaque œil) qui multiplie les images et alourdit le flux. Il existe des techniques de codage/décodage qui permettent d'alléger ces flux, nous allons nous intéresser à l'une d'entre elle qui est la quantification vectorielle (QV). Cette technique utilise des méthodes d'apprentissage et d'indexation pour construire des listes de couleurs (ou dictionnaires) représentant au mieux les vidéos source puis transmettre des indices au décodeur en allégeant le flux. Classiquement les deux composantes d'une image stéréo sont quantifiées séparément mais c'est le même dictionnaire qui est utilisé pour les images gauche et droite. Plus précisément, une même couleur est affichée sur l'œil droit et l'œil gauche pour reconstituer une couleur dans l'espace 3D. Ce travail de mémoire va exploiter la fusion binoculaire des deux points colorés qui reconstruit le point coloré dans l'espace 3D. En théorie, la combinatoire entre les couleurs de dictionnaires gauche et droite, augmente le nombre de couleurs (fusionnées) perçues en 3D. Nous avons pour but d'exploiter cette théorie pour le codage d'images stéréo. Des problématiques se posent :

- comment construire des dictionnaires aux couleurs complémentaires ?
- comment calculer ces couleurs fusionnées comme le ferait le SVH ?
- comment évaluer les erreurs faites ?
- comment coder les images stéréo avec le nouveau codec ?

Pour le mémoire nous apporterons des réponses à ces questions, pour cela on fera appel à des approches :

- de quantification vectorielle pour la construction de dictionnaires de couleurs,
- de techniques d'affichage en 3D,
- de connaissances sur la perception des couleurs.

Dans une première partie après avoir abordé quelques généralités sur les technologies de la 3D et leurs domaines d'application, nous verrons les différences entre les technologies 3D ainsi que les avantages de l'utilisation de la stéréoscopie pour le projet. Je proposerai ensuite un état de l'art sur les différents espaces de la colorimétrie utilisés pour réaliser la fusion binoculaire. Nous pourrons ensuite présenter les éléments de la quantification vectorielle optimale et plus précisément l'étude de l'algorithme *LBG* qui sera adapté pour les besoins du projet.
Dans une deuxième partie nous allons réaliser un codeur/décodeur (codec) pour quantifier/déquantifier des vidéos stéréoscopiques par fusion binoculaire. Ce codec sera décliné en deux modes, l'un pour construire un prototype et tester l'algorithme et l'autre pour effectuer des tests sur des images "réelles". Les résultats ainsi obtenus seront comparés et étudiés. Après l'exposé de notre démarche et de nos travaux, nous pourrons faire les conclusions de nos recherches et donner des perspectives.
Enfin, dans un dernier chapitre, seront présentés les aspects techniques de notre environnement de travail, ainsi qu'une nécessaire partie gestion du projet.

Chapitre 2

Etat de l'art

2.1 Introduction

Ce chapitre a pour objectif d'apporter des notions sur les différents domaines abordés lors de ce mémoire. Après un bref rappel sur l'historique du cinéma en 3D, nous nous intéresserons aux différentes technologies utilisées pour représenter une image en 3D. Cette description a pour objectif de bien cerner les différentes techniques utilisées pour la représentation 3D et de mieux comprendre nos choix techniques.
La deuxième partie a pour objectif d'apporter des notions sur la manipulation des couleurs. Nous verrons une description simplifiée de ce qu'est une couleur, la façon de la percevoir et la façon de la représenter. Nous verrons aussi dans cette partie, une description des espaces colorimétriques utilisés durant ce mémoire en justifiant ces choix.

2.2 Les technologies de la 3D

2.2.1 Historique du cinéma 3D

Les premières découvertes sur la technologie de l'image 3D datent de l'année 1838. A cette époque, la projection stéréoscopique est la technique utilisée pour créer une illusion tridimensionnelle. « L'arrivée du train », une réalisation des frères Lumière en 1903, est le premier film en format 3D qui a été diffusé en salles. Cette toute première projection d'image tridimensionnelle a suscité une grande peur chez les spectateurs. Après ce premier essai réussi, d'autres cinéastes ont réalisé d'autres films et émissions en images 3D. Les inventeurs n'ont cessé, par ailleurs, d'améliorer les rendus de cette nouvelle technologie. Entre 1900 et 1946, de nombreux réalisateurs de cinéma se sont plongés dans la conception de films 3D, afin de découvrir les véritables secrets de la projection stéréoscopique.
Le début des années 1950 marque la période faste du cinéma 3D, grâce à la réalisation de « United Artists » intitulée « Bwana devil ». Suite à ce grand succès, les grands studios se tournent vers le tournage de films 3D, dont les plus célèbres sont « Hondo, l'homme du

désert » réalisé par John Farrow en 1953 et « Dial M for murder » de Alfred Hitchcock en 1954. Cependant, le succès du cinéma 3D n'a duré que quelques années, faute de matériels de visionnement plus performants comme les lentilles spéciales ou lunettes polarisées.
En 1973, le cinéma 3D refait de nouveau surface avec des films d'animation comme « Comin at Ya » et « Jaws 3D ». Les lunettes de visionnement (cf. figure 2.1) demeurent toutefois les mêmes, peu performantes.

FIGURE 2.1: Lunettes polarisées.

Ce n'est qu'en 1986, lors de la sortie du format IMAX 3D, que le cinéma 3D a connu un essor considérable. Ce succès s'explique notamment par l'émergence des technologies de visionnement. Aujourd'hui, de nombreuses technologies contribuent à la réalisation du cinéma 3D afin d'obtenir des rendus d'images parfaits [13].
Nous pouvons constater que la technologie 3D existe déjà depuis longtemps et est passée par plusieurs étapes techniques. Cependant, cette technologie n'a pas réussi à s'imposer tant que la qualité de restitution n'était pas suffisante pour les spectateurs et c'est grâce au numérique que la 3D suscite de nouveau l'intérêt. Le succès de la 3D ne sera garantit que si la qualité

des images est au rendez-vous.

2.2.2 Les technologies de la 3D

La visualisation d'une vidéo 2D est aujourd'hui un procédé maîtrisé. Passer de la 2D à la 3D nécessite l'utilisation de techniques pour percevoir le relief, il existe plusieurs méthodes pour y arriver. Nous allons regarder quelles sont les principales techniques utilisées, de la plus ancienne qui est l'anaglyphe jusqu'au procédé plus innovant de l'autostéréoscopie. Nous regarderons plus en détail la technique de stéréoscopie qui est la technique la mieux maîtrisée pour la fusion binoculaire et qui a été utilisée pour ce mémoire.

- La stéréoscopie

  On appelle stéréoscopie ou vision stéréoscopique notre capacité à voir les corps solides en relief. Cette faculté est liée à l'interaction entre nos deux yeux, en effet, quand nous observons un objet, chaque œil voit cet objet depuis un angle différent et envoie à notre cerveau une image légèrement différente (cf. figure 2.2). C'est cette différence entre les deux images qui est utilisée par notre cerveau pour construire la perception de la profondeur.

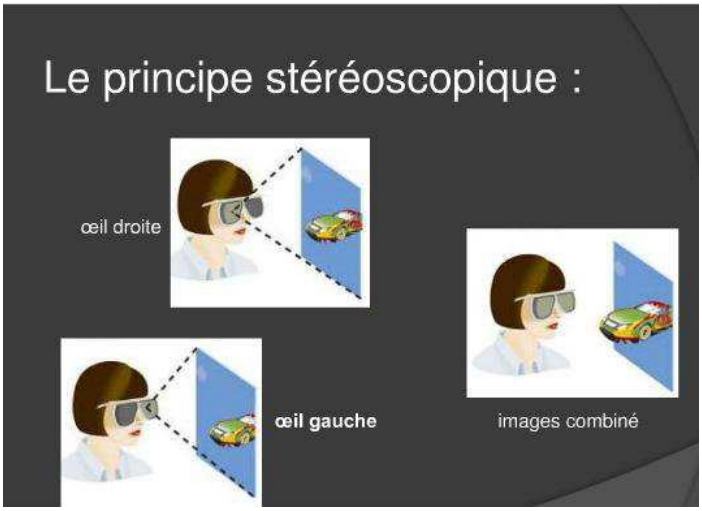

Figure 2.2: Principe de la stéréoscopie [16].

C'est donc le fait d'avoir deux yeux qui nous permet de capter, en même temps, deux images d'un même élément. Ces informations sont conduites par les nerfs optiques et acheminée au cortex visuel. Se produit alors mécanisme complexe qui rend possible la *fusion binoculaire* (fusion des deux images) et le résultat permet au cerveau de discerner reliefs et distances [9]. Ce qui est intéressant dans le cadre de ce projet, c'est que la perception d'un point coloré dans l'espace en 3D est donc le résultat de l'affichage de deux points colorés sur deux plans.

La stéréoscopie est donc l'ensemble des techniques mises en œuvre pour reproduire une perception du relief à partir de deux images planes [48]. Elle se base sur le fait que la perception humaine du relief se forme dans le cerveau lorsqu'il reconstitue une seule image à partir de la perception des deux images planes et différentes provenant de chaque œil (cf. figure 2.3). La réalisation d'images stéréoscopiques en 3D consiste donc à prendre deux images d'une même scène avec un léger décalage du point de vue comparable à celui perçu de nos yeux .

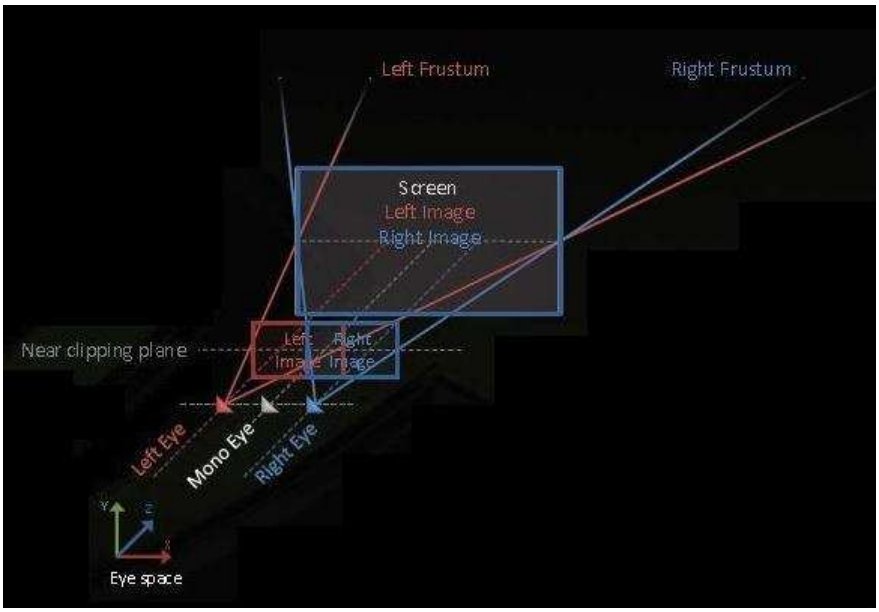

FIGURE 2.3: Deux yeux, un écran, deux images 2D, une image 3D [20].

Concernant l'écart inter-axial des deux points de vue, il ne doit pas nécessairement être égal à celui de nos yeux (soit 6 à 7 cm). Un écart plus grand permet de donner l'impression que la scène est plus petite qu'en réalité, tandis qu'un écart plus faible donnera l'impression que la scène est plus grande.

Nous pouvons parler ensuite de la notion de *disparité* ou *parallaxe* (cf. figure 2.4).

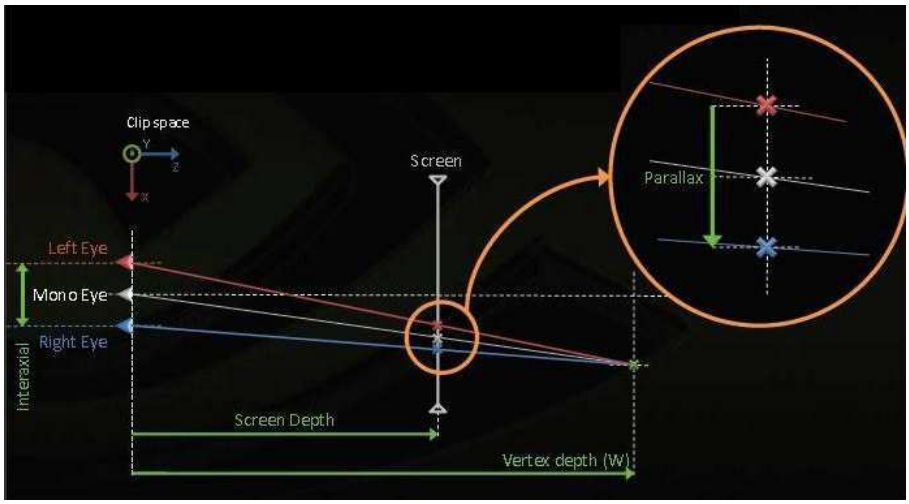

FIGURE 2.4: illustration de la *disparité* ou *parallaxe* [20].

Considérons deux yeux espacés par une distance inter-oculaire de 6,5 cm. Les yeux sont concentrés sur un point situé à une certaine distance. Ce point est appelé *point focal*. Le plan perpendiculaire à l'axe optique qui le coupe au point focal image s'appelle le *plan focal*(cf. figure 2.5) [24].

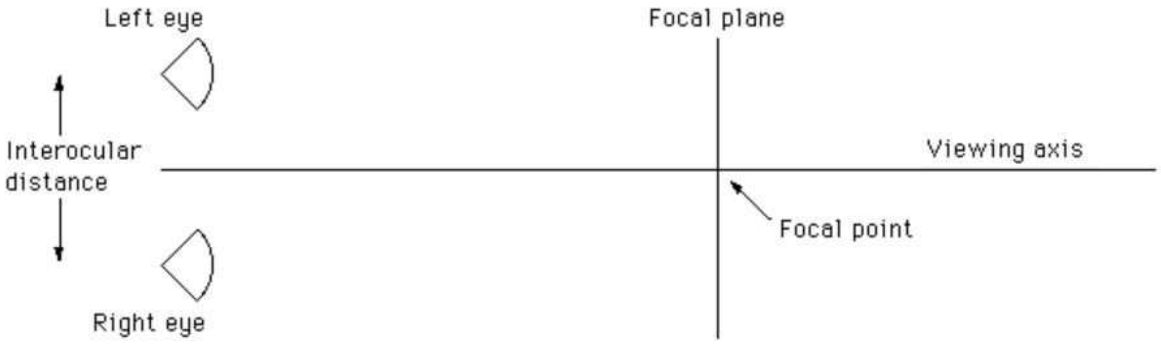

FIGURE 2.5: Illustration du point focal et plan focal [24].

Si un objet se trouve en dehors du *plan focal*, sa projection est illustrée sur le schéma 2.6. La projection pour l'œil gauche se trouve à gauche et celle pour l'œil droit à droite. La distance entre la projection des deux yeux est appelée *parallaxe horizontal*. Si la projection sur le plan focal de l'œil droit est à droite et de l'œil gauche à gauche, le

parallaxe est dit positive. Le maximum de *parallaxe positive* est atteint lorsque que la position de l'objet tend vers l'infini. La *parallaxe positive* est alors égale à la distance inter-oculaire [24].

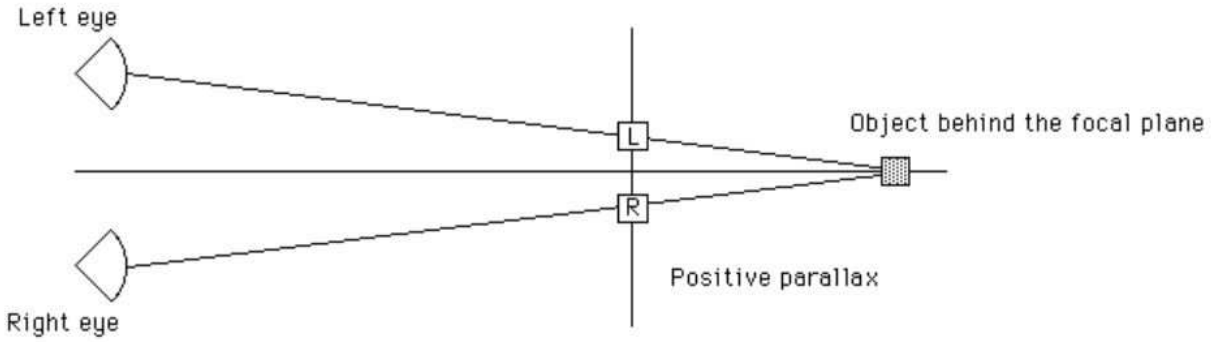

FIGURE 2.6: Illustration de la parallaxe positive [24].

Si la projection de l'objet se situe en avant du plan focal et que la projection de l'œil gauche se trouve à droite et vice versa, la *parallaxe horizontale* est alors appelée *parallaxe négative* (cf. figure 2.7).

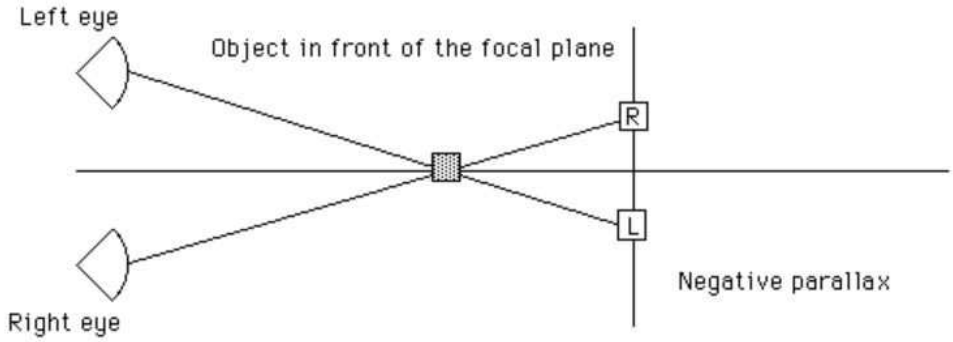

FIGURE 2.7: Illustration de la parallaxe négative [24].

Si la projection de l'objet sur le plan focal est confondue avec le point focal, ce parallaxe est appelée *parallaxe zéro* (cf. figure 2.8).

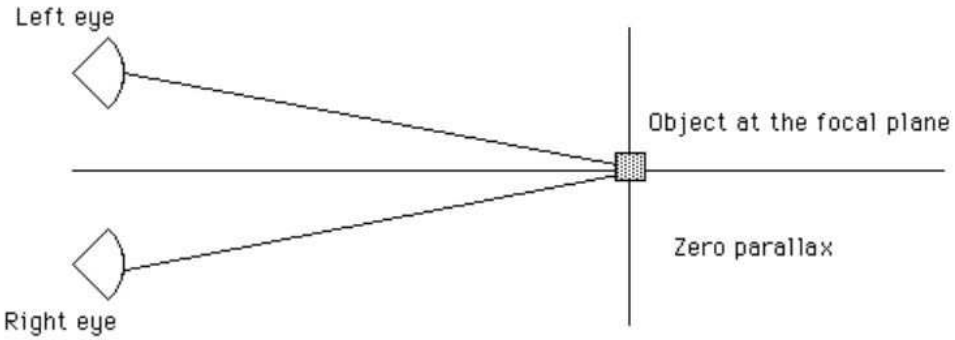

FIGURE 2.8: Illustration de la parallaxe zéro [24].

Par convention, si la *disparité* est positive le point est affiché en arrière de l'écran, si la *disparité* est négative le point est perçu devant l'écran, enfin pour une disparité nulle le point est sur l'écran. La *disparité* fixe donc la profondeur du point perçu. Classiquement, les points affichés sur le *plan focal* ont la même couleur qui correspond aussi à la couleur du point perçu dans l'espace 3D.

Il existe différentes manières pour capturer une image en stéréoscopie. Une première consiste à aligner deux appareils côte à côte, les deux images sont prises de façon simultanée. Une deuxième solution avec un seul appareil consiste à prendre successivement une première photo pour la vue de gauche puis une deuxième pour celle de droite. Dans le deuxième cas, il est important qu'aucun élément n'ait bougé entre les deux prises (*parallaxe temporelle*). Ces systèmes fonctionnent, mais sont parfois difficiles à maîtriser, l'enregistrement sur deux supports nécessite un montage sur un système externe (PC ou processeur vidéo) pour réaliser une vidéo 3D. L'industrie du cinéma demande à pouvoir réaliser plus de productions en 3D, c'est dans ce contexte que certains constructeurs ont développé de nouvelles caméras 3D FULL HD (cf. figure 2.9). Ces matériels intègrent un double objectif déjà réglé et intégré dans un boitier compact. L'enregistrement du double flux est réalisé directement dans la caméra et il n'est plus nécessaire de passer par un traitement extérieur pour réaliser la vidéo 3D. Le résultat

est de meilleure qualité et le temps de réalisation est considérablement réduit.

FIGURE 2.9: Caméra 3D FULL HD.

Le visionnage des images au format stéréoscopique demande un équipement spécial. Plusieurs constructeurs proposent différents systèmes à base de lunettes à obturation (cf. figure 2.10).

FIGURE 2.10: lunette à obturation nvidia.

Le principe est d'alterner vision gauche et vision droite suffisamment rapidement pour que le cerveau ne perçoive pas la différence. Les cristaux liquides présents dans les

lunettes deviennent opaques lorsque l'image diffusée à l'écran ne lui est pas destinée (cf. figure 2.11). La difficulté réside dans la synchronisation des lunettes avec l'écran. En effet, si ce fonctionnement n'est pas parfait, il y a risque de mauvaise synchronisation entre les deux vues : par exemple si un œil voit une personne avec le bras levé et l'autre œil voit la même personne avec le bras horizontal, le cerveau qui réalise la fusion des deux vues ne peut décider de la position du bras et cette partie de l'image devient confuse et semble clignoter.

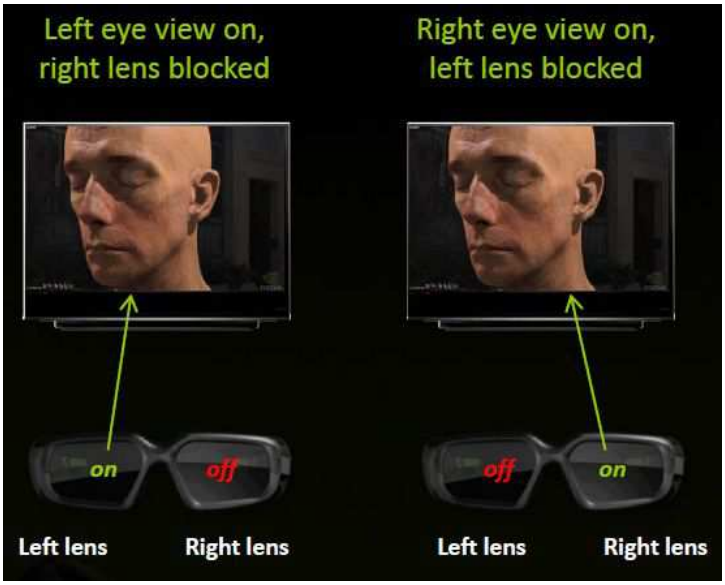

FIGURE 2.11: Fonctionnement de lunettes actives à obturation [20].

L'avantage de cette technologie est que chaque œil voit une image pleine résolution ($1920 \cdot 1080$), en revanche, il est souvent reproché à ce matériel une perte de luminosité qui peut assombrir les images. Cependant, cette technique reste la plus mature actuellement et c'est celle qui sera utilisée dans le cadre de ce projet.

- La 3D par méthode anaglyphe

  Il existe plusieurs méthodes pour créer des images en 3D, mais la première et la plus

connue est la méthode par anaglyphe. Un anaglyphe est une image stéréoscopique composée de deux vues superposées légèrement décalées (cf. figure 2.12). Les deux vues sont de couleurs complémentaires, généralement la vue gauche en rouge et la droite en cyan. La vision en relief est restituée par le port de lunettes bicolores : le filtre rouge pour l'œil gauche et le filtre cyan pour l'œil droit.

Figure 2.12: RHODODENDRON en 3D par anaglyphe.

Le principe de l'anaglyphe réside dans le fait que certaines couleurs sont invisibles à travers un filtre coloré donné : c'est le principe de la *synthèse soustractive*. Le filtre rouge du verre gauche de la lunette rend sombre l'image cyan, si bien que l'œil gauche ne perçoit qu'elle, et rend claire l'image rouge, qui disparait. L'effet est inverse pour lœil droit et le filtre cyan. Le cerveau recompose alors le relief à partir des points communs de ces deux images superposées.

La qualité des filtres est indispensable pour éviter l'apparition des *images fantômes*. Celles-ci apparaissent si, par exemple, le filtre rouge laisse passer un peu du rouge de l'image du fait de sa mauvaise qualité ; nous aurons alors l'impression de voir des éléments en rouge en double et en décalé, les éléments rouge à gauche sur l'image, normalement destinés à la vue par l'œil de droite, parvenant également à l'œil de gauche. Les anaglyphes sont en noir et blanc. On peut cependant réaliser des anaglyphes couleur, mais le rendu des couleurs est souvent décevant. Les couleurs sont soit pâles, soit désaturées, rendant l'ensemble souvent terne : le rouge tend vers le marron, le bleu vers le violet, le jaune vers le vert vif... C'est sans doute sur ce point, celui des couleurs, que l'anaglyphe atteint ses limites [2].

Cette méthode reste cependant la plus simple à mettre en œuvre, car elle ne nécessite que peu d'investissement pour visionner la 3D. En effet une simple paire de lunettes à moins de 10 € suffit à visionner les images. En contrepartie, la qualité des images en 3D, la gène occasionnée par le port des lunettes sont un frein à la généralisation de cette technologie pour le cinéma. Bien que cette technologie soit simple à mettre en œuvre, elle ne sera pas utilisée pour la réalisation durant le projet, car la qualité de rendu est définitivement limité.

- L'autostéréoscopie

Avec l'apparition des écrans Haute Définition (HD) sur le marché du grand public, les constructeurs essayent d'intégrer la 3D sur ces matériels. Cependant, les technologies existantes utilisant des lunettes seraient un frein à son utilisation et donc à sa commercialisation de masse. De nombreux constructeurs comme PHILIPS, HOLOVIZIO, 3DTV SOLUTIONS et ALISCOPY travaillent activement à la réalisation d'écrans ne nécessitant pas d'accessoires supplémentaires pour visualiser la 3D. Le principe est de

ne plus faire porter de lunettes à l'utilisateur, mais plutôt de « faire porter des lunettes à l'écran ».

Les écrans 3D autostéréoscopiques sont équipés d'un composant optique lenticulaire, qui permet à chaque œil de voir une image différente. Il remplace donc les lunettes habituellement requises pour regarder un film en relief. Le spectateur n'a plus besoin d'accessoires encombrants, qui l'isolent de son entourage et perturbent les contacts visuels. Le relief se voit naturellement ! L'impression de relief est obtenue grâce à un réseau de microlentilles très fines (*réseau lenticulaire*) placé à la surface d'un écran plat. Le réseau permet d'envoyer à chaque œil une image différente, le cerveau du spectateur reconstituant alors le relief. Ce système étant sensible à la position de l'utilisateur, il est nécessaire de multiplier les points de vue pour que l'utilisateur puisse se déplacer pour apprécier le relief sous différents angles. Certains systèmes peuvent afficher plus de neuf points de vue différents, ce qui permet le déplacement de l'utilisateur, mais surtout offre la possibilité à plusieurs utilisateurs de regarder le même écran (ce qui est impossible avec seulement deux points de vue).

Par exemple, des écrans 3D Alioscopy de grande taille peuvent être vus confortablement par un public de 20 à 50 personnes, réparties sur une zone de 90° et respectant les distances recommandées par le constructeur (10 m maximum pour un écran 47 pouces). Une étroite zone de transition entre œil, où le relief paraît flou se répète tous les 50 cm environ. Il suffit de se déplacer légèrement de côté pour retrouver un relief naturel [25]. De plus, ces écrans sont aussi compatibles 2D et 3D ce qui leur permet d'afficher tous les programmes existants.

Détaillons à présent le point de vue technologique. Les écrans autostéréoscopiques sont donc des écrans *FULL HD* sur lesquels on rajoute des composants optiques lenticulaires (cf. figure 2.13). Ces lentilles se comportent comme de minuscules loupes qui grossissent un point de vue différent selon l'angle sous lesquels on regarde l'écran. Ces lentilles empêchent chaque œil de voir plus d'une image à la fois. Puisque les yeux ne

regardent pas l'écran depuis le même angle, chaque œil reçoit une image différente qui permet au cerveau de reconstituer l'image en 3D.

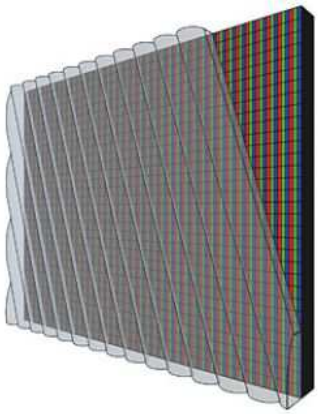

Figure 2.13: Ecran équipé d'un réseau de lentilles [25].

La société ALIOSCOPY détaille sa technologie de la manière suivante. Chaque pixel de l'écran est composé de trois sous pixels de couleurs rouge, vert et bleu. Un écran HD est ainsi composé de près de deux millions de pixels. Le système de lentilles est usiné au 1/100ème de microns puis positionné en diagonale sur l'écran (cf. figure 2.14). Chaque lentille couvre 8 sous-pixels, soit 2 pixels 2/3 (cf. figure 2.15).

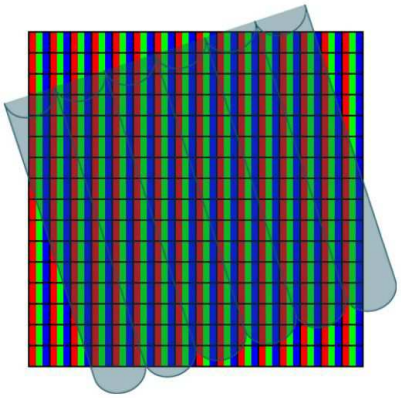

Figure 2.14: Positionnement des lentilles sur un écran ALIOSCOPY [25].

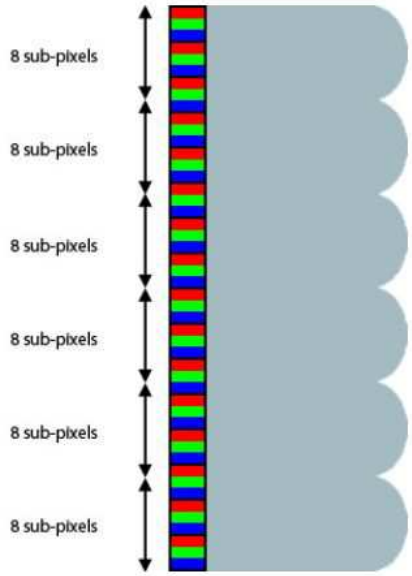

Figure 2.15: Lentilles couvrant 8 sous-pixels [25].

Chaque sous pixel est rattaché à point de vue différent (cf. figure 2.16). La position de l'observateur détermine ce qu'il perçoit à travers les lentilles. En effet, tous les 6.5 cm, l'image grossie par les lentilles est différente. Les yeux étant séparés en moyenne de 6.5 cm voient donc aussi chacun une image différente, ce qui permet au cerveau de reconstituer les images en 3D.

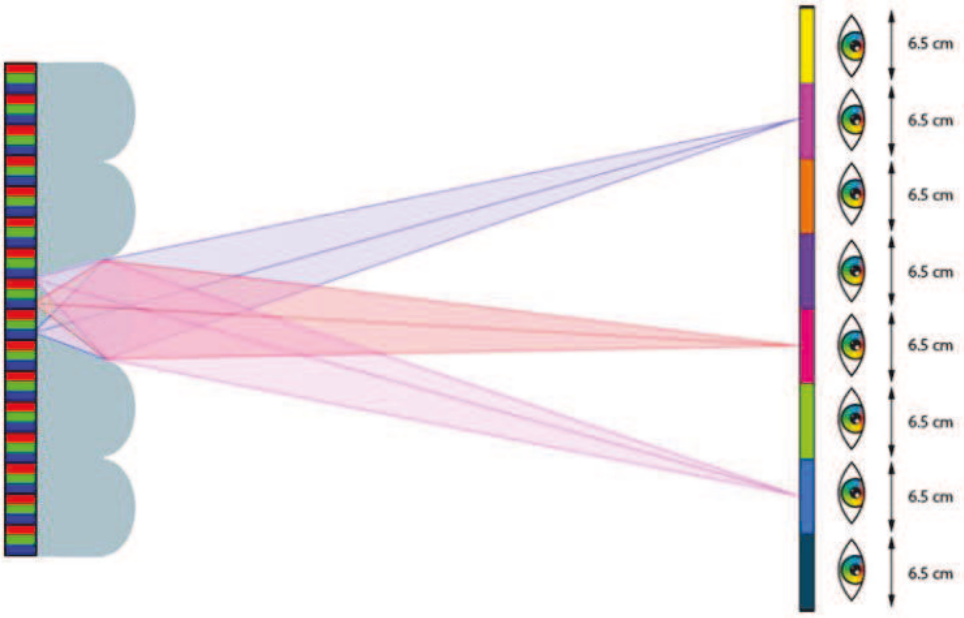

Figure 2.16: Tous les 6.5 cm, l'œil peut voir à travers chaque lentille les parties d'une image différentes [25].

Cette technologie est opérationnelle, mais elle est encore limitée. En effet, même s'il est possible ainsi de regarder la 3D sans lunettes, les images restent voilées et parfois floues. La position de l'utilisateur devant l'écran conditionne la qualité de l'image même s'il est avantageux de pouvoir ainsi se déplacer ou de regarder à plusieurs. L'autre gros problème est la qualité des images qui n'est pas *FULL HD*. Cette baisse de résolution vient du fait qu'un pixel est utilisé pour chaque vue, ce qui a pour effet de réduire la définition de l'image perçue. De plus, cette technologie est encore très onéreuse. Il faut compter un budget de 4 000 € pour un écran 20 pouces et 8 000 € pour un écran 40 pouces. Cependant, les constructeurs ne cessent d'améliorer la technologie et selon le PDG d'ALIOSCOPY, les écrans de ce type toucheront le grand public d'ici cinq à sept ans.

Bien que le laboratoire Images et Vidéocommunications (IVC) soit équipé de matériels

auto-stéréoscopiques (ALIOSCOPY et PHILIPS), cette technologie n'est pas retenue dans le cadre de ce projet. En effet, même s'il est possible d'effectuer de la fusion binoculaire, cette technologie est trop médiocre en termes de qualité d'affichage. De plus, nous avons vu qu'il est nécessaire de respecter une distance et un positionnement pour visualiser les images, ce qui est une contrainte trop forte pour le projet. Cette technologie reste cependant pleine d'avenir, s'affranchir du port de lunettes étant un des challenges pour imposer la 3D à la maison.

Conclusion

Il existe différentes technologies exploitant la fusion binoculaire pour l'affichage d'images en 3D, mais la qualité de restitution pour la vidéo 3D est très inégale. La technologie de l'anaglyphe est souvent utilisée pour créer des images 3D facilement, mais elle ne nous intéresse pas dans le cadre du projet, car elle nécessite le port de lunettes pour une qualité de restitution des couleurs qui est très médiocre. L'auto-stéréoscopie est une voie intéressante, car elle permet de visionner par plusieurs utilisateurs simultanément des vidéos 3D sans lunette, cependant, la résolution des images est réduite par le nombre de points de vue : on perd alors en qualité de restitution pour la vidéo 3D. La stéréoscopie nécessite l'utilisation de lunettes, mais c'est la seule technologie en mesure d'exploiter les images en *FULL HD*, c'est donc la stéréoscopie qui sera retenue pour ce projet.

2.3 Eléments de la colorimétrie

2.3.1 Introduction

Dans ce mémoire nous allons traiter de fusion binoculaire et donc de fusion de couleurs en 3D, pour cela il est nécessaire de définir les éléments de la colorimétrie que nous allons utiliser. Dans un premier temps nous allons décrire rapidement comment sont perçues les images au travers de l'œil puis le cerveau. Cet aspect est important, car nous verrons par exemple que toutes les couleurs ne sont pas perçues par notre système visuel, il est essentiel d'en connaitre les limites pour la *fusion binoculaire*. Nous verrons ensuite les *espaces colorimétriques* utilisés dans ce mémoire. Le choix de ces espaces dépend de l'utilisation que nous en ferons, un modèle standard pour la transmission (YUV), un espace pour l'affichage (RGB) à l'écran et un autre pour les calculs dans un espace perceptuel (LUV). Il existe d'autres *espaces colorimétriques* que nous n'utiliserons pas dans ce projet et qui ne seront donc pas abordés pas dans ce mémoire.

2.3.2 La physiologie sensorielle

La colorimétie permet de qualifier scientifiquement la perception des couleurs, la création de l'image passe par cette perception des couleurs, mais n'est pas la seule composante entre l'image et l'utilisateur. En effet, l'utilisateur est une composante à part entière dans la perception de l'image, la combinaison des aspects physiques, physiologiques et psychologiques sont des maillons importants à prendre en compte. C'est notre cerveau qui en fonction de la réception des informations sous forme de flux lumineux, nous permet de distinguer les nuances entre ces flux pour les interpréter comme des couleurs. Pour mieux comprendre la perception des couleurs, il faut donc d'abord regarder comment fonctionne un œil humain. Pour que l'image soit transmise à l'œil, il est nécessaire d'avoir une source de lumière (écran, néon, etc.). Lorsque cette lumière rencontre un objet, ce dernier en absorbe une partie correspondant à sa couleur et en disperse le reste en une infinité de rayons signalant sa

présence et sa couleur (cf. figure 2.17) [43]. L'œil reçoit les formes projetées sur sa rétine ainsi que les couleurs du spectre émis.

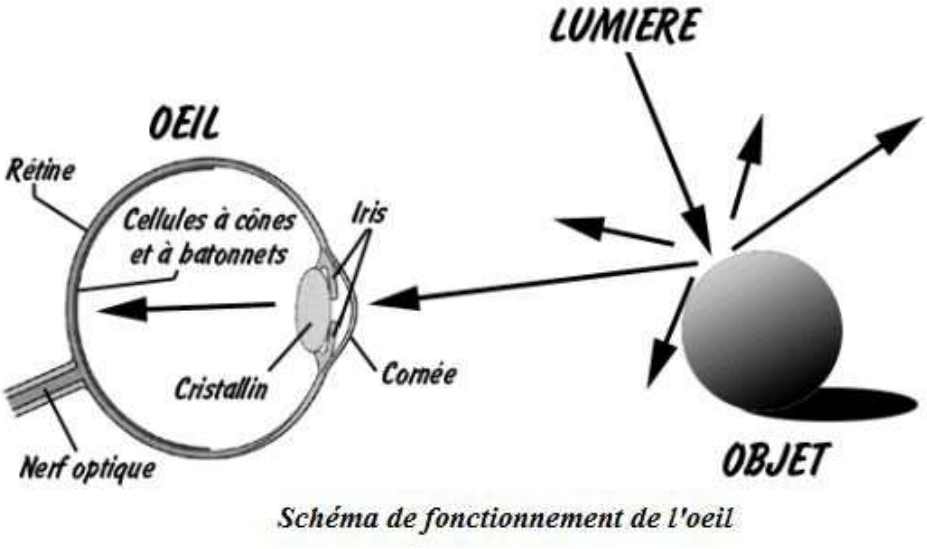

FIGURE 2.17: Fonctionnement de l'œil [43].

L' œil est équipé d'un appareil optique complet. L'*iris* sert de diaphragme, il s'ouvre et se ferme pour accepter plus ou moins de lumière. Le *cristallin* fait la mise au point en fonction de la distance de l'objet. Les flux ainsi reçus par la *rétine* sont traités par des cellules visuelles, qui sont classées en deux catégories :

- les *bâtonnets* qui ont un rôle dans la vision en noir et blanc (vision nocturne ou *scotopique*),
- les *cônes* qui ont un rôle dans la vision des couleurs (vision Diurne ou *photopique*).

La figure 2.18 montre le fond de l'œil et les différentes couches intervenant dans le traitement de la lumière. L'organisation est à la fois radiaire, depuis les photo-récepteurs (à droite) jusqu'aux cellules ganglionnaires (à gauche), mais aussi tangentielle avec de nombreuses interactions entre cellules voisines [47]. Les *cônes* et les *bâtonnets* sont donc répartis à droite sur la rétine.

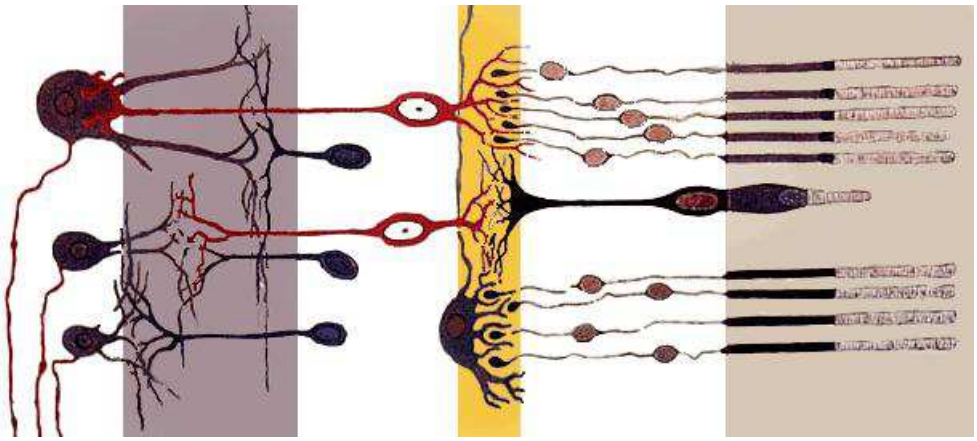

FIGURE 2.18: Organisation axiale simplifiée de la rétine (la lumière arrive par la gauche) [47].

La *rétine* est tapissée de récepteurs photosensibles utilisés en vision diurne par les *cônes* et les *bâtonnets*, mais pas en vision nocturne où seuls restent actifs les *bâtonnets* (il existe néanmoins un mode de vision intermédiaire entre diurne et nocturne, c'est la vision *Mésopique*, dans ce cas, *cônes* et *bâtonnets* sont mobilisés). La perception des couleurs de l'objet est faite par les *cônes* et dépend directement du niveau de l'éclairement.

Le Système Visuel Humain (SVH) est sensible à trois couleurs "pures", car trois types de cônes permettent de les distinguer : cônes L pour la bande rouge, cônes M pour la bande verte, cônes S pour la bande bleue. Les bâtonnets R(od) couvrent surtout le cyan, mais sont utilisés comme capteurs "noir & blanc" (cf. figure 2.19) [40].

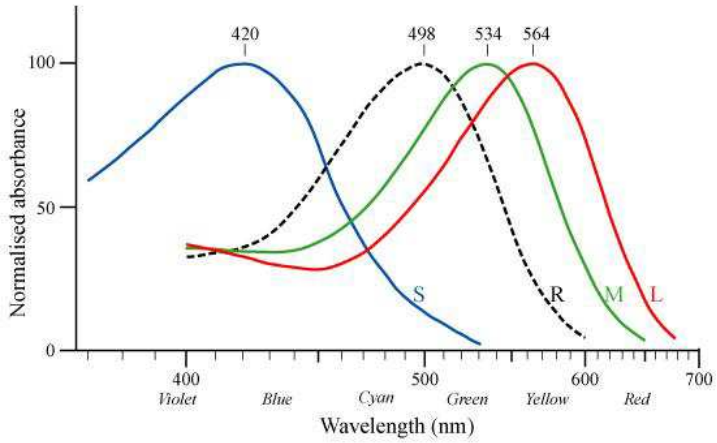

FIGURE 2.19: Courbes d'absorption des cônes et des bâtonnets [40].

Il a donc été montré que l'œil est en mesure d'apprécier que trois couleurs, rouge, vert et bleu, cependant notre cerveau est en mesure de reconstituer beaucoup plus de nuances et il existe donc de nombreuses couleurs qui n'existent pas "monochromatiquement". C'est la combinaison des couleurs, rouge, vert et bleu du flux lumineux reçu par nos yeux, qui nous permet de reconstituer toutes les autres couleurs. Par exemple, la couleur magenta est perçue quand le flux contient du rouge et du bleu. Malgré tous les efforts scientifiques pour décrire la perception des couleurs, elle reste une mesure approximative de notre perception visuelle, qui est influencée par énormément d'éléments physiologiques et psychologiques. Par exemple, il existe des différences au niveau de la perception des couleurs selon les individus, phénomène qui s'amplifie avec l'âge : il est donc possible que deux personnes aient un avis différent sur une couleur perçue.

Comprendre la perception des couleurs est important pour le projet. Nous avons vu que cette perception dépend de la lumière, de l'objet à illuminer et du SVH. Nous avons vu aussi qu'il existe basiquement trois couleurs primaires et de nombreuses nuances qui constituent un ensemble de couleurs perceptibles par notre SVH. Nous allons voir maintenant comment classer et représenter ces couleurs dans des espaces colorimétriques et comment les utiliser dans le cadre de la fusion binoculaire.

2.3.3 Les espaces RVB[1]

Basiquement la représentation d'une couleur peut se faire par un triplet associé à un point dans un espace vectoriel à trois dimensions. Plus précisément, une couleur peut être représentée par un vecteur dont le module correspond au niveau lumineux et l'orientation à la chromaticité (cf. figure 26) [30].

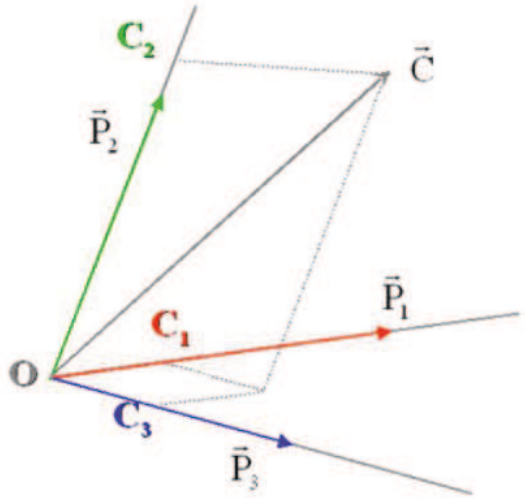

FIGURE 2.20: Espace vectoriel de représentation des couleurs [30].

Dans l'espace vectoriel tridimensionnel de la figure 2.20, une couleur quelconque $\vec{C}$ est définie par un vecteur dont les composantes $C1$, $C2$ et $C3$ sont comptées sur chacun des trois axes d'un système de coordonnées ayant pour origine O. Les vecteurs unitaires $\overrightarrow{P1}$, $\overrightarrow{P2}$ et $\overrightarrow{P3}$ représentent des couleurs primaires. Un vecteur $\vec{C}$ est donc représenté par le triplet ($C1$,$C2$,$C3$), il existe donc différents espaces vectoriels possibles pour différents usages.
Nous avons vu précédemment que les couleurs perçues par l'œil sont le résultat d'une combinaison de trois couleurs caractérisées par des longueurs d'ondes, rouge, vert et bleu (cf. figure 2.19), il existe donc un espace théorique des couleurs perçues. C'est en 1931 que le CIE a tenté de quantifier le premier espace colorimétrique *CIE RVB* qui se rapproche le mieux des combinaisons des trois *couleurs pures* pouvant être affichées sur un écran.

[1] On note ces espaces aussi bien RVB (Rouge,Vert,Bleu) que RGB (Red, Green, blue).

Afin de s'adapter aux systèmes d'affichage, des espaces RGB sont définis. Un espace de couleur *RGB* correspond à la façon dont les couleurs sont généralement codées informatiquement, ou plus exactement à la manière dont les tubes cathodiques des écrans d'ordinateurs représentent les couleurs. Il existe plusieurs systèmes *RGB* qui dépendent du choix de la *teinte*, caractérisée par la longueur d'onde et de la *saturation* (pureté) de chaque composante. La saturation est l'intensité d'une teinte spécifique. Elle est fondée sur la pureté de la couleur ; une teinte hautement saturée a une couleur vive et intense tandis qu'une teinte moins saturée paraît plus fade et grise. Sans aucune saturation, une teinte devient un niveau de gris [52].

Un espace de couleurs est aussi appelé "gamut", qui prend la forme d'un triangle dans les représentations (cf. figure 2.21) et qui correspond au sous-ensemble complet de l'espace de couleurs qu'un type donné de matériel peut reproduire. Ces couleurs sont produites par la synthèse additive de 3 composantes Rouge, Vert et Bleu. Il faut aussi avoir à l'esprit qu'une partie seulement du *gamut* est synthétisable, car les couleurs sont quantifiées (typiquement il y a 256 Rouges possibles, 256 Bleus et 256 Verts soient 256^3 couleurs possibles). Cette notion traduit le fait qu'un appareil, tel qu'un écran d'ordinateur ou une imprimante, ne peut reproduire toutes les couleurs visibles par l'œil humain, elle retient des teintes, mais pas toutes les saturations (cf. figure 2.22).

Le point noté "E" ou "D65" correspond à la notion de "point blanc" qui correspond aux coordonnées chromatiques du point blanc dans le système colorimétrique considéré (cf. figure 2.21) [1].

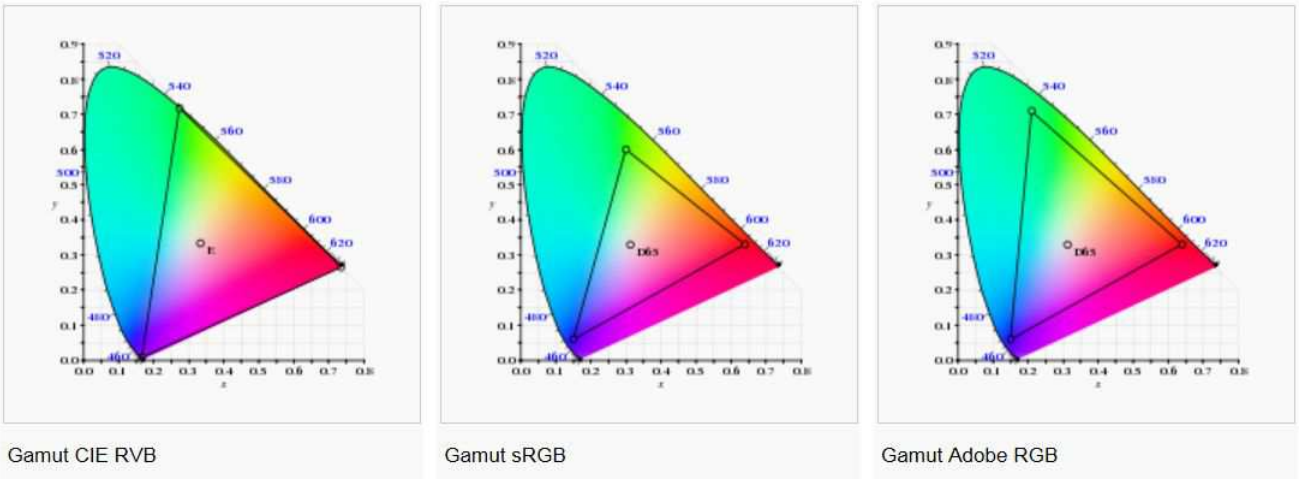

FIGURE 2.21: Différents systèmes *RGB* représentés dans le système CIE xyY[2] [1].

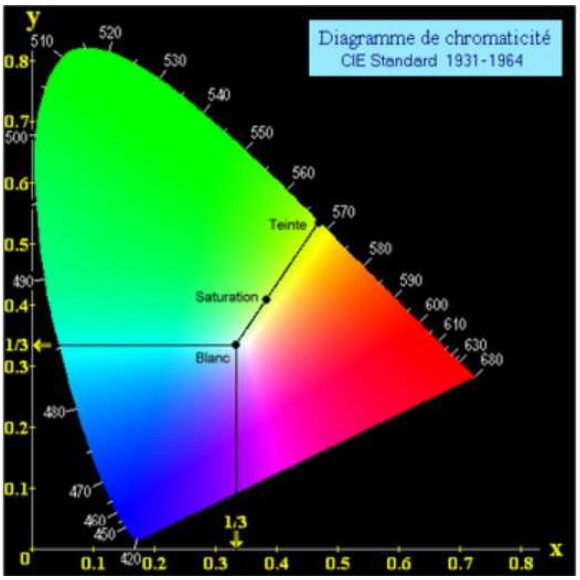

FIGURE 2.22: Diagramme de chromaticité CIE Standard 1931-1964 (CIE xyY) [1].

L'utilisation d'un modèle ou d'un autre dépend de l'utilisation qui en est faite. *sRGB* est un espace qui a été conçu pour être utilisé pour l'affichage, alors qu'un espace *Adobe RGB* est utilisé pour la photographie haut de gamme et l'impression. Pour passer d'un modèle à l'autre, il existe des matrices de transformation. Le système *sRGb* est donc le système le plus compatible avec les écrans mais il réduit le nombre de couleurs possibles, le système *CIE RVB* est alors le meilleur compromis entre la compatibilité d'affichage et la représentation des couleurs [1]. C'est ce système qui sera utilisé pour le projet.
Une autre caractéristique des écrans à prendre en compte, qu'ils soient LCD, plasma ou cathodique, est qu'ils ne sont pas linéaires. Ils déforment la distribution tonale des images selon une courbe appelée "gamma" qui produit une image plus sombre ou plus contrastée que celle attendue (cf. figure 2.23). Il faut donc effectuer un traitement numérique pour adapter les images à notre vision, c'est la correction gamma [28].

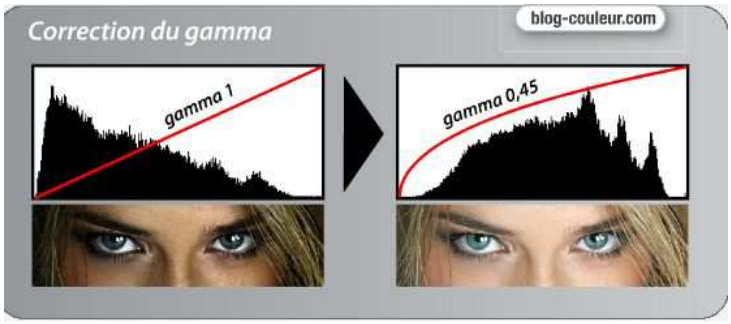

FIGURE 2.23: Correction gamma [28].

Les systèmes RVB présentent l'inconvénient de recourir à des composantes négatives pour caractériser les couleurs très saturées non-inclues dans le "gamut". Ce système de représentation est adapté à l'affichage mais n'est pas un espace perceptuel (non apte à représenter les combinaisons faites par le SVH). Les calculs pour la fusion seront faits dans un autre espace.

[2]L'espace CIE xyY est une version normalisée de l'espace CIE XYZ, nous expliquerons cette notion au paragraphe 2.3.4.

2.3.4 L'espace XYZ

Pour impliquer toutes les couleurs distinguées par le SVH, le CIE a également défini en 1931 le système colorimétrique *CIE XYZ* et a ainsi fondé la colorimétrie scientifique [40]. Une des motivations dans la construction du modèle XYZ (cf. figure 2.24) était de n'utiliser que des *composantes positives* pour toutes les primaires, quitte à définir des primaires virtuelles.

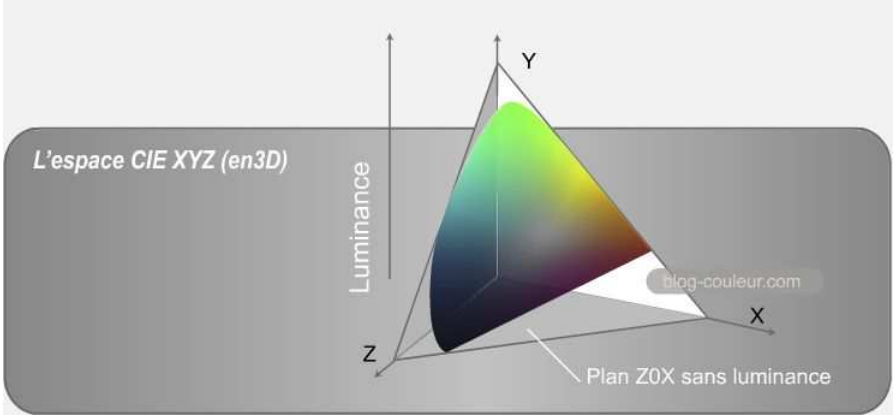

FIGURE 2.24: Diagramme XYZ [28].

Le système *CIE XYZ* introduit la notion de luminance Y. Cet espace est aussi en trois dimensions, formé par la combinaison de trois primaires, trois "lumières" différentes. La luminance est importante, car elle agit sur la sensation visuelle que l'on a de la lumière et donc directement sur la perception de la couleur : elle détermine notre capacité à percevoir correctement les images. En effet, le manque de luminance provoque une pénombre, et au contraire, trop de luminance une perception "blanche" des couleurs.
La luminance est influencée par plusieurs facteurs comme par exemple l'intensité lumineuse de la source de la lumière (cas pour un écran de télévision), elle dépend aussi de la taille de la source lumineuse : **Luminance = intensité lumineuse / surface apparente.**
La luminance est donc un paramètre important dans la perception des couleurs, la figure 2.25 illustre bien la fonction de la luminance dans le spectre des lumières visibles.

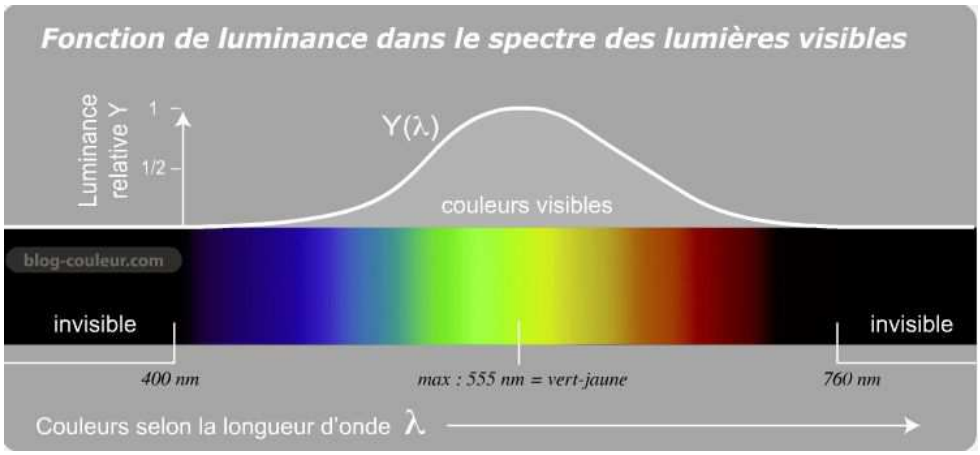

FIGURE 2.25: Fonction luminance dans le spectre des couleurs visibles [28].

Le diagramme de chromaticité *CIE xyY* que nous avons vu dans la section *RGB* (cf. figure 2.22) est en fait une projection en 2D de l'espace XYZ. En se plaçant dans le référentiel XYZ, en utilisant les coordonnées x=X/(X+Y+Z) et y=Y/(X+Y+Z), on obtient la traditionnelle représentation connue sous le nom du "fer à cheval" dans laquelle s'inscrit le *gamut* . Il existe des matrice de passage d'un système à l'autre que nous ne décrirons pas dans ce mémoire.
Le pouvoir de discrimination de l'œil dépend de la teinte, de la saturation et de la luminosité des couleurs observées. Ceci a comme conséquence que suivant les régions du diagramme *CIE xyY*, la sensibilité à différencier des couleurs évolue d'une zone à l'autre. De nombreuses études ont essayé d'apprécier ces variations, comme celle de Mac Adam (cf. figure 2.26) qui a conduit à définir autour de 25 points répartis dans le domaine chromatique, des ellipses. Ces ellipses sont définies telles que, un observateur moyen, ne peut percevoir de différence entre les couleurs définies en son sein. Nous pouvons remarquer que la taille et l'orientation des ellipses changent suivant le point de référence [30].

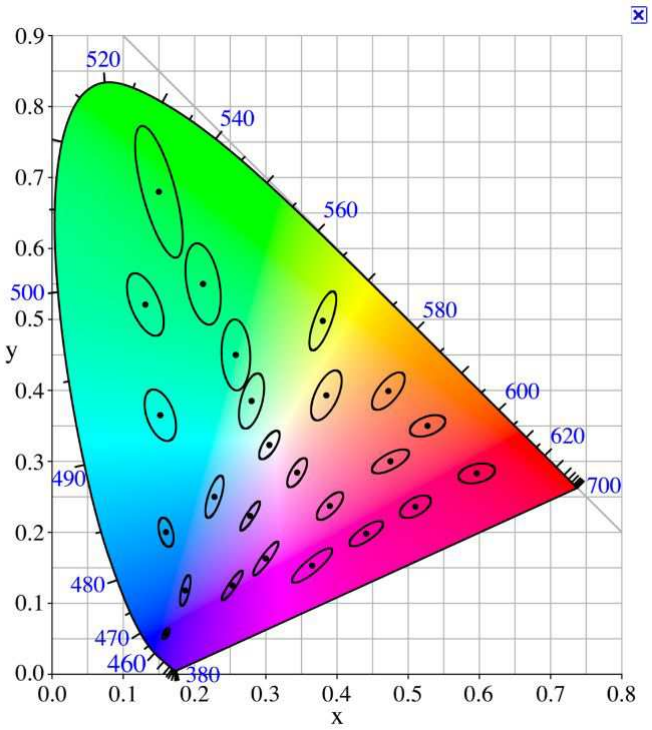

FIGURE 2.26: Ellipses de Mac Adam. Elles sont ici agrandies 10 fois afin de mieux les comparer.

On voit donc que dans l'espace *CIE xyY*, la sensation de différence de couleurs perçues "**n'est pas uniforme**". Ceci est problématique car les distances entre couleurs dans cet espace *CIE xyY* ne représentent pas les distances entre les couleurs perçues, or nous aurons besoin de telles métriques pour le projet.

Nous avons vu dans cette section que l'espace *CIE XYZ* permet de représenter toutes les couleurs perceptibles. Cet espace possède plusieurs avantages comme, des coordonnées positives, des coordonnées indépendantes des primaires *RGB* et la possibilité de comparer les *GAMUT* entre eux. Cependant, cet espace n'est pas uniforme en ce qui concerne la perception des couleurs or pour réaliser des fusions de couleurs dans un espace colorimétrique, nous avons besoin d'espace qui le soit. Il existe plusieurs évolutions au système XYZ, nous allons nous intéresser à l'une d'elle qui est l'espace *LUV*.

2.3.5 L'espace *LUV*

Nous avons vu précédemment quel espace colorimétrique utiliser pour l'affichage de couleurs sur un écran, nous avons vu que cet espace ne représente pas l'ensemble des couleurs perceptibles par le SVH. Nous avons aussi vu que le CIE a défini un nouvel espace *CIE XYZ* qui définit l'ensemble des couleurs perçues par le SVH, mais que cet espace n'est pas uniforme pour la représentation des couleurs. Dans le cadre du projet de mémoire il est important que nous disposions d'un espace de représentation des couleurs permettant de les "manipuler" (par exemple pour les fusionner, ou mesurer des erreurs) en produisant des résultats en lien avec le SVH : c'est l'espace *LUV*.

L'espace *LUV* est un espace colorimétrique défini par la CIE en 1976. Ce système est lui même fondé sur le système *CIE XYZ* (1931), il appartient à la famille des systèmes chromatiques uniformes. Cet espace a été conçu pour qu'une distance entre couleurs dans l'espace représente le même "écart visuel" quelle que soit la région où l'on considère cette distance. On l'utilise par exemple pour le calibrage des moniteurs. Le modèle CIE *LUV* est considéré comme perceptuellement uniforme et est désigné comme un modèle de couleur uniforme.

"Perceptuellement uniforme" signifie que la différence mesurée entre deux couleurs dans l'espace *LUV* est proportionnelle à la différence perçue par un oeil humain. Pour accomplir cette démarche, une échelle colorimétrique uniforme "Uniform Chromaticity Scale (UCS)" a été proposée par la CIE. Elle a été développée pour que la distribution des couleurs soit plus homogène dans le cas de couleurs ayant des clartés très proches. Le diagramme UCS (cf. figure 2.26) utilise une formule mathématique pour transformer les valeurs (X,Y,Z) ou coordonnées (x,y) en un nouvel ensemble de valeurs (u',v') qui présentent l'écart visuel plus précisément .

L'échelle Y est remplacée par une nouvelle échelle appelée L qui est approximativement uniformément espacée.

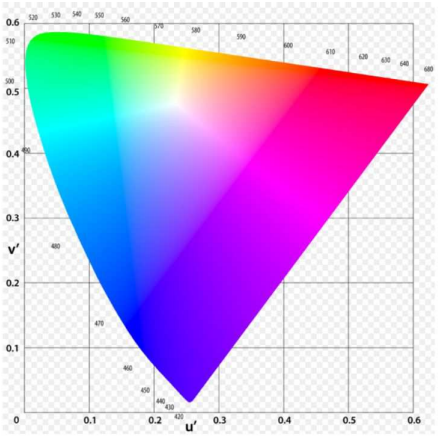

FIGURE 2.27: CIE 1976 UCS.

Les couleurs RVB représentables n'occupent qu'une partie de l'espace couleur LUV, donc il y a beaucoup de combinaisons LUV qui aboutissent à des valeurs RVB invalides (cf. figure 2.28).

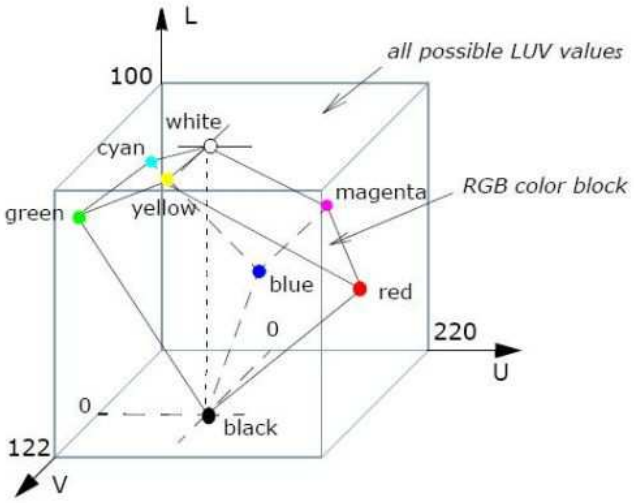

FIGURE 2.28: Cube de couleurs RVB dans le CIE LUV Espace colorimétrique [23].

Il a été montré que l'espace LUV a été conçu de sorte que les différences de couleurs jugées par l'œil humain, soient aussi traduites fidèlement en termes de distance euclidienne, ce qui permet de mesurer les "distances" entre couleurs perçues. Un autre intérêt est que la fusion

binoculaire peut-être traduite dans cet espace comme la somme (vectorielle) de deux points colorés. Cette propriété est intéressante dans le cadre du projet, car la fusion binoculaire peut donc être calculée dans l'espace LUV. Pour l'affichage à l'écran, il faudra une nouvelle transformation du point vers l'espace RGB. Cependant cette démarche ne tient pas compte complètement de l'aspect perceptuel, une couleur fusionnée calculée dans l'espace colorimétrique LUV ne sera pas toujours perçue ainsi par le SVH. Ce point épineux ne sera pas traité en détail dans le cadre de ce mémoire. Après avoir vu les espaces colorimétriques utilisés dans ce mémoire, nous allons détailler le format vidéo YUV qui est utilisé comme source avant d'être transformé dans l'espace RGB et LUV pour traitement.

2.3.6 Le format *YUV*

C'est le modèle de base utilisé pour la transmission de la télévision couleur. En effet, les systèmes de capture vidéo sont équipés de capteurs CCD pour convertir la lumière en composantes RGB pour chaque pixel de l'image. Pour des raisons techniques de limitation de la bande passante et de compatibilité (pour les anciens systèmes) avec les télévisions "noir et blanc", ces signaux de couleurs sont transformés dans un format analogique (anciens systèmes) ou numérique, c'est le format YUV.
Ce format comprend la luminance (Y) et deux différences de couleur (U, V). La luminance peut être calculée comme la somme pondérée des composantes rouge, verte et bleue, les différences de couleur, ou de *chrominances*, sont formées en soustrayant la luminance au bleu et au rouge (cf. figure 2.29) [23].

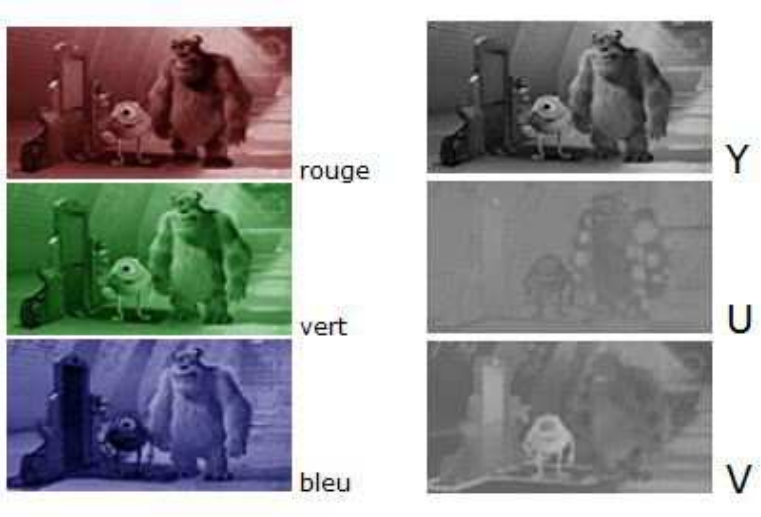

FIGURE 2.29: Image au format *RGB* puis *YUV* [23].

Le principal avantage du modèle *YUV* est le découplage de la luminance et de la couleur de l'information. Il est ainsi possible de traiter la luminance sans affecter sa composante de couleur. Par exemple, l'ajustement du contraste d'une image numérique au format *YUV* peut être effectué simplement par un traitement (égalisation d'histogramme) de sa composante Y [23].

Il y a de nombreuses combinaisons de valeurs *YUV* à partir de plages nominales qui aboutissent à des valeurs RVB invalides, parce que les couleurs RVB possibles n'occupent qu'une partie limitée de l'espace *YUV* (cf. figure 2.30).

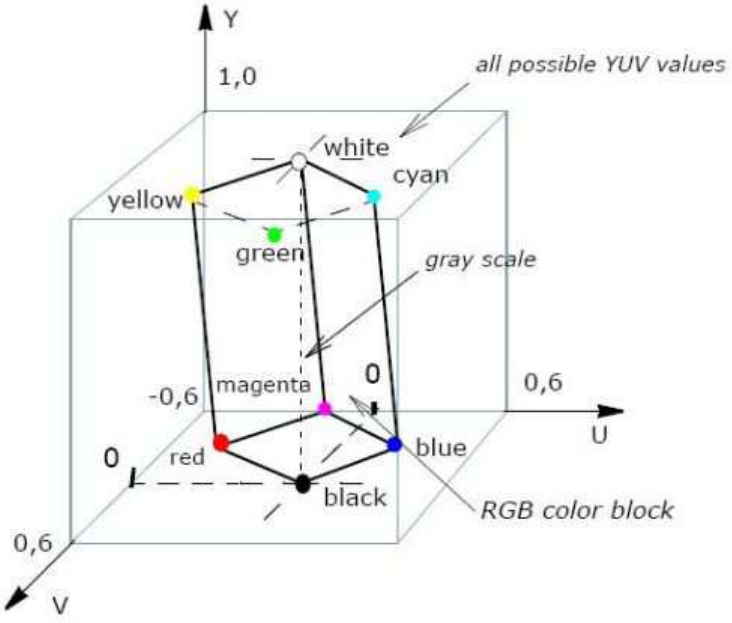

FIGURE 2.30: RVB Cube dans l'espace couleur *YUV* [23].

Un autre avantage du modèle *YUV* est de pouvoir compresser directement l'image par échantillonnage des composantes couleur. Le flux correspondant à une image numérique couleur est constitué de l'ensemble des vecteurs de ses pixels, chaque pixel est caractérisé par 3 composantes, la luminance (Y), la chrominance bleue (U) et la chrominance rouge (V) (cf. figure 2.31).

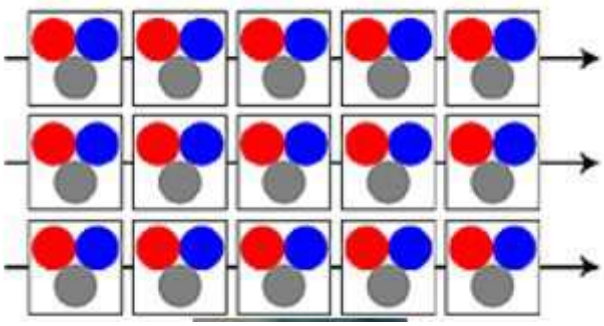

FIGURE 2.31: Format YUV "brut" (4 :4 :4).

Le principe de la compression exploite les caractéristiques de l'œil qui est plus sensible aux variations d'intensité lumineuse qu'aux variations dans l'échelle des teintes (chrominance). La composante Y est alors conservée et les composantes U et V sont échantillonnées. On utilise la notation $Y : U : V$ pour indiquer les proportions respectives de Y, de U et de V. Ainsi un format 4 : 2 : 2 indique que pour 4 Y on n'a que 2 U et 2 V, on a échantillonné les chrominances d'un facteur 2 suivant les colonnes. Les vidéos utilisées dans le cadre du mémoire seront compressées au format YUV 4 : 2 : 0 soit un échantillonnage d'un facteur 2 suivant les lignes et les colonnes de la chrominance.
Le modèle *YUV* est un format standard en vidéo, qui a été développé pour des applications de transmission. Pour l'affichage, le passage vers le système *RVB* est un processus maîtrisé avec des matrices de conversion. Les vidéos *lecteur YUV* sont encapsulées dans un format ("container") standard comme AVI pour être lues par n'importe lequel lecteur.

Conclusion

La colorimétrie est la science de la couleur, elle s'appuie sur des bases physiques qui caractérisent la lumière et sur des bases physiologiques qui caractérisent le SVH. Les couleurs ainsi perçues sont représentées par une combinaison de trois couleurs primaires, le rouge, le bleu et le vert. Le CIE a défini des espaces colorimétriques pour représenter ces couleurs, un pour l'affichage sur des périphériques (RGB), un autre pour représenter toutes les couleurs perçues par le SVH (XYZ). Dans ce dernier, les valeurs de X et Y représentent la composante couleur et le Y représente une autre composante qui est la luminance. L'espace XYZ représente les couleurs sous forme de vecteurs ayant des composantes dans un espace à 3 dimensions, il est donc possible d'utiliser les caractéristiques d'un espace vectoriel pour effectuer des opérations d'addition de couleurs. Cependant, pour que ces opérations soient en phase avec la perception par le SVH, il est nécessaire d'utiliser un espace de couleurs dit "uniforme", c'est le cas de l'espace LUV qui est une évolution du CIE XYZ. La figure 2.32 illustre comment nous allons utiliser ces différents espaces de représentation des couleurs dans ce mémoire.

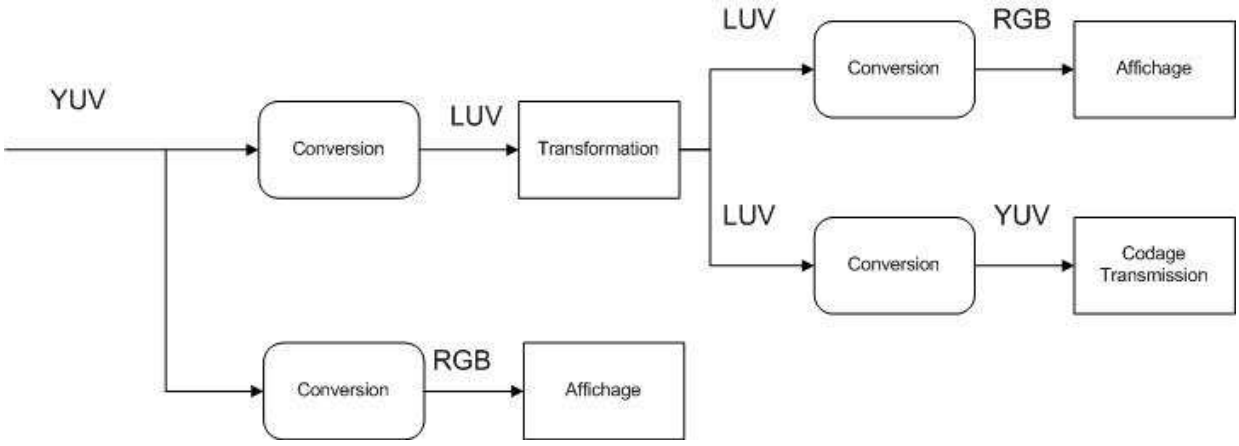

Figure 2.32: Différents espaces pour différents usages.

2.4 La quantification vectorielle

2.4.1 Introduction

Cette partie concerne le schéma de compression de données par quantification vectorielle (QV[3]). En traitement du signal, la quantification consiste à représenter un signal "continu" par un ensemble fini de valeurs discrètes. La QV est une technique souvent utilisée pour la compression de données, l'idée basique est donc de représenter un ensemble de vecteurs source par des représentants. En compression vidéo, la QV est utilisée pour construire des sous-ensembles appelés "listes" ou "dictionnaires". Ce qui nous intéresse dans le contexte de ce mémoire, c'est la capacité de la QV à construire des listes (ou dictionnaires) de vecteurs représentant la source. Nous mettrons à profit cette propriété pour construire des "listes" de couleurs pouvant être fusionnées. Ce chapitre est organisé comme suit : d'abord nous allons définir les outils et notions propres à la QV et ses principes de base. Enfin nous verrons l'algorithme *LBG* qui est un quantificateur optimal basé sur l'algorithme de Lloyd-Max, ainsi qu'une méthode par quantification vectorielle arborescente (QVar) pour tenter d'en améliorer les performances.

2.4.2 Généralités

RICORDEL (1996) [36] a expliqué que : "La quantification consiste en l'approximation d'un signal d'amplitude continue par un signal d'amplitude discrète. La quantification vectorielle consiste alors à représenter tout vecteur $\vec{x}$ de dimension k par un autre vecteur de dimension $\vec{y_i}$ de même dimension, mais ce dernier appartenant à un ensemble fini $\mathcal{D}$ de L vecteurs. Les $\vec{y_i}$ sont appelés vecteurs représentants, les vecteurs de reproduction ou les code vecteurs. $\mathcal{D}$ est appelé dictionnaire ou le "catalogue des formes".

Un Quantificateur Vectoriel de dimension k et de taille L peut-être défini mathématiquement

[3] QV désignera aussi bien la Quantification Vectorielle que le Quantificateur Vectoriel.

comme une application Q de $\mathbb{R}$ vers $\mathcal{D}$:

$$\begin{aligned} Q : \mathbb{R}^k &\longmapsto \mathcal{D} \\ \vec{x} &\qquad Q(x) = \vec{y_i} \end{aligned}$$

avec

$$\mathcal{D} = \{\vec{y_i} \in \mathbb{R}^k / i = 1, 2, ..., L\}$$

Considérons par exemple un ensemble de vecteurs source en dimension 2, chaque vecteur est représenté par un couple (x_1, x_2) (cf. figure 2.33).

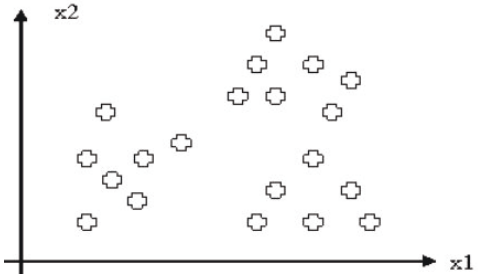

FIGURE 2.33: Ensemble de vecteurs source (en dimension 2) [33].

Dans le plan, si le vecteur $\vec{x}$ a pour coordonnées (x_1, x_2), sa norme s'écrit :

$$\left\| \vec{x} \right\| = \sqrt{{x_1}^2 + {x_2}^2}$$

La QV procède ainsi : considérons un nombre de représentants fixés (éléments rouges sur la figure 2.34), l'espace est divisé en régions avec un représentant pour chaque partition. L'ensemble des représentants est appelé "dictionnaire". Pour quantifier un vecteur (x_1, x_2), on lui attribue les valeurs du représentant le plus proche $\vec{y_i}$.

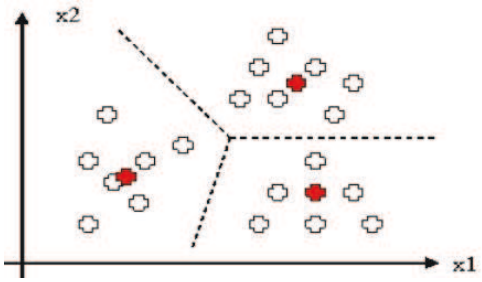

FIGURE 2.34: Partition de l'espace source par QV [33].

On a donc une partition de l'espace source $\mathbb{R}^k$ en L régions C_i. Chaque région est appelée "région de Voronoï" et possède un vecteur représentant, on a :

$$C_i = \{\vec{x} \in \mathbb{R}^k / Q(\vec{x}) = \vec{y_i}\}$$

Le Voronoï réunit donc les points qui sont les plus proches du "vecteur représentant" (cf. figure 2.35).

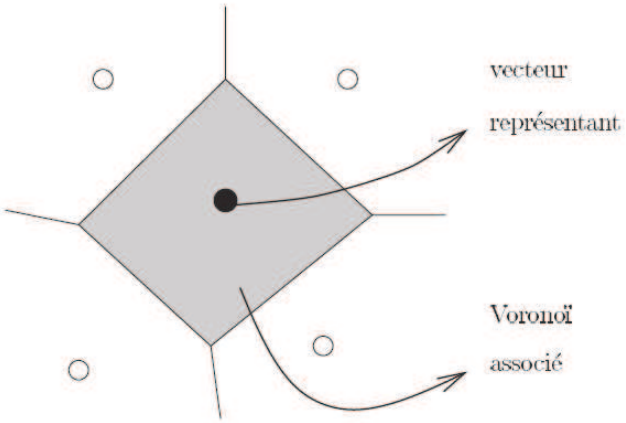

FIGURE 2.35: Principe de la quantification vectorielle [36].

Pour quantifier un groupe de vecteurs source, la QV procède à partir d'un dictionnaire, composé de représentants notés $\vec{y_i}$. On choisit le meilleur représentant de chaque vecteur

$\vec{x}$ à coder au sens d'une certaine distance (la notion de métrique sera expliquée plus loin). Après avoir détaillé les principaux éléments de la QV, nous pouvons regarder comment la QV est appliquée au codage.

2.4.3 Quantification vectorielle et codage

BAS (2005) [3] a expliqué qu'il est possible de diminuer le coût de codage d'une source en regroupant les symboles qu'il faut coder : c'est le principe des méthodes de codages telles que le codage de Huffman, la quantification vectorielle et les codages par blocs en général.
RICORDEL (1996) [36] a expliqué que : "La **compression** ou **codage** de données (ici des images) vise à diminuer la quantité des éléments binaires nécessaire à la représentation de l'information dans le signal à transmettre ou à archiver. Cette diminution peut autoriser ou non la perte d'information. Une bonne compression est réalisée si nous réunissons la loi d'encodage **E** minimisant le nombre d'éléments binaires à transmettre et la loi de décodage **D** assurant une bonne reconstruction du signal source. Lorsque $\mathbf{D}^{-1} = \mathbf{E}$, il s'agit d'un codage sans perte d'information dit **réversible**, celui-ci exploite les redondances du signal, mais ne permet pas des taux de compression élevés. Sinon $\mathbf{D}^{-1} \neq \mathbf{E}$, il s'agit d'un codage avec pertes d'information dit **irréversible**. Cette dernière méthode où le signal décomprimé est dégradé permet des taux de compression importants ; le but est alors que les erreurs engendrées sur les images soient imperceptibles pour l'observateur."
La QV s'inscrit parmi ces méthodes de codage avec perte d'information. Elle conjugue une opération de codage et une opération de décodage (cf. figure 2.36).

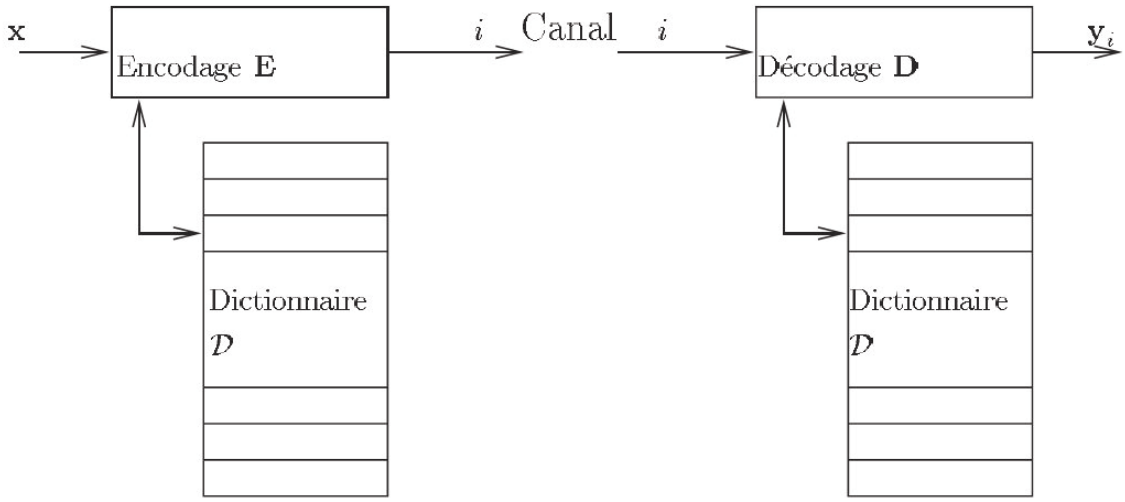

FIGURE 2.36: Schéma général d'un QV.

RICORDEL (1994) [39] définit le codage de la façon suivante : "le rôle du codeur consiste, pour tout vecteur $\vec{x}$ du signal en entrée, à rechercher dans le dictionnaire $\mathcal{D}$ le code vecteur $\vec{y}$ le plus proche du vecteur source x. La notion de proximité est appréciée par la distance euclidienne entre vecteurs. C'est uniquement l'index du code-vecteur $\vec{y}$ ainsi sélectionné qui sera transmis."

Pour l'opération de décodage, le *décodeur* possède une réplique du dictionnaire qu'il balaie pour restituer le code-vecteur correspondant à l'index qu'il reçoit. Le signal ainsi quantifié n'est donc plus identique au signal source, c'est ce procédé qui génère une perte d'information et rend le codage par QV irréversible. Cependant, pour que le signal codé soit de meilleure qualité possible, il est nécessaire que le dictionnaire représente au mieux le signal d'entrée. La constitution du dictionnaire est donc une étape importante, la "qualité" de ses représentants par rapport au signal source doit être la meilleure possible.

Lors de l'encodage d'un vecteur $\vec{x}$, nous allons donc chercher le vecteur $\vec{y_i}$ le plus proche possible au sein du dictionnaire $\mathcal{D}$. Pour cela, nous devons définir la notion de proximité, communément c'est la distance euclidienne qui est utilisée pour mesurer la *distorsion* entre

les vecteurs $\vec{x}$ et $\vec{y_i}$:

$$d(\vec{x}, \vec{y_i}) = (\sum_{n=1}^{k}(x(n) - y_i(n))^2)^{\frac{1}{2}}$$

Un autre paramètre caractérisant le QV est le débit. Classiquement, si L est la taille du dictionnaire, il peut être calculé selon :

$$R = \frac{Log_2(L)}{k}$$

et est exprimé en [bit/dimension]. Nous allons utiliser dans le mémoire, cette définition. RXk représente le coût [en bit] de codage d'un vecteur $\vec{x}$.

L'objectif étant d'obtenir le meilleur compromis entre le coût du codage et l'erreur faite.

Pour mesurer les performances du système, nous allons utiliser l'erreur quadratique moyenne (EQM). Elle permet de mesurer la distorsion moyenne caractérisant les performances globales du quantificateur vectoriel. Elle se calcule de la façon suivante :

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{n=1}^{k}(x(n) - y_i(n))}^2$$

Posons :

$$e(n) = x(n) - y_i(n)$$

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{n=1}^{k} e^2(n)}$$

où $\vec{e} = (e(n))$: est une erreur commise entre $\vec{x}$ et $\vec{y_i}$.

$$EQM = \frac{1}{N}\sum_{i=1}^{N}(e(n)^2)$$

où N est le nombre de vecteurs source considérés. L'EQM est une mesure de l'erreur moyenne, elle est influencée plus par les grandes erreurs que par les petites. Un score de 0 signifiant

une qualité parfaite.

Dans le cas d'images monochromes (pixel codé sur 1 octet), c'est souvent le PSNR[4] qui est utilisé. C'est une mesure de distorsion utilisée en image numérique, tout particulièrement en compression d'images. Il s'agit de mesurer la performance des codeurs en mesurant la qualité de reconstruction de l'image compressée par rapport à l'image originale [51].

Le PSNR pour une image monochrome est défini par :

$$PSNR = 10.log_{10}(\frac{d^2}{EQM})$$

où d est la dynamique du signal (la valeur crête maximale possible pour un pixel). Les valeurs typiques de PSNR pour des images de bonne qualité varient entre 30 et 40 dB.

RICORDEL (1996) [36] a rapporté qu'en théorie les performances de codage sont meilleures si on utilise de vecteurs plutôt que des scalaires. Cependant, la théorie ne décrit pas comment concevoir le codeur "parfait". La quantification optimale a été définie. L'algorithme correspondant procède à partir d'une séquence d'apprentissage, et permet de construire le dictionnaire qui représente au mieux (c.a.d au sens de l'EQM) la séquence d'apprentissage. Cet algorithme est coûteux en termes de calculs, c'est néanmoins celui que nous allons utiliser pour ce mémoire.

2.4.4 L'algorithme Lloyd-Max

Nous avons vu que pour la QV, l'objectif est de calculer un nombre fixé de représentants pour représenter au mieux un signal source. Cet ensemble de représentants est appelé "dictionnaire" et est fixé suivant le débit alloué au quantificateur. Nous avons aussi expliqué que la "qualité" de ce "dictionnaire", c'est-à-dire la qualité de reconstruction du signal source, peut être mesurée par l'EQM. Nous allons voir maintenant comment obtenir ce dictionnaire

[4] Peak Signal to Noise ratio

optimal qui minimise l'EQM à l'aide de l'algorithme de *Lloyd-Max*.
RICORDEL (1996) [36] a expliqué qu'un dictionnaire localement optimal doit réunir les conditions suivantes :

- L'encodeur optimal (pour un dictionnaire fixé) : il respecte la règle du plus proche voisin. Cette règle détermine la partition $\mathbb{R}^k$ en Voronoï ;
- Le décodeur optimal (pour une partition de $\mathbb{R}^k$ donnée) : le vecteur représentant $\vec{y_i}$ minimise la distorsion associée au Voronoï C_i, $\vec{y_i}$ est donc le centroïde de cette région ;
- Il faut que la probabilité d'avoir un vecteur à coder à la même distance de deux représentants soit nulle, sinon ce vecteur source est affecté à l'un des 2 représentants, la partition optimale de l'espace n'est plus et donc la condition de décodeur optimal est devenue impossible. Si les vecteurs source sont à amplitude continue, cette condition est toujours vérifiée.

L'algorithme de *Lloyd-Max* est une méthode qui permet de générer un quantificateur optimal à partir d'une source à coder et d'un dictionnaire initial. Il s'agit d'un algorithme de classification qui fonctionne par itérations (itérations de Lloyd). A chaque itération deux opérations sont réalisées successivement, l'optimisation et la classification (cf. figure 2.37).

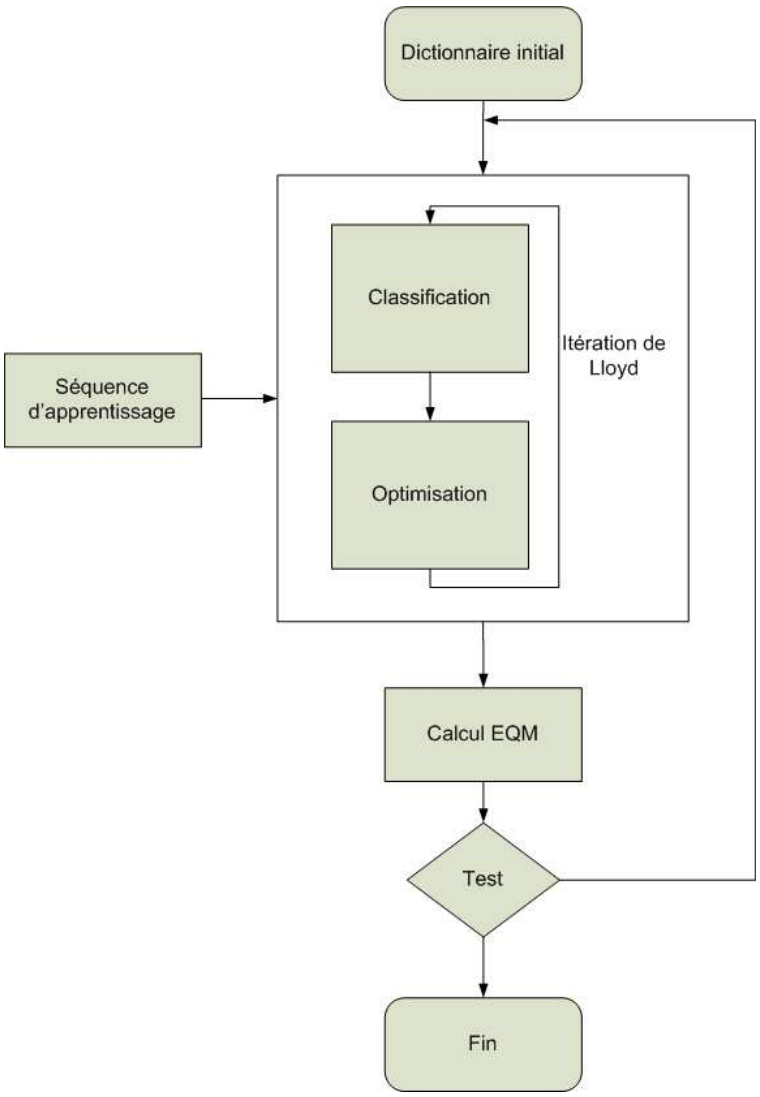

Figure 2.37: Fonctionnement de l'algorithme de Lloyd.

Le module "classification" (pour un dictionnaire fixé)est l'étape d'encodage optimale décrite avant. C'est une étape extrêmement coûteuse en termes de calculs car chaque vecteur $\vec{x}$ de la séquence d'apprentissage doit être comparé (cf. distance euclidienne) avec chacun des vecteurs $\vec{y_i}$ du dictionnaire afin de retenir le représentant le plus proche.
Le module "optimisation" (pour une classification figée) correspond à l'étape de décodage optimale décrite avant.
L'exemple de la figure 2.38 illustre comment fonctionne l'algorithme. Les points représentent la séquence d'apprentissage et les croix représentent les éléments du dictionnaire. Dans ce cas simple, on observe qu'au bout de 4 itérations les croix se trouvent proches des points, l'algorithme se termine avec une EQM qui doit être proche de 0.

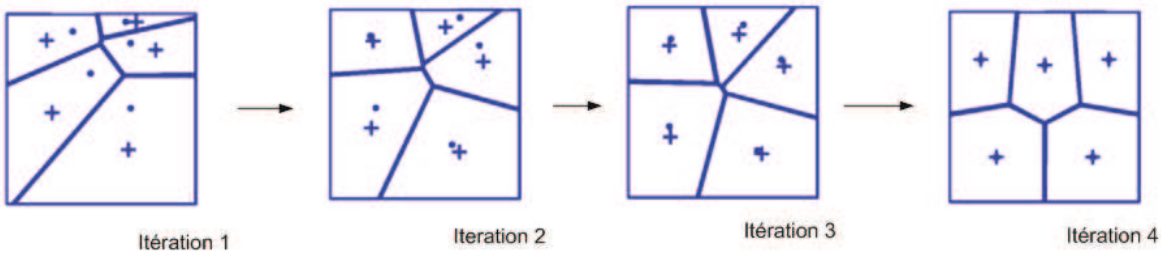

FIGURE 2.38: Exemple de fonctionnement de l'algorithme de *Lloyd-Max*.

La performance de cet algorithme dépend du choix du dictionnaire initial. Il existe plusieurs méthodes pour son choix :

- une initialisation aléatoire (par exemple choix des y_i au hasard parmi les vecteurs de la séquence d'apprentissage) : les résultats observés sont très médiocres ;
- un algorithme à seuil (par exemple choix des $\vec{y_i}$ au sein de la séquence d'apprentissage mais en imposant une distance minimale entre eux). Les résultats observés ne sont pas satisfaisants, car la distance entre les $\vec{y_i}$ est difficile à déterminer et dépend de la séquence d'apprentissage ;
- Une méthode par dichotomie dite méthode de "splitting" qui sera choisie pour le projet et décrite dans la section 2.4.5.

Nous avons vu dans cette partie qu'il existe une méthode pour optimiser un dictionnaire fixé par rapport à une séquence d'apprentissage. Cette méthode utilise des *itérations de Lloyd* pour réduire l'EQM et ainsi optimiser l'emplacement des représentants. Cependant, cet algorithme ne décrit pas la façon de créer le dictionnaire initial. L'algorithme *LBG* est une évolution de l'algorithme de *lloyd Max* et apporte une solution à ce problème.

2.4.5 L'algorithme LBG

Cet algorithme itératif proposé par Linde, Buzo et Gray correspond à une extension de l'algorithme connu sous le nom de *Lloyd-Max* décrit dans la section 2.4.4. Pour un "débit donné", alloué au dictionnaire, il permet de produire le dictionnaire de représentants le plus proche (au sens de l'EQM) d'une séquence d'apprentissage donnée. La séquence d'apprentissage est une série importante de vecteurs source, il s'agit d'une réalisation suffisamment représentative de la source.

L'algorithme LBG combine l'algorithme de *Lloyd* et une technique de *dichotomie vectorielle* appelée "splitting". Le LBG apporte une solution au problème de dictionnaire initial (il n'y en a pas) mais procède en faisant croître progressivement le dictionnaire via notamment la phase de splitting. La phase de "splitting" procède de la façon suivante : elle consiste à découper chaque vecteur représentant $\vec{y_i}$ en 2 nouveaux vecteurs $(\vec{y_i} + \vec{\epsilon})$ et $(\vec{y_i} - \vec{\epsilon})$ ($\vec{\epsilon}$ étant un vecteur de perturbation de faible amplitude), avant d'appliquer au nouveau dictionnaire obtenu les "itérations de Lloyd". Le dictionnaire initial est alors le centroïde de la séquence d'apprentissage, puis l'algorithme génère une succession de dictionnaires (à chaque boucle le nombre de vecteurs de reproduction est multiplié par 2). Le schéma 2.44 représente l'algorithme *LBG* en mettant en évidence la phase d'initialisation et de "splitting" avec l'algorithme de *Lloyd-Max*. Les figures 2.39, 2.40, 2.41, 2.42, 2.43 montrent un exemple d'exécution en 2D de cet algorithme.

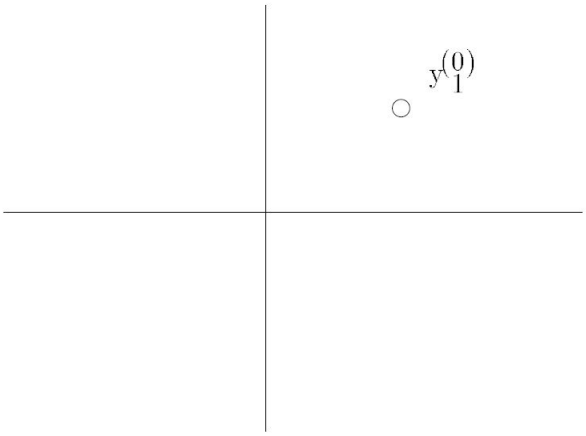

FIGURE 2.39: Initialisation avec le centroïde de la séquence d'apprentissage [39].

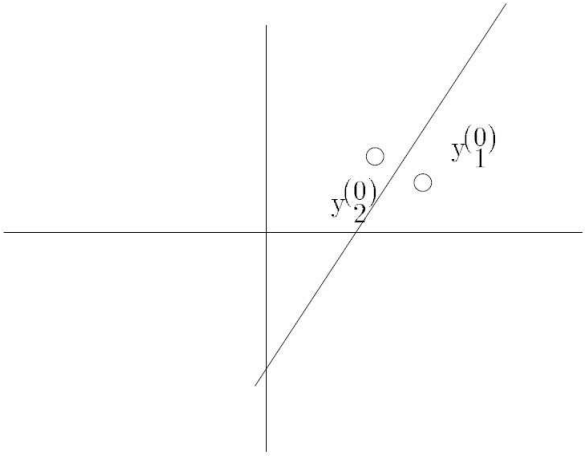

FIGURE 2.40: Splitting 1 [39].

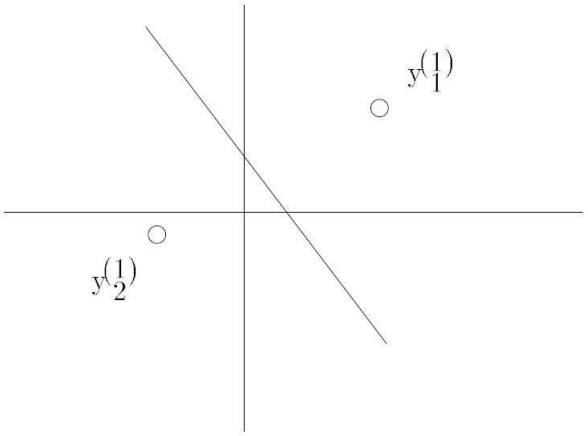

FIGURE 2.41: Optimisation 1 (issue de l'itération de Lloyd) [39].

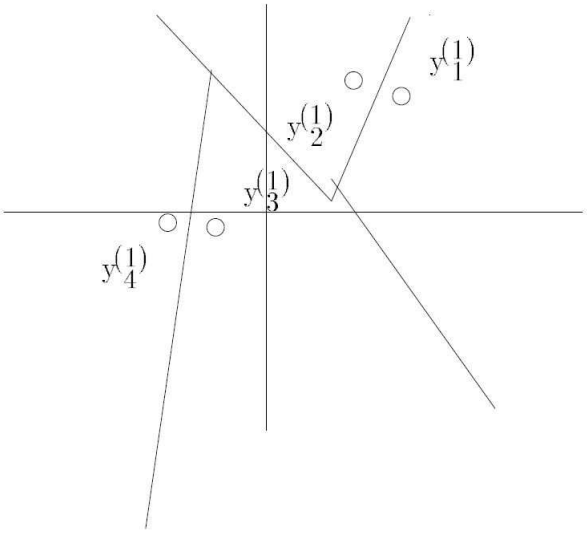

FIGURE 2.42: Splitting 2 [39].

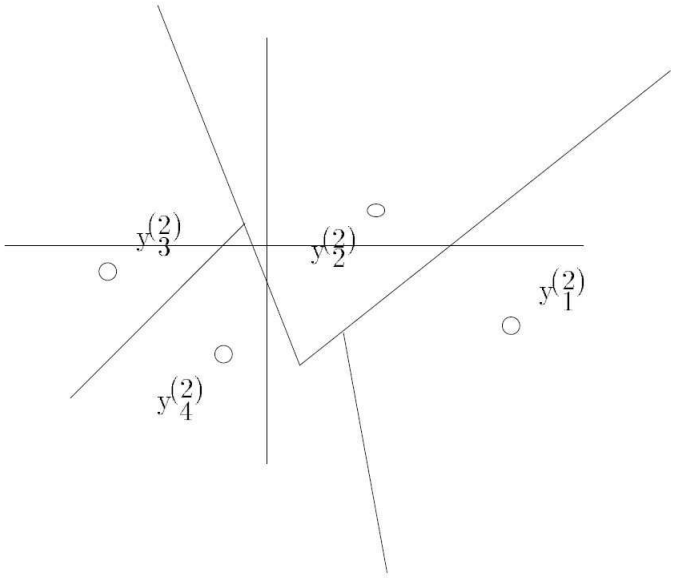

FIGURE 2.43: Optimisation 2 (issue de l'itération de Lloyd) [39].

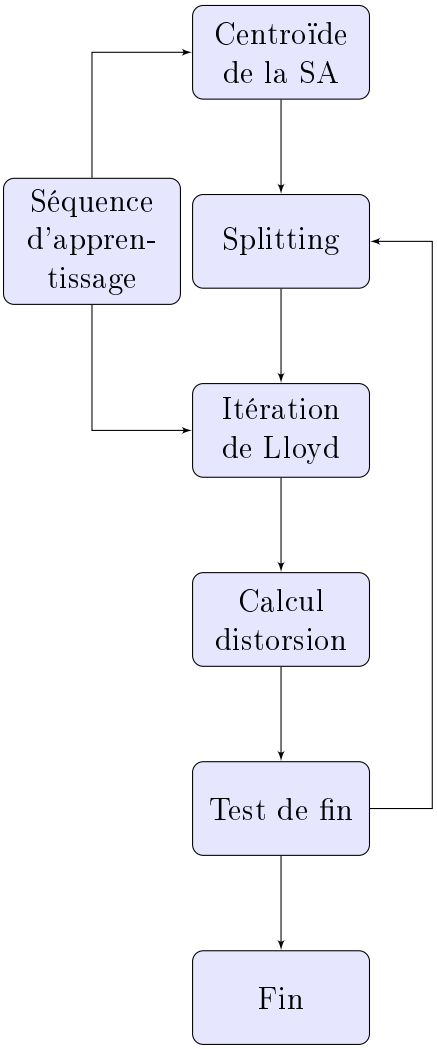

Figure 2.44: Schéma de fonctionnement de l'algorithme *LBG*.

l'algorithme Algorithme de Linde, Buzo, Gray (LBG) peut donc se décrire de la façon suivante :

1. Initialisation : Calcul du barycentre (centroïde) de la séquence d'apprentissage.
2. « Splitting » : Tous les éléments du dictionnaire sont « éclatés » en 2 vecteurs.
3. L'optimisation utilise l'algorithme de *Lloyd-Max* pour améliorer le dictionnaire "splitté".
4. Test de fin, qui peut-être :
   - la taille voulue du dictionnaire est atteinte,
   - le seuil de qualité de reconstruction de la séquence d'apprentissage est atteint.

   Si le test est concluant passe à l'étape 5 sinon passe à l'étape 2.
5. Arrêt

Nous avons vu qu'il existe une méthode efficace qui permet de créer un dictionnaire représentant au mieux une séquence d'apprentissage en fonction d'un seuil d'EQM, c'est l'algorithme *LBG*. Il existe de nombreuses évolutions de cet algorithme qui ne seront pas étudiés dans ce mémoire (évolutions détaillées dans la thèse de RICORDEL (1996) [36]). Nous avons vu une méthode basée sur le "splitting" et qui s'adapte à la séquence d'apprentissage pour la création du dictionnaire. Cependant, cet algorithme a pour inconvénient majeur le coût de calcul(les phases de classification et d'optimisation). Durant le projet de mémoire, nous allons chercher à concevoir une amélioration de l'algorithme *LBG* afin de créer des dictionnaires pour la fusion binoculaire. Un autre avantage de l'algorithme *LBG* est qu'il permet de facilement créer des dictionnaires ayant une structure arborescente, on parle de QVAr (QV arborescente). Nous voulons exploiter cette propriété.

2.4.6 La Quantification Vectorielle Arborescente

La quantification vectorielle arborescente (QVAr) regroupe plusieurs approches de quantification qui utilise un codage à base d'arbres de décision. Cette méthode de quantification offre plusieurs avantages comme la réduction de la charge de calcul, car la recherche au sein de sous-dictionnaires est accélérée (de plus, elle offre une structure de représentation progressive et appropriée pour le codage à débit variable [36]).
RICORDEL (1996) [36] a défini le formalisme relatif aux arbres de la façon suivante :
Dans le cas d'un arbre B-aire (si $B = 2$ l'arbre est binaire), d'un nœud père partent B arêtes vers les nœuds fils. On dit alors que l'arbre est planté s'il possède une racine et est équilibré si ses nœud terminaux sont tous au même niveau(cf. figure 2.45).

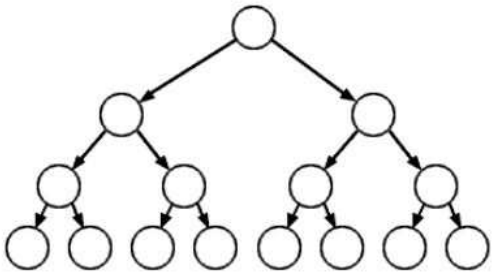

FIGURE 2.45: Arbre binaire équilibré.

Pour la QVAr les représentants sont les L feuilles de l'arbre. Si cet arbre est B-aire et équilibré, sa hauteur est donnée par :

$$H = log_B L$$

Le débit binaire s'exprime en [bit/dimension] :

$$R = \frac{1}{k}.H.log_2 B$$

Deux grandes méthodes de construction de dictionnaires arborescents sont définies :

- L'approche descendante

  Qui intègre la construction de l'arbre à celle du dictionnaire.

  Cette approche peut-être utilisée avec l'algorithme *LBG*. Il suffit de mémoriser les dictionnaires intermédiaires obtenus après "splitting" et optimisation.

- L'approche ascendante

  L'arbre est construit une fois le dictionnaire complet calculé. Une phase de fusions successives des feuilles doit être faite.

Nous nous intéresserons à l'approche descendante pour notre projet de mémoire. En effet, cette méthode permet d'accélérer la phase d'encodage grâce à l'utilisation de l'arbre de décision. Cela accélère aussi la construction du dictionnaire (du module optimisation du LBG).
L'autre type d'arbre est l'arbre non-équilibré. Dans ce cas les feuilles ne se trouvent pas au même niveau (cf. figure 2.46).

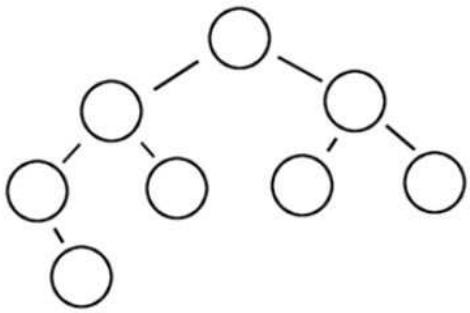

FIGURE 2.46: Arbre binaire non équilibré.

Il existe deux techniques principales pour la construction d'un arbre non-équilibré :

- Technique d'élagage : un arbre complet est d'abord construit, cet arbre est ensuite élagué.

- Technique de découpage : lors de la construction du dictionnaire par une approche descendante, une application non systématique du découpage des nœuds conduit à concevoir un arbre non équilibré. Dans le cas du LBG, cela implique que le "splitting" n'est pas systématiquement appliqué à tous les représentants du dictionnaire.

Dans le cadre de mémoire, nous verrons que nous allons utiliser la QVAr pour construire un arbre non-équilibré avec la technique de découpage pour tenter d'améliorer les performances de l'algorithme.

Conclusion

Ce chapitre montre qu'il existe un outil efficace pour quantifier un signal, c'est la QV. Cette méthode utilise un dictionnaire de représentants pour réduire le nombre de données à représenter. Ce sont ensuite les indices des représentants choisis qui sont transmis et utilisés par le décodeur pour restituer le signal. Cette opération nécessite que le dictionnaire soit envoyé au décodeur pour reconstruire les informations, cette opération implique aussi une perte d'informations irréversible. La qualité du dictionnaire par rapport à la source doit être optimale. Il existe différentes méthodes pour construire ce dictionnaire, nous utiliserons l'algorithme LBG permettant la création d'un dictionnaire à partir d'une séquence d'apprentissage. C'est l'algorithme permettant de construire le dictionnaire optimal (au sens qualité de reconstruction de la séquence d'apprentissage, qualité appréciée via l'EQM). Cependant il est très couteux (construction du dictionnaire, encodage des vecteurs), la QVAr peut améliorer cet aspect. Nous verrons que le LBG et la QVAr seront les outils de base adoptés pour solutionner notre problème de fusion binoculaire. La propriété fondamentale ayant conduite à ces choix étant la capacité de ces algorithmes à produire des représentations compactes et de qualité de l'information (dans notre cas, les couleurs).

Chapitre 3

Fusion d'images stéréo et codage : étude d'une solution à base de deux dictionnaires

3.1 Objectif

L'objectif du projet est d'optimiser la compression des vidéos stéréoscopiques. L'une des pistes étudiées par le laboratoire IVC est d'exploiter les propriétés de fusion binoculaire qui réalise la perception de scènes couleur 3D à partir de l'affichage de 2 images plan codées. Classiquement 2 points appareillés issus d'un même objet de la scène, sont affichés pareillement (même couleur) sur les 2 images gauche et droite. Cette couleur est donc aussi celle du point de l'objet perçu dans l'espace 3D. Basiquement, si on dispose d'un catalogue de $N1$ couleurs affichables pour tous les pixels de l'image gauche, et de $N2$ couleurs pour ceux de l'image droite, en théorie il y a une combinatoire de $N1.N2$ couleurs possibles par les points fusionnés et perçus en 3D. Si les points fusionnés doivent être de la même couleur dans les images gauche et droite, cela donne un catalogue de $N1$ couleurs pour les pixels de l'image gauche et de $N1$ (mêmes) couleurs pour l'image droite, et finalement un potentiel que de N1 couleurs perçus en 3D. Le projet vise à tirer profit du fait que le point perçu en 3D est donc la fusion de deux points affichés en 2D. En jouant sur la combinaison des couleurs 2D (en faisant cette fois la fusion de deux couleurs différentes), il doit être possible d'accroître le nombre de couleurs affichées en 3D (cf. figure 3.1).

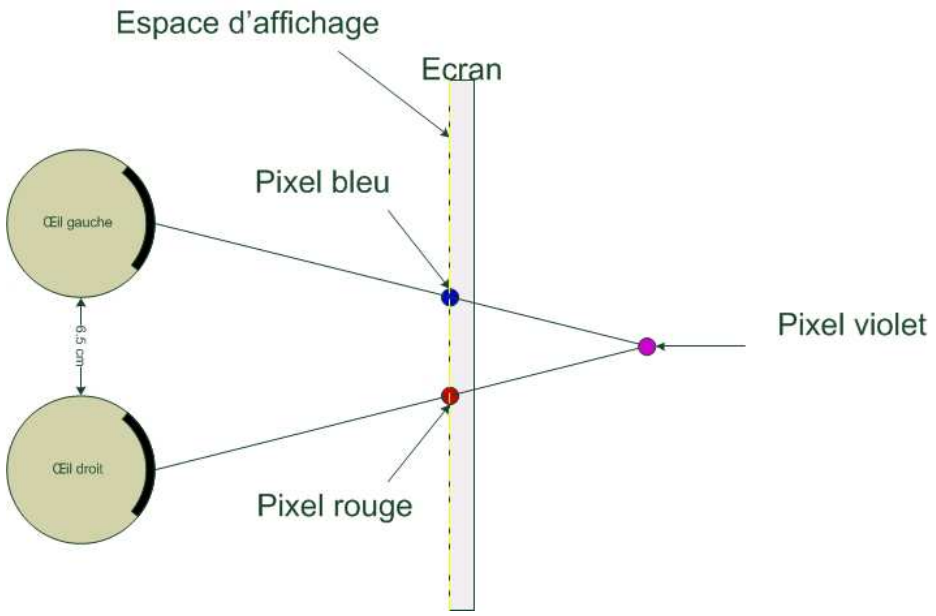

FIGURE 3.1: Principe de la fusion binoculaire de deux points colorés différemment.

L'idée est donc de constituer un dictionnaire (ou catalogue) de couleurs pour l'œil droit et un autre pour l'œil gauche de telle sorte que la combinaison de ces dictionnaires produise un dictionnaire encore plus large de couleurs perceptibles pour les objets de la scène 3D.
Nous avons vu dans l'étape de bibliographie qu'il était possible de combiner les couleurs dans l'espace LUV (il y a aussi de nombreuses autres règles liées à la perception).
Nous avons également vu qu'il était possible de construire avec l'algorithme *LBG* un dictionnaire de représentants à partir d'une séquence d'apprentissage. C'est notre but, construire ainsi des dictionnaires de couleurs, mais il va falloir adapter les algorithmes à notre contexte particulier. En effet, nous ne disposons pas à l'initialisation de deux séquences d'apprentissage chacune caractéristique respectivement des images gauche et des images droite de la séquence stéréo, et tenant compte des contraintes de la fusion binoculaire. Il va falloir adapter le LBG de façon à effectuer (itérativement) :

- cette séparation en deux séquences d'apprentissage complémentaires ;
- puis la production de deux dictionnaires de couleurs "fusionnables", l'un pour l'image

gauche, l'autre pour l'image droite.

Nous verrons aussi que ce schéma de codage très particulier aura d'autres implications, par exemple :

- le procédé de codage des images stéréo ;
- la méthode de calcul des erreurs de codage faites.

3.2 Spécifications générales

Pour ce projet de mémoire, il est demandé de concevoir un programme permettant de coder une séquence vidéo stéréoscopique présentée dans un format YUV, puis de la décoder dans un format AVI, et de pouvoir la visionner en 3D stéréoscopique à l'aide du kit 3D Nvidia. La vidéo ainsi produite doit contenir un flux vidéo stéréo, et le résultat final doit être une vidéo affichée en 3D. Le processus de transmission entre le codeur et le décodeur ne sera pas étudié durant ce mémoire, nous allons donc considérer que les informations transmises au décodeur sont identiques aux données en sortie du codeur (cas d'une transmission parfaite).

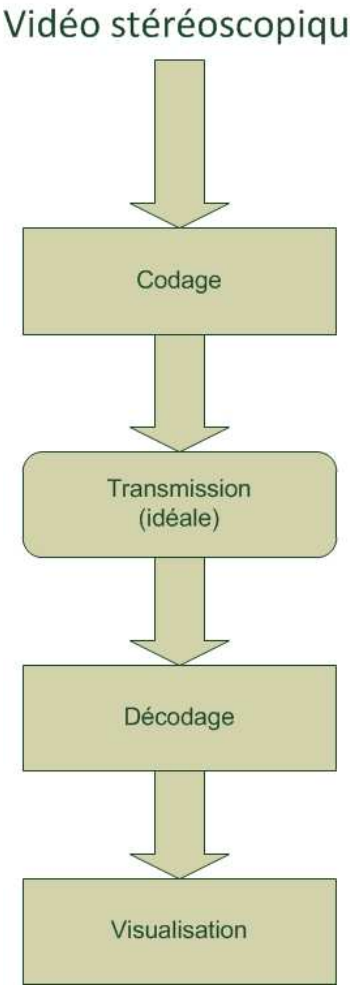

FIGURE 3.2: Spécification de l'algorithme général.

Ce programme sera appelé "codec de fusion binoculaire", il doit pouvoir fonctionner selon deux modes :

- Avec des données synthétiques pour le prototypage de l'algorithme. La visualisation sera faite en 2D car il est nécessaire d'implémenter des étapes supplémentaires pour la visualisation en 3D (cf. figure 3.3) ;
- Avec tout ou partie d'une séquence vidéo stéréoscopique (cf. figure 3.4).

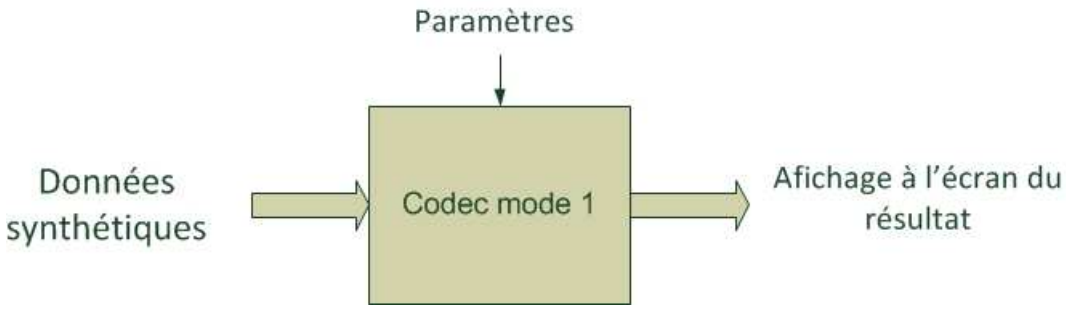

FIGURE 3.3: Codec "mode 1" (codage de données synthétiques).

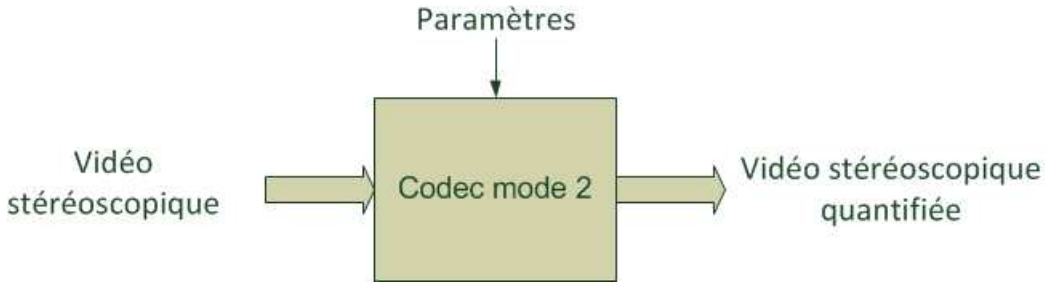

FIGURE 3.4: Codec "mode 2" (codage d'une vidéo stéréoscopique).

La figure 3.5 montre de façon simplifiée les grandes étapes des codecs. Il existe une partie commune en bleu dans la figure 3.5, qui concerne la création d'un dictionnaire que nous appellerons dictionnaire "composé". Dans la section suivante, nous allons détailler les étapes pour la création d'un tel dictionnaire.

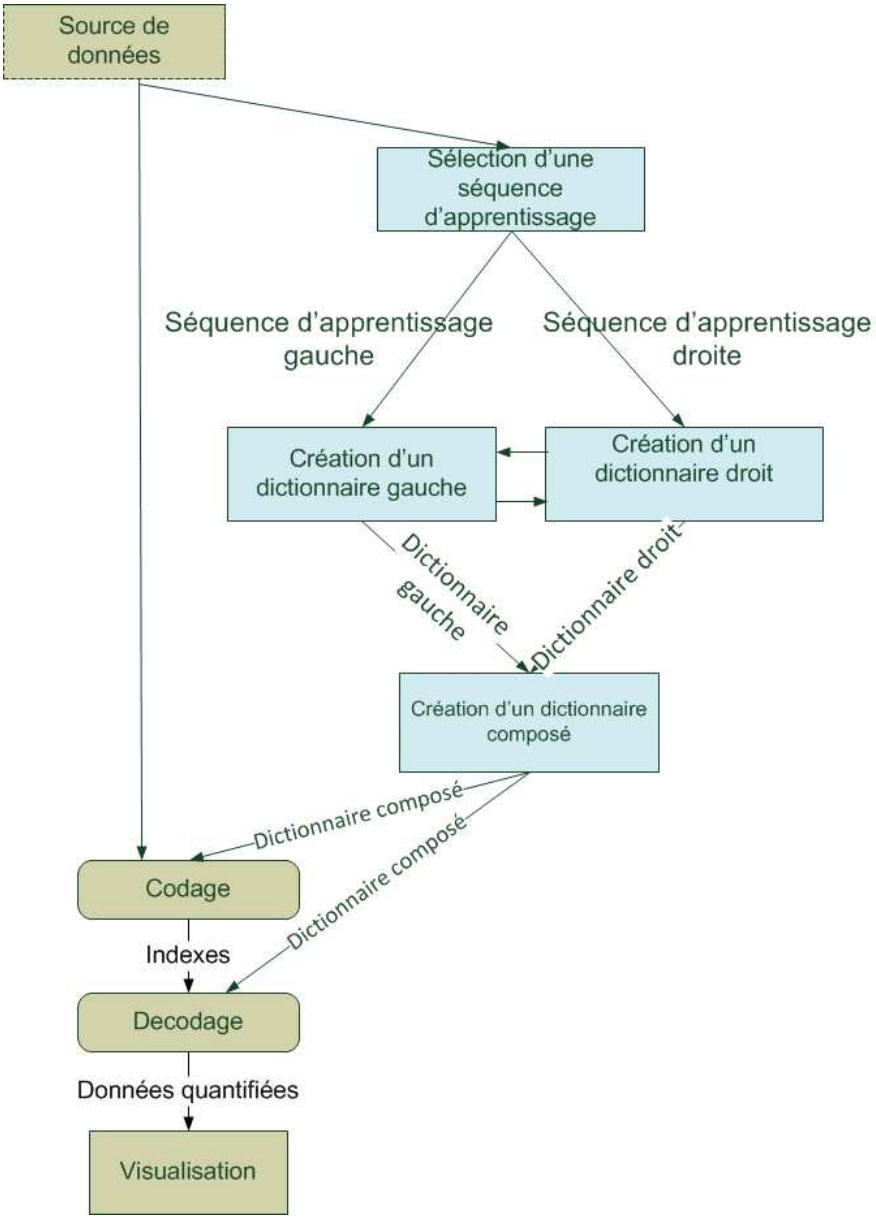

FIGURE 3.5: Principales étapes du codec.

3.3 Codec "mode 1" avec données synthétiques

Ce codec a pour objectif de manipuler des données synthétiques de façon à contrôler le bon déroulement de l'algorithme. Grâce à ce codec, nous pouvons effectuer des scénarios simples et analyser les résultats afin de s'assurer de son bon fonctionnement. Cette vérification ne serait pas possible sur une vidéo "naturelle", car le nombre de vecteurs générés est trop grand, ce qui rendrait la conception et la vérification des étapes difficiles.

3.3.1 Algorithme pour la création d'un dictionnaire composé

La figure 3.5 nous montre que la création d'un dictionnaire composé est en fait le résultat de la création de 2 dictionnaires (un gauche, et un droit) qui sont combinés. Nous disposons à l'initialisation d'une séquence d'apprentissage unique qui est ici une séquence synthétique. Il faut donc adapter l'algorithme *LBG* classique de façon à produire : 2 séquences d'apprentissage complémentaires (une gauche et une droite), et leurs dictionnaires respectifs, dictionnaires qui doivent pouvoir se combiner. L'idée retenue pour constituer ces séquences est d'utiliser un dictionnaire (par exemple le gauche) précédemment créé pour générer une nouvelle séquence (par exemple la droite), cette nouvelle séquence servira pour la création d'un nouveau dictionnaire (ici le droit) et ainsi de suite. Plus précisément, nous allons utiliser le dictionnaire droit pour créer la séquence gauche, et donc actualiser le dictionnaire gauche. Pour ce faire, nous allons utiliser une méthode par projection de la séquence "au travers" un dictionnaire. Cette opération sera réitérée autant de fois que nécessaire en utilisant le dictionnaire actualisé lors de la précédente itération.
Pour mieux identifier ces étapes, nous devons créer deux nouvelles opérations pour générer ces nouvelles séquences. L'étape de création de la séquence droite sera nommée Cal_Sd et l'étape de création de la séquence gauche sera nommée Cal_Sg.

A partir d'une nouvelle séquence, il est possible de créer un nouveau dictionnaire, pour cela

nous allons utiliser une version adaptée de l'algorithme *LBG* que nous appellerons *LBG stéréo*. Nous détaillerons dans la section 3.3.2 les adaptations apportées à cet algorithme.
Une fois les dictionnaires droit et gauche créés, nous pouvons générer le nouveau dictionnaire composé nommé Yc, cette étape sera nommée $Cal\_\ Yc$. Si ce dictionnaire vérifie les conditions de sortie, les dictionnaires droit et gauche sont alors considérés comme utilisables pour quantifier la source.
Le schéma 3.6 décrit plus en détail les étapes réalisées pour la création de ce dictionnaire Yc. Pour identifier les différents éléments, nous utiliserons la convention suivante :

- Sa : séquence d'apprentissage (la séquence synthétique ou la séquence stéréo initiale) ;
- Sd : séquence droite ;
- Sg : séquence gauche ;
- Yd : dictionnaire droit ;
- Yg : dictionnaire gauche ;
- Yc : dictionnaire composé ;
- $Cal\_\ Sd$: calcul de la séquence droite $Sa\_\ d$;
- $Cal\_\ Sg$: calcul de la séquence gauche $Sa\_\ g$;
- $Cal\_\ Yc$: calcul du dictionnaire composé Yc.

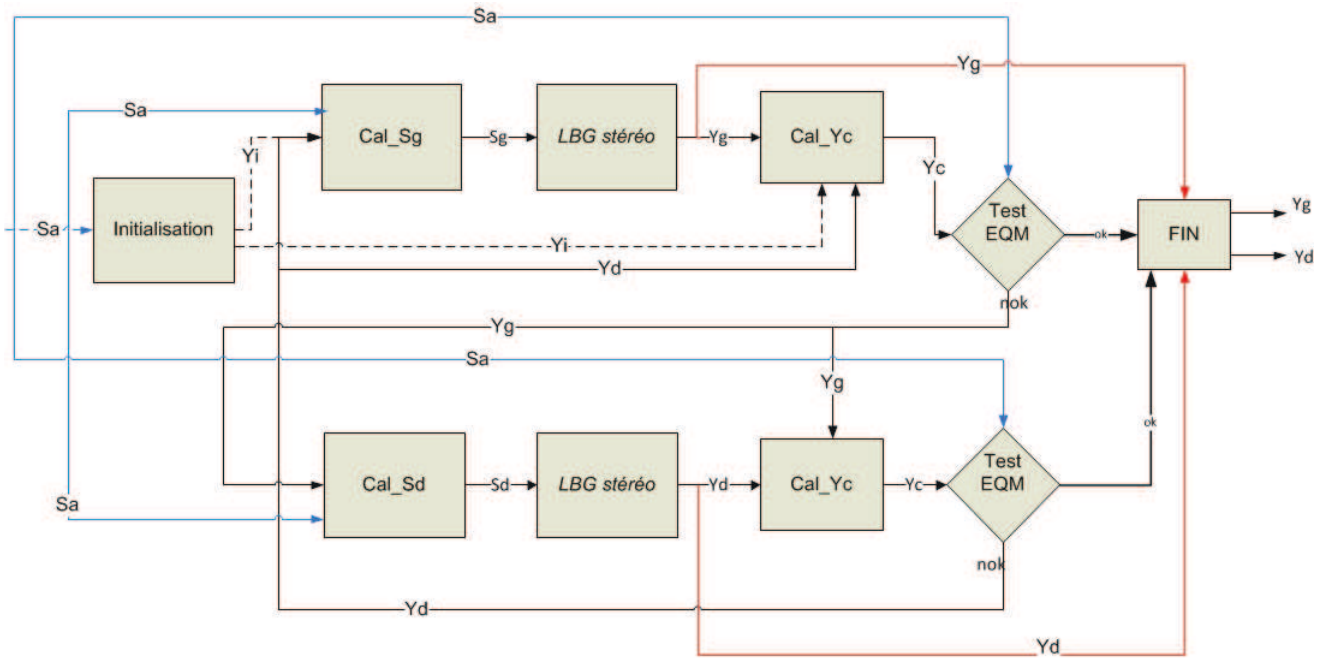

FIGURE 3.6: Algorithme de création du dictionnaire composé Yc.

L'étape "d'initialisation" est particulièrement importante. En effet, au début de l'algorithme, les dictionnaires droit Yd et gauche Yg sont vides. Or, nous avons vu précédemment qu'il est nécessaire d'avoir un dictionnaire pour créer une séquence droite ou gauche, il faut donc fixer le premier dictionnaire de façon arbitraire. Le choix a été d'utiliser la séquence d'apprentissage Sa pour la création du premier dictionnaire Yi (qui se substitue au premier dictionnaire droit Yd), le nouveau dictionnaire sera alors composé d'un seul vecteur qui sera le barycentre de la séquence d'apprentissage.

La fusion est une étape qui consiste à construire le dictionnaire composé Yc à l'aide des dictionnaires droit Yd et gauche Yg disponibles, le calcul est effectué dans l'espace LUV (cf chapitre 2.3.5) :

$$\vec{y_c} = \frac{\vec{y_d} + \vec{y_g}}{2} \tag{3.1}$$

où :

- $\vec{y_c}$: vecteur représentant du dictionnaire Y_c ;
- $\vec{y_d}$: vecteur représentant du dictionnaire Y_d ;
- $\vec{y_g}$: vecteur représentant du dictionnaire Y_g ;

Nous pouvons maintenant détailler les étapes *Cal_Sd* et *Cal_Sg* qui fonctionnent de façon symétrique. Prenons l'exemple de *Cal_Sd*, à l'aide du dictionnaire *Yg* créé lors de la boucle précédente, nous construirons la nouvelle séquence *Sd* par projection de la séquence d'apprentissage *Sa* au travers *Yg*. Ce processus se déroule en 2 étapes, la classification et la projection (cf. figure 3.7). Pour projeter les vecteurs, nous effectuons une classification de tous les vecteurs de la séquence d'apprentissage *Sa* avec le dictionnaire (ici *Yg*). L'étape suivante consiste à projeter les points au sein des Voronoï "au travers" les vecteurs représentants de ces mêmes cellules. Cette projection créée de nouveaux vecteurs par symétrie, ces points formeront la nouvelle séquence *Sd*. Mathématiquement, cette opération de projection inverse l'équation de la fusion (cf. équation 3.1), avec :

$$\vec{x_d} = 2.\vec{x} - \vec{y_g}$$

où :

- $\vec{x_d}$: vecteur de la nouvelle séquence d'apprentissage S_d ;
- $\vec{x}$: vecteur de la séquence d'apprentissage S_a ;
- $\vec{y_g}$: vecteur représentant du dictionnaire Y_g ;

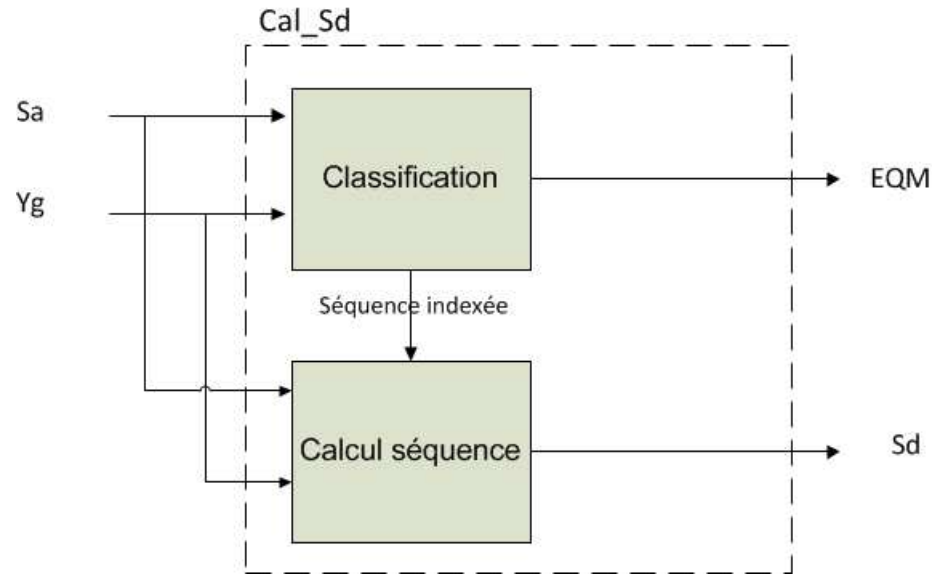

FIGURE 3.7: Détail de l'algorithme de création des dictionnaires (ici *Cal_Sd*).

l'étape *test EQM* de la figure 3.6 se calcule en utilisant la définition de l'EQM du paragraphe 2.4.3 qui requiert que l'on dispose des deux dictionnaires Yd et Yg (l'un a été nouvellement actualisé), on peut donc calculer le nouveau dictionnaire composé Yc. On calcule donc *EQM(Yc,Sa)* qui permet de savoir si Yc représente suffisamment bien ou non la séquence d'apprentissage Sa.

Après avoir défini les différentes étapes, nous pouvons présenter l'algorithme complet à la figure 3.8 en précisant tous les flux d'entrée et sortie de chaque processus. Il nous reste à détailler l'algorithme *LBG stéréo* qui est une évolution de l'algorithme *LBG* que nous avons vu au paragraphe 2.4.5.

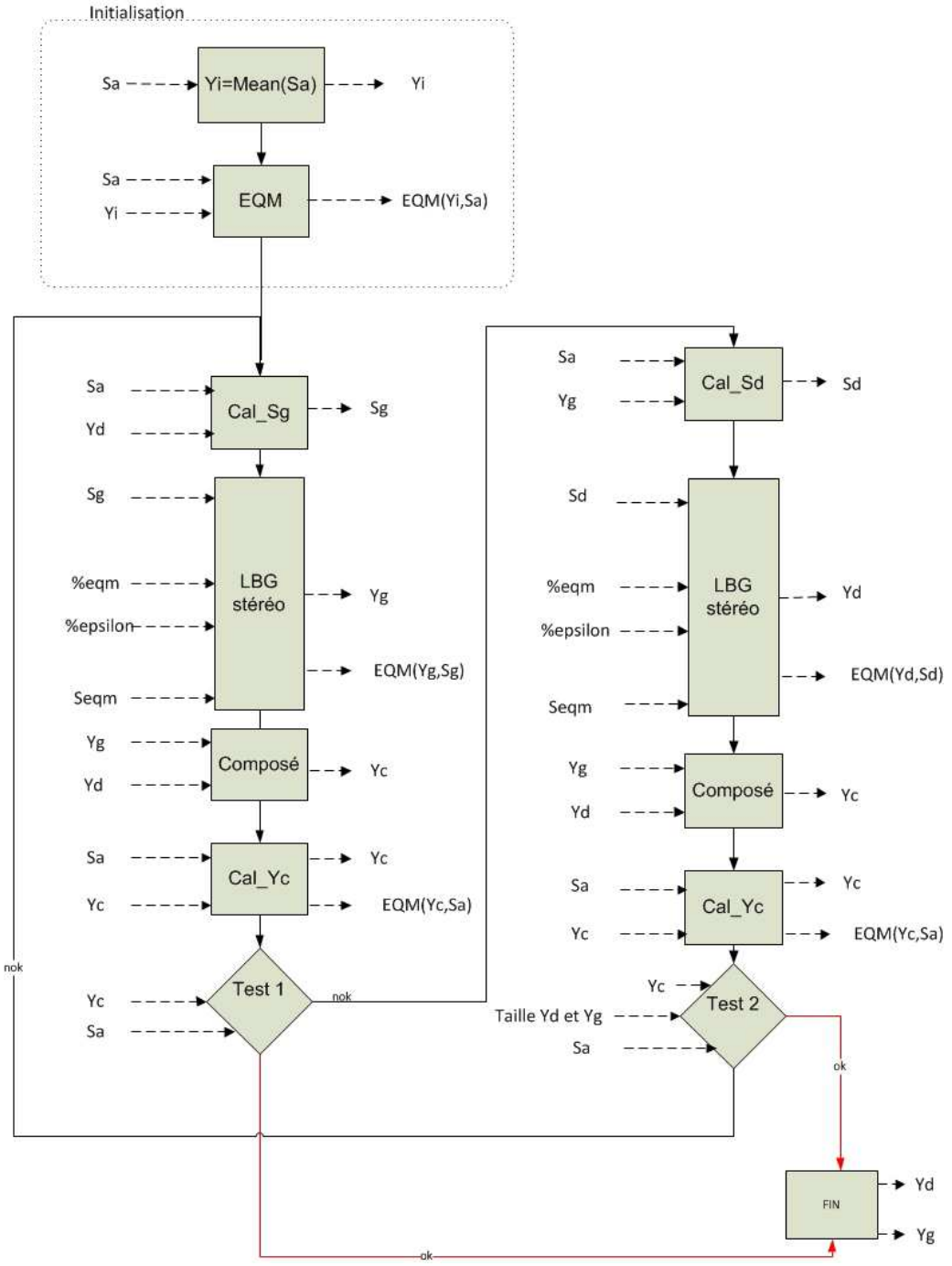

FIGURE 3.8: Détail de l'algorithme de création des dictionnaires Yd et Yg (le détail des paramètres est donné au paragraphe 3.3.2.

3.3.2 L'algorithme LBG stéréo

L'algorithme *LBG stéréo* est une évolution de l'algorithme *LBG*. Globalement, il permet à partir d'une séquence d'apprentissage, de créer un dictionnaire optimisé en fonction de paramètres qui lui sont fixés. Concernant notre projet, cet algorithme intervient toujours après la création d'une séquence, donc toujours après un processus *Cal_Sd* ou *Cal_Sg* (cf figure 3.6). La figure 3.9 montre les grands principes de l'algorithme *LBG stéréo* basé sur 4 grandes étapes, l'initialisation, le "splitting", l'algorithme de *Lloyd-Max* et le test de sortie.

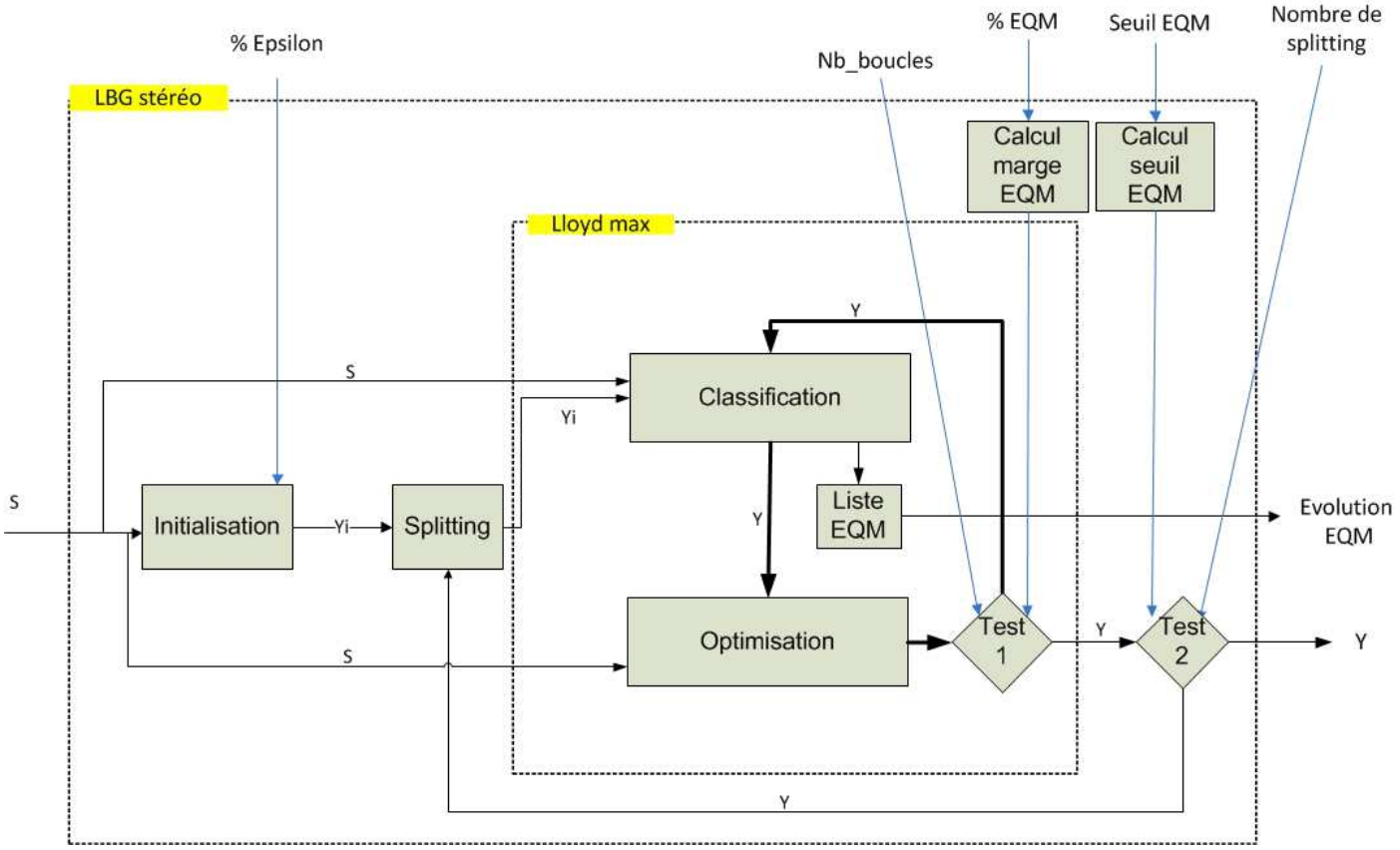

FIGURE 3.9: Principe de l'algorithme *LBG stéréo*.

Nous allons utiliser de nouveaux termes pour décrire cette partie :

- S : Séquence droite ou gauche ;

- *Epsilon* : Perturbation utilisée pour le "splitting" ;
- *EQM(Y,S)* : EQM entre le dictionnaire Y et la séquence S ;
- *tree* : Structure interne pour mémoriser les données lors des étapes successives de construction de l'arbre ;
- *nb_boucle* : Nombre de boucles de l'algorithme *LBG stéréo* ;
- *Seuil_EQM* : Pourcentage de l'EQM utilisé pour sortir l'algorithme ;
- *dynEQM* : Valeur de l'EQM pour stopper l'algorithme ;
- *Sindex* : Séquence S indexée en utilisant un dictionnaire Y.

Initialisation :

L'étape d'initialisation reste inchangée par rapport à l'algorithme *LBG*. Elle consiste à créer un vecteur représentant *Yi* qui est le barycentre de la séquence d'apprentissage *Sa* en entrée de l'algorithme.

Splitting :

Cette étape est importante, car elle influence la qualité du dictionnaire. Son principe est de créer deux vecteurs représentant pour chaque vecteur représentant déjà présent dans le dictionnaire. L'orientation du "splitting" des vecteurs peut-être déterminée selon plusieurs critères, de façon aléatoire, de façon alternée ou de façon orientée. Nous utiliserons le "splitting" suivant la règle $\vec{y} + \vec{\epsilon}$ et $\vec{y} - \vec{\epsilon}$ (cf. figure 3.10).

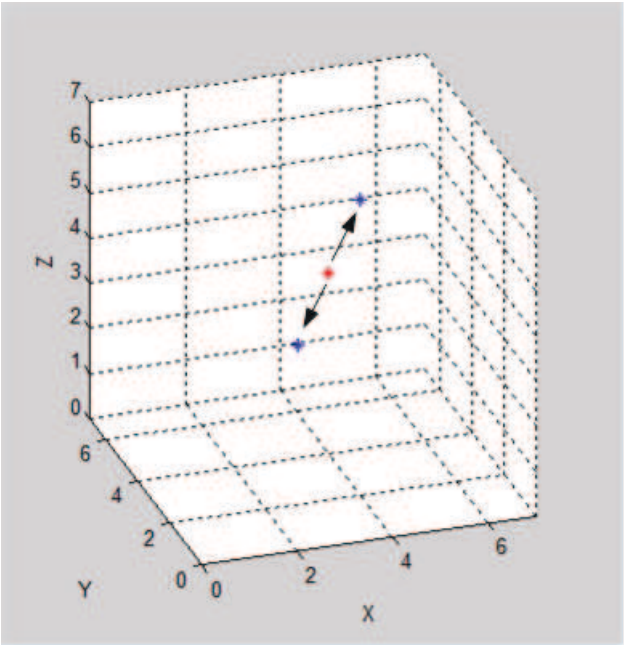

FIGURE 3.10: Illustration d'un splitting $\vec{y} + \vec{\epsilon}$ et $\vec{y} - \vec{\epsilon}$.

Des points importants sont la distance de "splitting" (la norme $\|\vec{\epsilon}\|$) par rapport au vecteur représentant traité, et son orientation, il existe plusieurs méthodes. Pour s'adapter au mieux à la séquence d'apprentissage en entrée de l'algorithme, il a été décidé de tenir compte de la dynamique des vecteurs de la séquence d'apprentissage. Parmi les paramètres du codec, il existe celui nommé "pourcentage Epsilon" ($\%Epsilon$). On calcule alors la distance entre les deux points les plus éloignés de la séquence d'apprentissage et on applique ce pourcentage pour déterminer la perturbation $Epsilon$:

$$\vec{\epsilon} = ((\overrightarrow{max(S)} - \overrightarrow{min(S)})/2) \cdot \%epsilon$$

L'algorithme de Lloyd-Max :

Cet algorithme itératif décrit dans la section 2.4.4 est composé de deux grandes opérations successives : la classification et l'optimisation. Pour les besoins du projet, deux conditions de sortie de la boucle ont été ajoutées, une condition sur le nombre d'itérations, et une condition sur l'EQM (cf. figure 2.8). Ces conditions garantissent la sortie de l'algorithme soit sur un critère de nombre suffisant d'itérations, soit si l'EQM ne baisse plus de façon significative entre deux itérations. La figure 3.11 illustre l'évolution d'une EQM ainsi que sa variation entre paires d'itérations successives. On remarque que cette courbe peut se prolonger sans pour autant faire varier la valeur de l'EQM de façon significative. Après plusieurs tests, il a été décidé de fixer le pourcentage de variation à 0.0001%, ce qui implique que l'algorithme continue ses itérations que si la variation entre les valeurs de l'EQM de l'itération *n+1* et celle de l'itération *n* est supérieure.

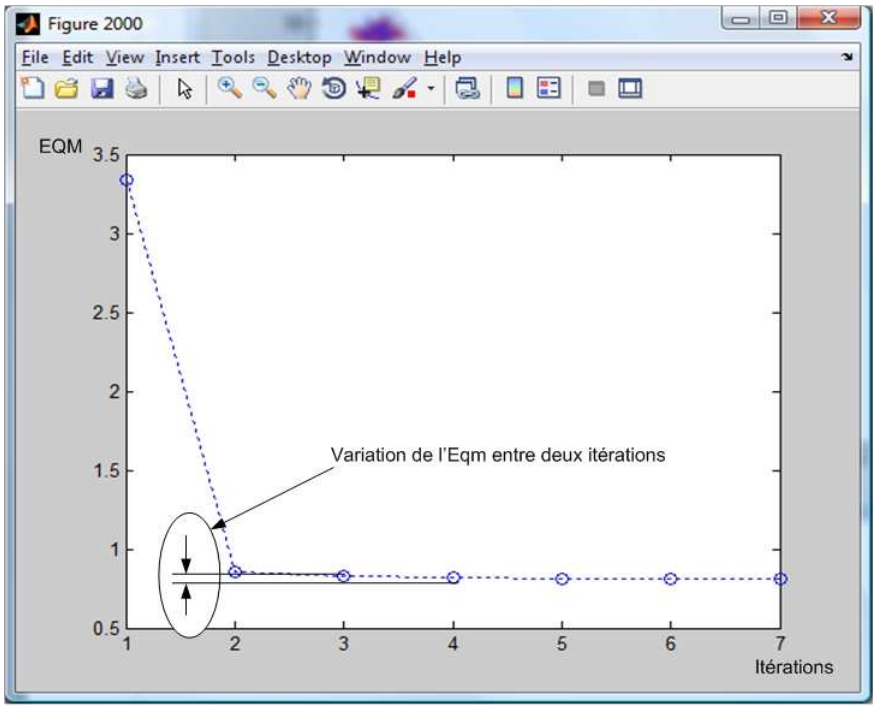

FIGURE 3.11: Exemple de courbe d'évolution de l'EQM.

Test de sortie

L'algorithme *LBG stéréo* procède par itérations, et chaque boucle accroit la taille du dictionnaire. Après classification de la séquence d'apprentissage à l'aide de ce nouveau dictionnaire, on calcule l'EQM qui sera comparée à un seuil. Pour s'adapter au mieux à la séquence en entrée, il a été décidé de fixer ce seuil en fonction de l'EQM initial ($EQM(Y_i,S)$)lorsque le dictionnaire est le barycentre de la séquence en entrée. La valeur du seuil de l'EQM pour sortir de l'algorithme ($dynEQM$) est :

$$dynEQM = \%EMQ(Yi, S)$$

3.3.3 Amélioration possible de l'algorithme

Nous avons vu que l'algorithme permet la création d'un dictionnaire composé Yc à partir de deux dictionnaires Yd et Yg. C'est la combinaison (fusion) de tous ces représentants (des dictionnaires droit et gauche) qui génère le dictionnaire composé qui est testé pour savoir s'il satisfait les conditions de sortie. On peut représenter la construction itérative de type LBG d'un dictionnaire sous la forme d'un arbre ayant pour racine le barycentre de la séquence initiale, les feuilles étant le dictionnaire final et où :

- à chaque nœud est associé un représentant,
- les nœuds intermédiaires, les représentants obtenus après splitting et itération de Lloyd,
- les feuilles, les représentants du dictionnaire final.

A chaque nœud correspond une cellule de Voronoï et son représentant, il est donc facile de calculer l'EQM de chaque représentant. Une amélioration possible consisterait à chaque boucle du LBG, plutôt que de "splitter" tous les nœuds, nous pouvons splitter uniquement le nœud pour lequel l'EQM est la plus grande. Après chaque "splitting" et *LBG stéréo*, il

faut calculer le dictionnaire composé Yc et vérifier si l'EQM de ce nouveau dictionnaire est assez petite. Si ce n'est pas le cas, il faut continuer à "splitter" le Voronoï ayant l'EQM la plus grande. Grâce à cette méthode, on peut cibler la zone pour laquelle il est nécessaire d'augmenter les représentants. Cette méthode assurera une économie en termes calculatoire mais demeurera sous-optimale en termes de représentation de la séquence d'apprentissage.

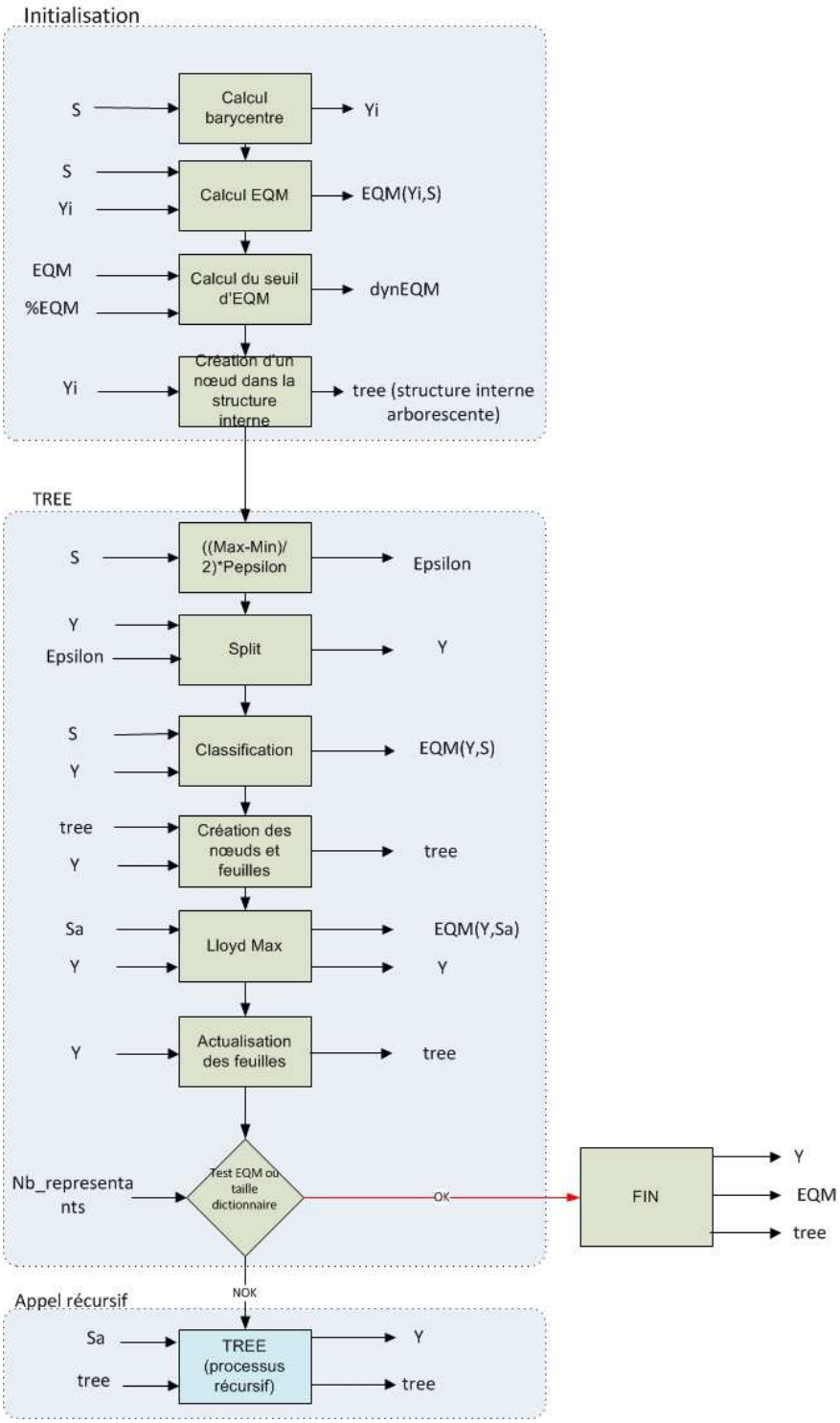

FIGURE 3.12: Détail de l'algorithme *LBG stéréo*.

On intègre dans l'algorithme *LBG stéréo* la structure d'arbre (cf. figure 3.14) notée *tree* qui mémorise toutes les informations de chaque vecteur représentant des différents dictionnaires obtenus lors des différentes boucles. Grâce à cette structure et aux propriétés énoncées dans la théorie des graphes, nous allons pouvoir exploiter toutes les possibilités de parcours de l'arbre, et à l'issue exporter le dictionnaire composé prenant en compte uniquement les feuilles.

L'autre intérêt de cette structure, lors d'une nouvelle itération de l'algorithme, il est possible de choisir quelle feuille doit être "éclatée".

Voici le contenu (la structure associée) d'une cellule de type *tree* :

N°	Contenu
1	Niveau (profondeur)
2	Index du père
3	Coordonnées des fils
4	Coordonnées du vecteur représentant (père)
5	EQM associée
6	Coordonnées des points source associés au Voronoï
7	Index des fils

Les liaisons se font de la façon suivante :

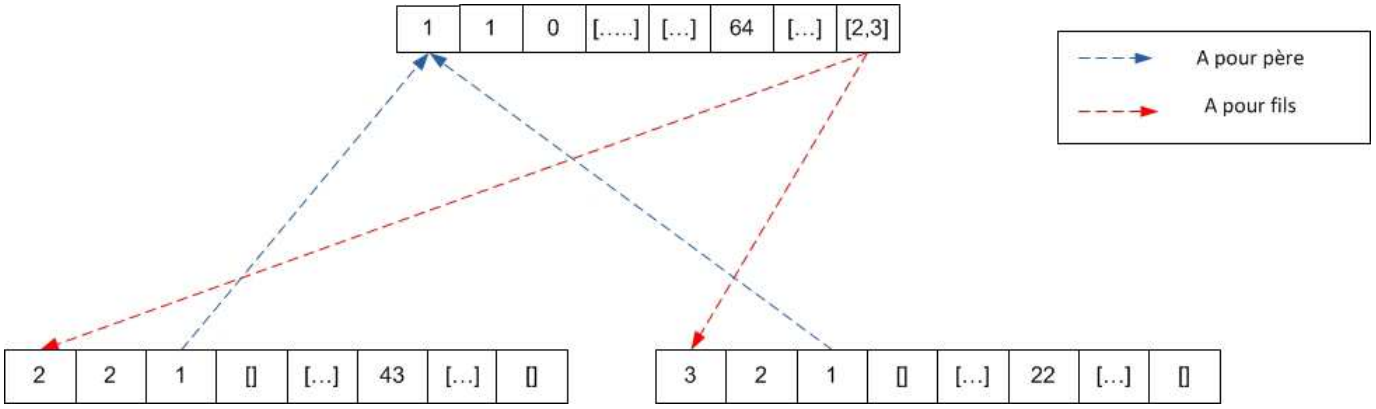

FIGURE 3.13: Illustration des liaisons entre cellules.

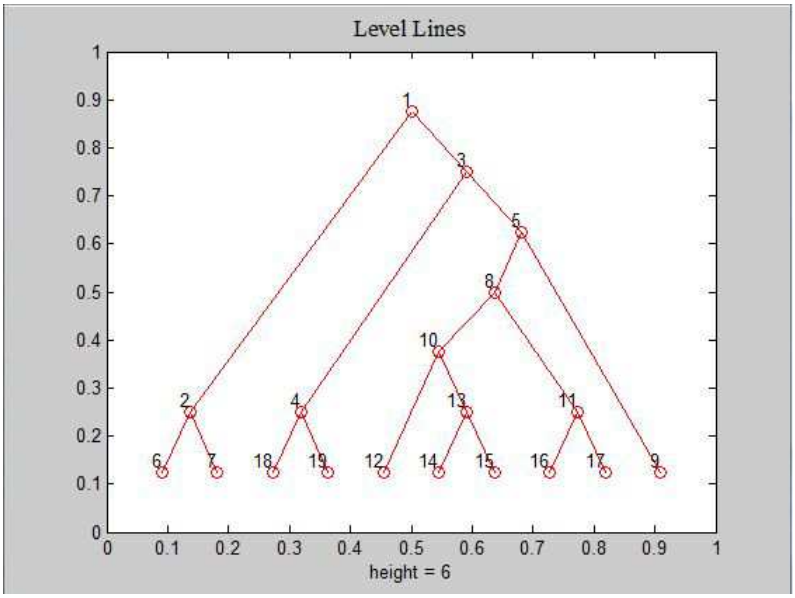

FIGURE 3.14: Exemple d'arbre correspondant à la construction d'un dictionnaire.

3.3.4 Exemple de fonctionnement du codec (mode 1)

Voici un exemple en 3D illustrant le déroulement du processus de création d'un dictionnaire composé. On considère une séquence d'apprentissage Sa (en rouge) constituée de quatre nuages de 50 vecteurs (soit une séquence d'apprentissage de 200 vecteurs).

- Initialisation (cf. figure 3.15) :

  Le premier dictionnaire Yi correspond au barycentre de la séquence Sa.

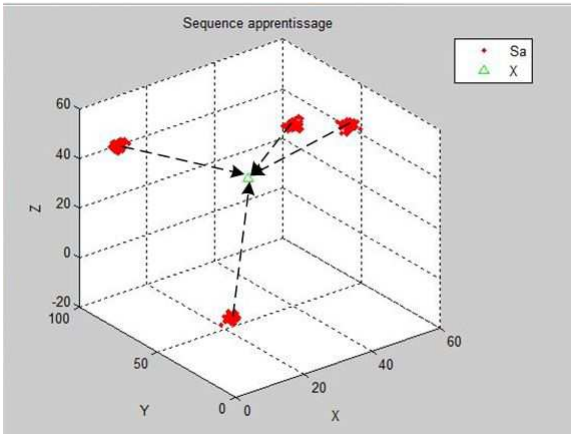

FIGURE 3.15: Initialisation.

- Construction de la séquence gauche Sg (cf. figure 3.16) :

  Projection des vecteurs source de la séquence Sa au travers du dictionnaire Yi.

  On créé une nouvelle séquence Sg (en jaune) avec autant de vecteurs que la séquence d'apprentissage.

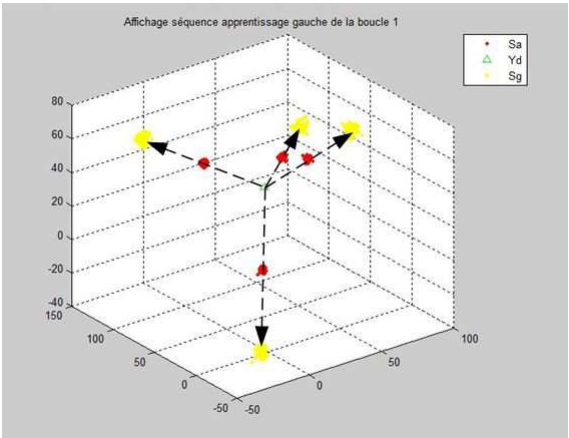

FIGURE 3.16: Séquence gauche Sg.

- Construction du dictionnaire gauche Yg (cf. figure 3.18) :

  L'algorithme passe ensuite par l'étape *LBG stéréo* pour créer le nouveau dictionnaire Yg obtenu après classification de la séquence d'apprentissage gauche.

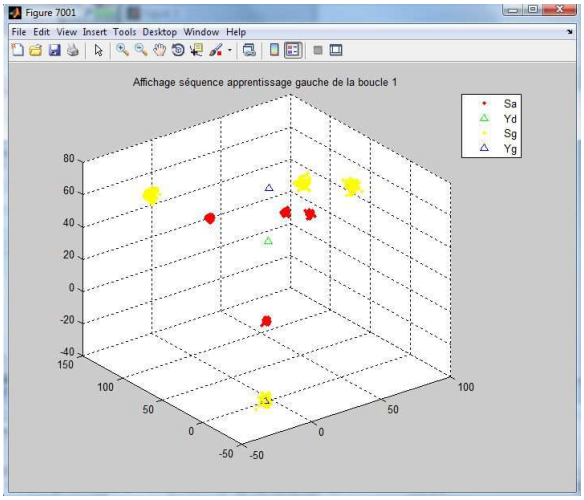

FIGURE 3.17: Séquence gauche *Sg* et dictionnaire gauche *Yg*.

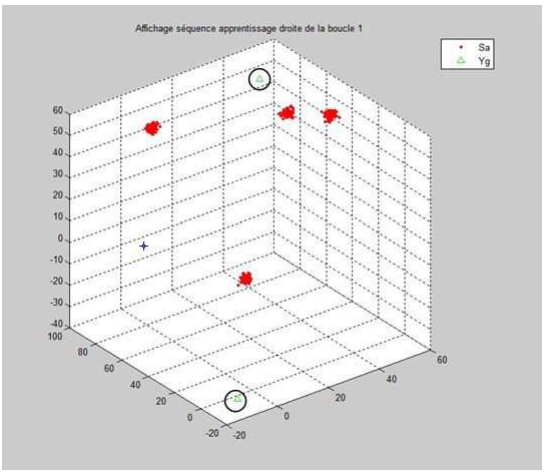

FIGURE 3.18: Dictionnaire gauche *Yg*.

Dans cet exemple, deux Voronoï sont créés. Une cellule de Voronoï contient 3 nuages (en haut) et une autre contient un seul nuage (en bas). Dans le cas de la première cellule de Voronoï, le représentant du dictionnaire est visible au milieu des 3 nuages et dans le cas de la deuxième cellule de Voronoï, le représentant est au milieu du nuage (cf. figure 3.17).

- Construction de la séquence droite Sd (cf. figure 3.19) :

  On applique le même principe que celui utilisé pour la séquence gauche.

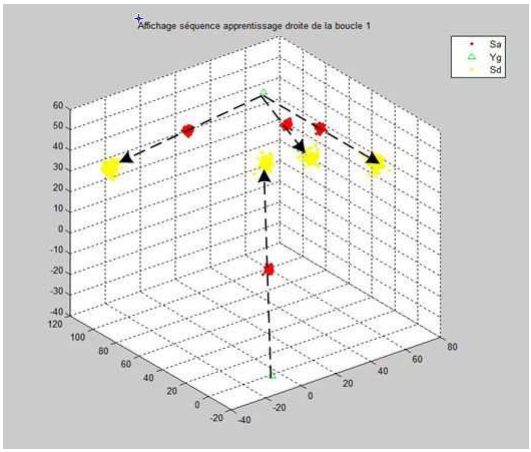

FIGURE 3.19: Séquence droite Sd.

- Construction du dictionnaire droit Yd (cf. figure 3.20) :

  On applique le même principe que celui utilisé pour le dictionnaire gauche.

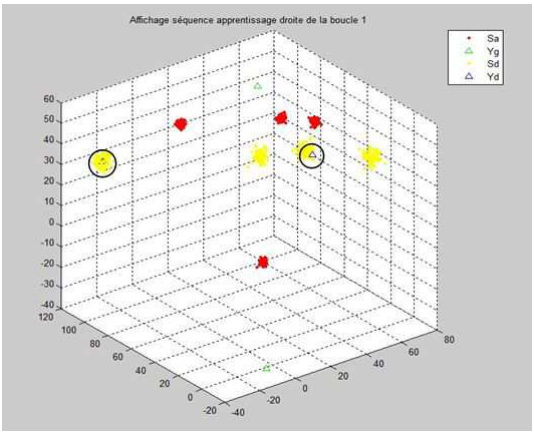

FIGURE 3.20: Dictionnaire droit Yd.

- Dictionnaire droit Yd et gauche Yg (cf. figure 3.21)

On conserve ensuite les deux dictionnaires créés ainsi que la séquence d'apprentissage.

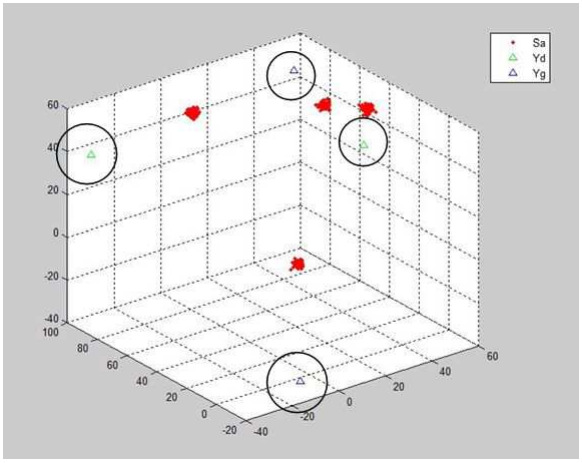

FIGURE 3.21: Dictionnaire droit et gauche Yg.

- Dictionnaire composé Yc (cf. figure 3.22) :

  Le dictionnaire composé est ensuite créé en effectuant toutes les combinaisons possibles entre les éléments du dictionnaire droit et du dictionnaire gauche.

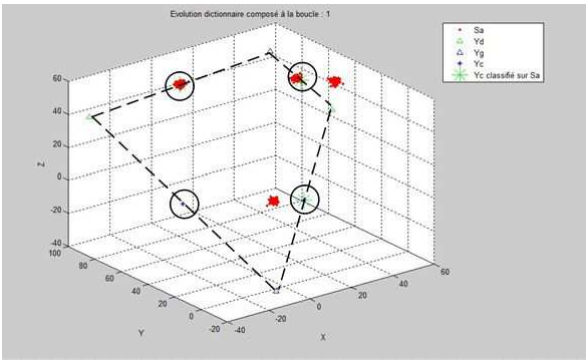

FIGURE 3.22: Dictionnaire composé Yc.

- Dictionnaire composé utile Ycu (cf. figure 3.23)

  On effectue ensuite une classification de la séquence d'apprentissage à l'aide du dictionnaire composé. En effet, certains éléments de ce dictionnaire ne sont pas utiles s'ils

ne représentent pas de façon efficace une partie de la séquence d'apprentissage.

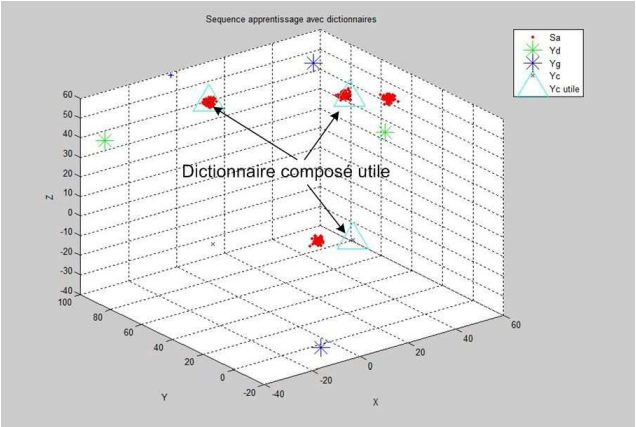

FIGURE 3.23: Dictionnaire composé : liste des représentants utiles *Ycu*.

Dans cet exemple, le dictionnaire composé correspond à quatre représentants, mais seuls trois représentants sont utilisés.

3.4 Codec (mode 2) avec une vidéo stéréoscopique

Maintenant que nous avons détaillé comment fonctionne l'algorithme avec une séquence synthétique, nous allons voir comment réaliser ces opérations sur des vidéos stéréo "réelles", en effectuant les étapes de création des dictionnaires par la méthode *LBG stéréo*, de quantification et de quantification inverse. La figure 3.24 nous montre quelles sont les principales étapes du codec.

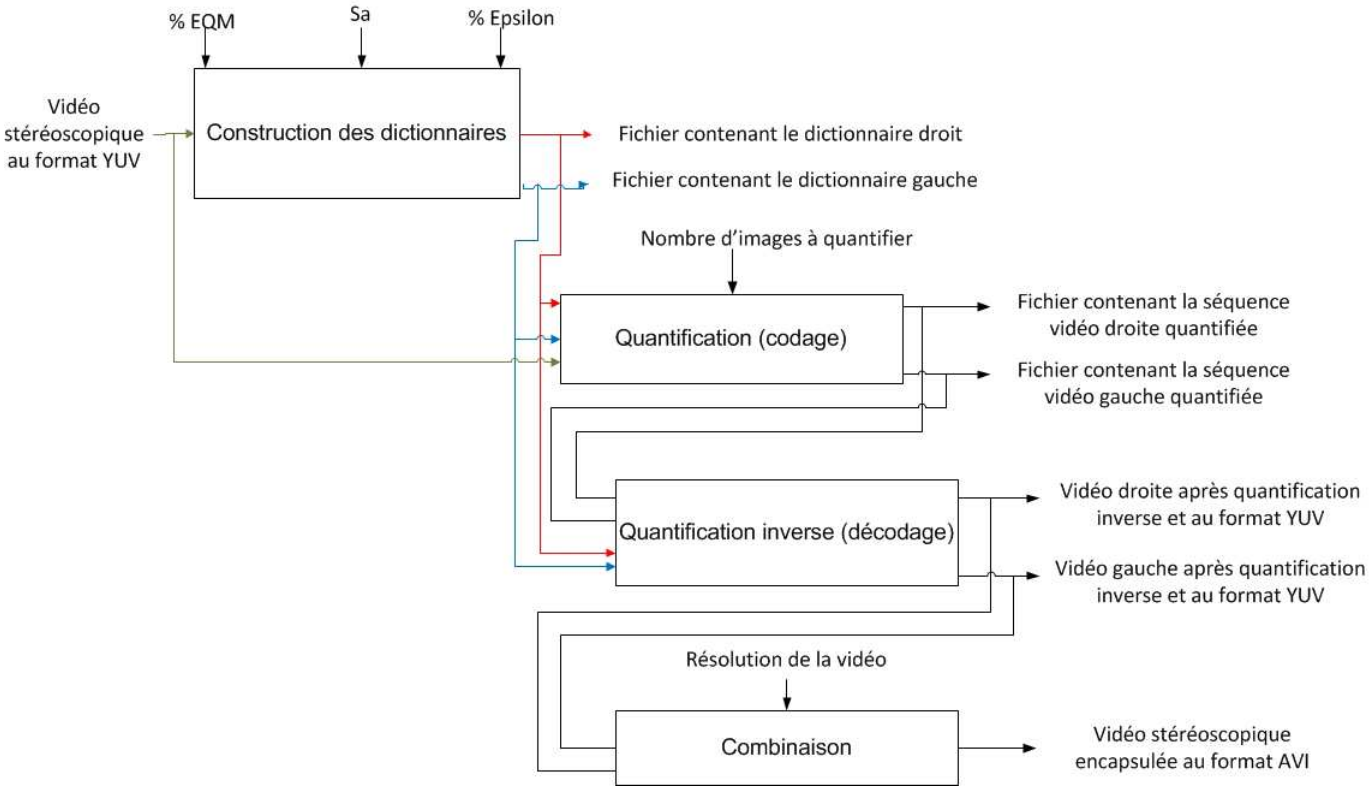

FIGURE 3.24: Codec "mode 2" quantification/déquantification d'une vidéo stéréo.

La génération des dictionnaires a déjà été décrite dans la section 3.3. La grande différence réside dans la constitution de la séquence d'apprentissage, pour cela on sélectionne un nombre d'images dans la séquence à coder. Cela se fait via trois nouveaux paramètres, le numéro de l'image de début, le numéro de l'image de fin et le pas (échantillonnage temporel) entre chaque image. Il est ainsi possible de sélectionner la fréquence et le nombre d'images de la vidéo stéréo. Une fréquence faible sur une longue vidéo couvrira un large spectre de couleurs mais les calculs seront longs, une sélection peut donc se concentrer sur moins d'images d'une portion précise de la vidéo.

La figure 3.25 décrit le déroulement total du processus en détaillant la création des dictionnaire par la méthode *LBG stéréo*. Le travail effectué sur le codec en "mode 1" est intégré dans la partie *Construction des dictionnaires*. Nous avons vu lors de l'utilisation du codec

"mode 1" que les données manipulées sont des vecteurs, un problème se pose alors, comment transformer une vidéo stéréoscopique en vecteurs, c'est le rôle des boites *Convertisseur format LUV* et *Création séquence d'apprentissage* que nous détaillerons dans les paragraphes 3.4.1 et 3.4.2.

Les étapes de codage et décodage sont réalisées de la manière décrite pour l'algorithme *LBG* (cf. chapitre 2.4), c'est-a-dire par "recherche du plus proche voisin" pour le codage, et "restitution du vecteur représentant" pour le décodage.

Pour coder et décoder une vidéo stéréoscopique, nous pouvons réaliser l'opération sur les images droite et gauche, ce qui implique d'utiliser deux séquences distinctes. Une solution serait de coder l'image source droite avec le dictionnaire droit et l'image source gauche avec le dictionnaire gauche, mais cette solution (que nous allons nommer *Side by Side*) ne tient pas compte de la couleur des pixels fusionnés que nous cherchons à optimiser.
Une autre solution est donc de réaliser la fusion des couleurs des pixels des images droite et gauche. La fusion des images gauche et droite est basée sur l'équation 3.1 qui devient :

$$P_f = \frac{P_g + P_d}{2} \tag{3.2}$$

où :

- P_f : est la valeur du pixel dans l'image fusionnée I_f.
- P_g : est la valeur du pixel de l'image gauche I_g.
- P_d : est la valeur du pixel de l'image droite I_d.

Les positions des 3 pixels dans les 3 images sont identiques, nous ne tenons donc pas compte de la disparité (hypothèse simplificatrice).

Pour le codage de la séquence fusionnée, pour chaque pixel fusionné, on obtient un représentant du dictionnaire composé. Ensuite, pour cet élément du dictionnaire composé, nous proposons de retenir son élément droit et son élément gauche, nous pouvons alors constituer deux nouvelles séquences vidéo, une droite indexée avec le dictionnaire droit et une gauche indexée avec le dictionnaire gauche.
Nous pouvons faire l'hypothèse que le schéma de codage avec une séquence fusionnée sera plus efficace que le schéma de codage *Side by Side*. Les résultats expérimentaux nous donneront des réponses.

Le format de sortie des fichiers quantifiés sera le format YUV, de façon simplifiée nous pouvons décrire l'opération de conversion en YUV comme étant l'opération inverse de la conversion en LUV.

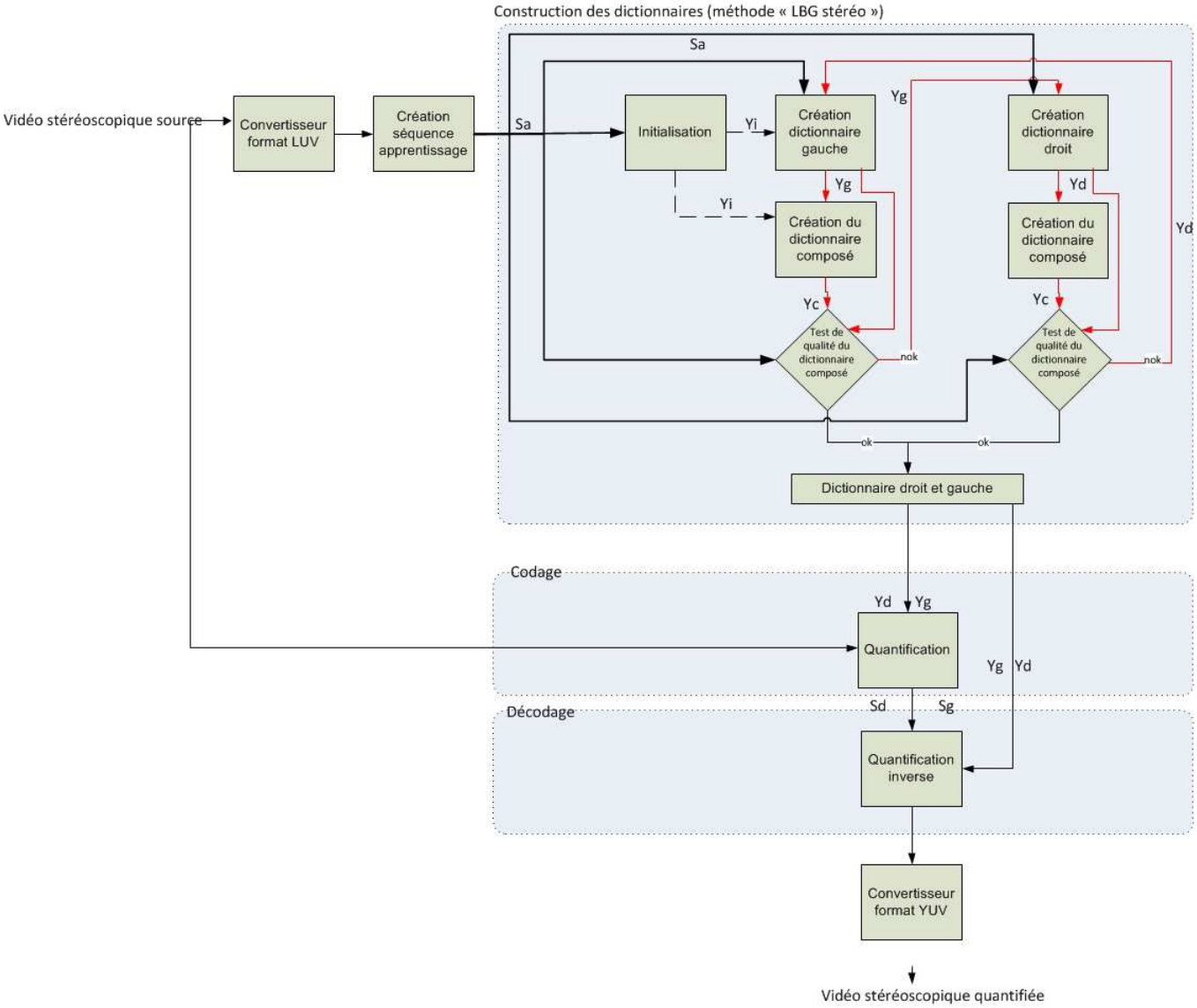

Figure 3.25: Algorithme global intégrant l'algorithme *LBG stéréo*.

3.4.1 Conversion des images au format LUV

Les vidéos stéréoscopiques à quantifier sont fournies au format YUV (cf. chapitre 2.3.6), nous avons vu dans le chapitre 2.3.5 que pour fusionner les couleurs il est nécessaire de passer dans l'espace colorimétrique LUV (cf. chapitre 2.3.5), c'est le rôle du programme *Convertisseur format LUV* (cf. figure 3.26).

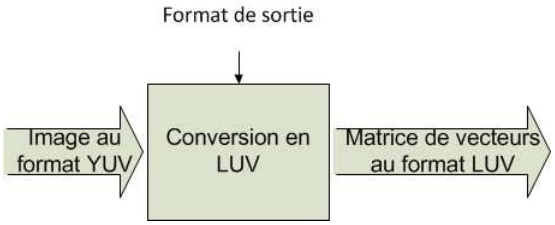

FIGURE 3.26: Convertisseur YUV vers LUV.

Le résultat de cette conversion est une matrice de n lignes et 3 colonnes, le nombre de lignes dépend de la résolution de sortie choisi ($longueur * largeur$) et le nombre de colonnes correspond aux données de LUV.

Le paramètre "Format de sortie" permet de préciser le format des images converties en LUV (espace vectoriel) et ainsi réduire le nombre de vecteurs en sortie, cette opération a pour objectif d'alléger les traitements effectués dans le processus "création séquence d'apprentissage".

Pour passer d'un espace colorimétrique à un autre, nous allons utiliser une conversion dans l'espace RGB (cf. figure 3.27).

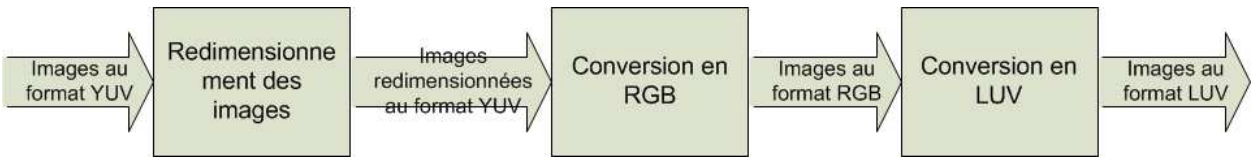

FIGURE 3.27: Principales étapes pour la conversion YUV vers LUV.

Les images sont ainsi converties dans l'espace LUV , format nécessaire pour les opérations effectuées sur les vecteurs représentants les couleurs de la séquence d'apprentissage.

3.4.2 Choix d'images dans une vidéo stéréoscopique pour la création d'une séquence d'apprentissage

Prenons par exemple la vidéo *nurburgring_24hour_race* qui a pour thème celui d'une course automobile. Cette vidéo est composée d'une séquence de 299 images stéréo au format "HD" $1920 \cdot 1080$, elle a des plans ayant des "profils" de couleurs différents (cf. figure 3.28).

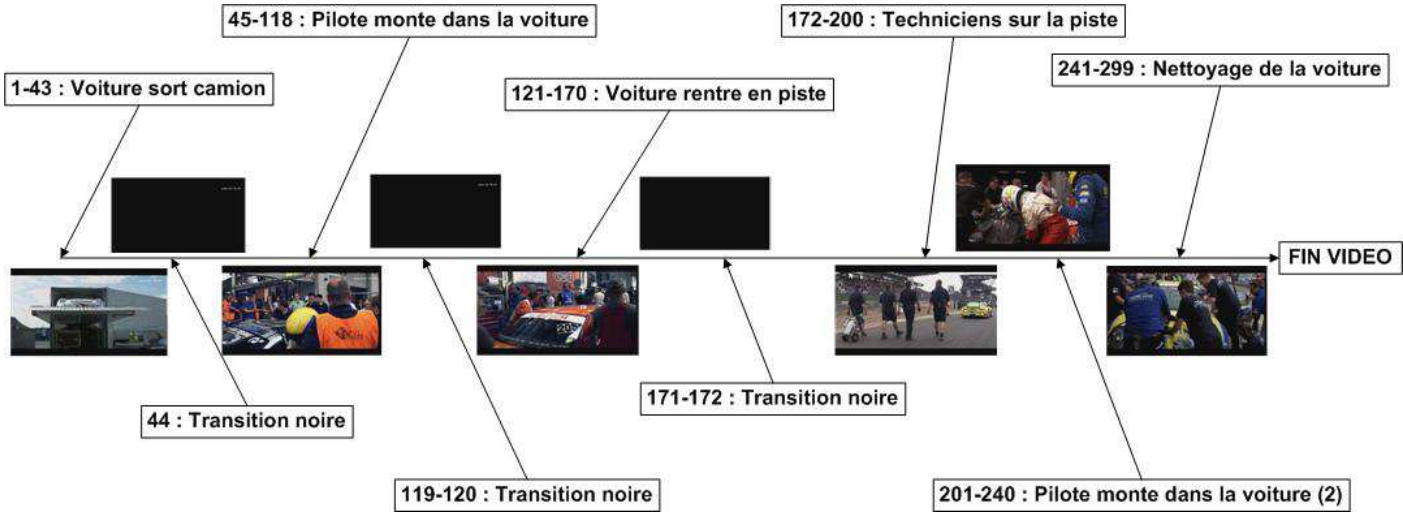

FIGURE 3.28: Plans de la vidéo *nurburgring_24hour_race*.

On peut par exemple générer une séquence d'apprentissage de petite taille en choisissant les images numéro 200 et 210 (cf. figure 3.29), ou bien générer une séquence d'apprentissage de grande taille en prenant par exemple une 1 image toutes les 10 dans l'intervalle $10-300$ soit 29 images.

FIGURE 3.29: Exemple de sélection d'images pour une petite séquence d'apprentissage.

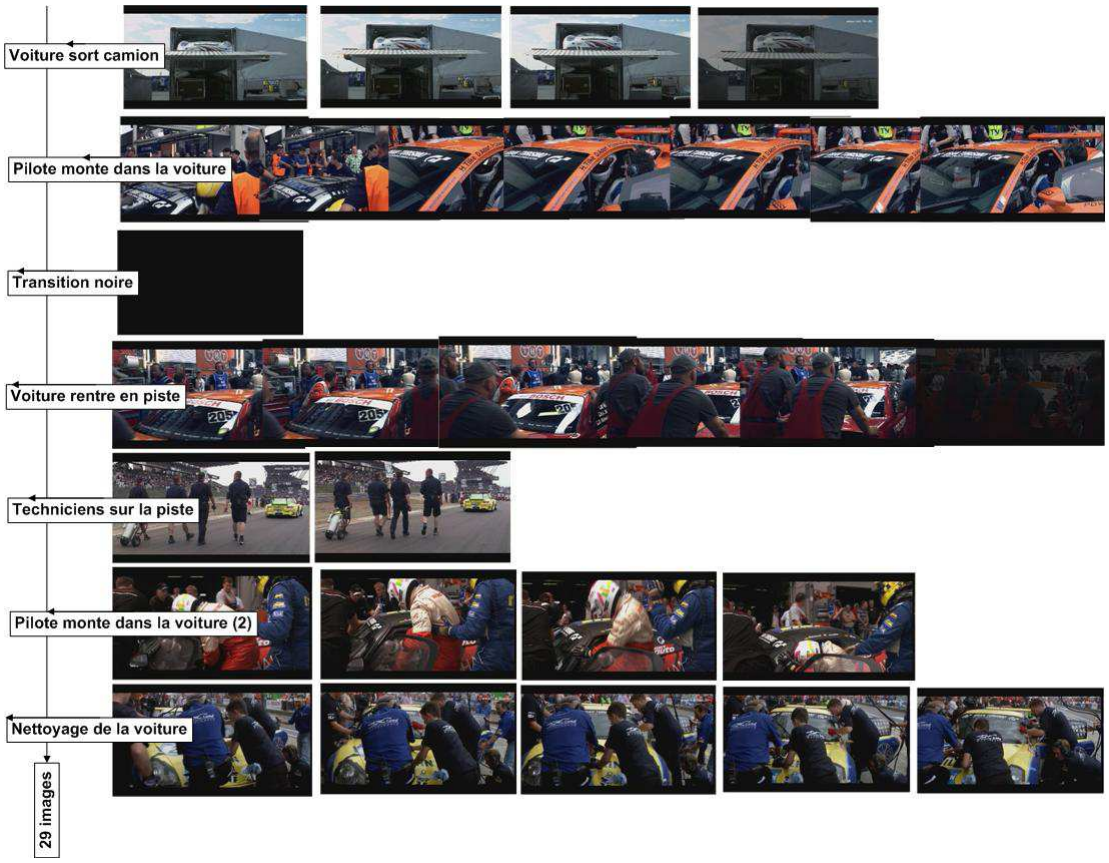

Figure 3.30: Exemple de sélection d'images pour une grande séquence d'apprentissage.

Le choix d'une séquence d'apprentissage pour la création d'un dictionnaire conditionne les couleurs que nous pouvons retenir pour quantifier une vidéo stéréoscopique et donc la qualité de reconstruction de la vidéo.

3.4.3 Comparaison des performances des algorithmes *LBG stéréo* et *LBG* pour la construction des dictionnaires

Maintenant que nous disposons d'un algorithme de création de dictionnaires pour la fusion stéréoscopique (*LBG stéréo*), la prochaine étape consiste à comparer les performances de cet algorithme avec celui à base de *LBG*. Cette comparaison peut se faire dans notre cas de deux façons, à l'aide d'une technique visuelle et subjective, à l'aide d'une méthode analytique (objective) basée sur le calcul du PSNR ou par comparaison des EQM dans l'espace LUV.

Pour réaliser ces tests, un nouveau programme a été réalisé permettant, a partir de deux images stéréoscopiques, la création de deux paires de fichiers stéréoscopiques, une réalisée par quantification à base de dictionnaires de type *LBG stéréo*, et l'autre par quantification à base de dictionnaires de type *LBG* (cf. figure 3.31). Les images ainsi créées seront utilisées pour effectuer les tests.

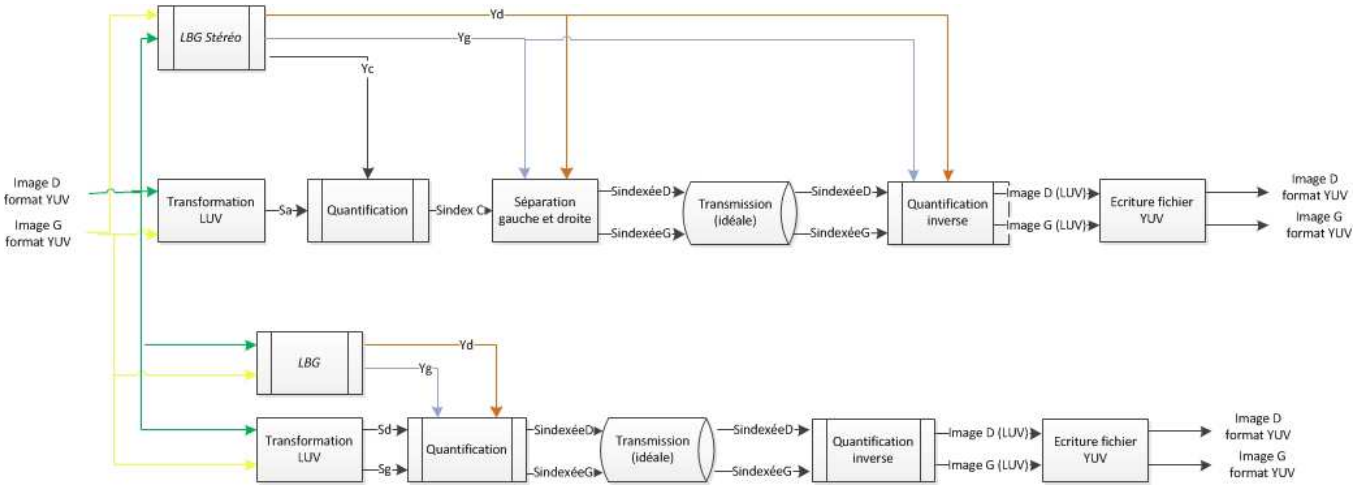

FIGURE 3.31: Processus de création d'images quantifiées avec l'algorithme *LBG stéréo* ou *LBG*.

Les images utilisées pour les tests sont des images provenant de la base de données du site *Middlebury Stereo Vision* [41]. Nous utiliserons plus particulièrement les images 1 et 5 de la base "2005".

Tests objectifs de la construction des dictionnaires par comparaison d'EQM

Pour comparer les performances de l'algorithme *LBG* et l'algorithme *LBG stéréo*, nous pouvons observer l'évolution des EQM obtenues lors de la construction des dictionnaires selon la méthode *LBG stéréo* ou *LBG*, et en utilisant pour S_a la séquence fusionnée (cf. équation 3.2). La figure 3.32 montre les résultats obtenus avec les images *Drumsticks_ s1390x1110p25n1v0.yuv* et *Drumsticks_ s1390x1110p25n1v1.yuv* au format $YUV4:2:2$, le nombre d'itérations de Lloyd étant fixé à 10 et la taille maximum des dictionnaires à 200.

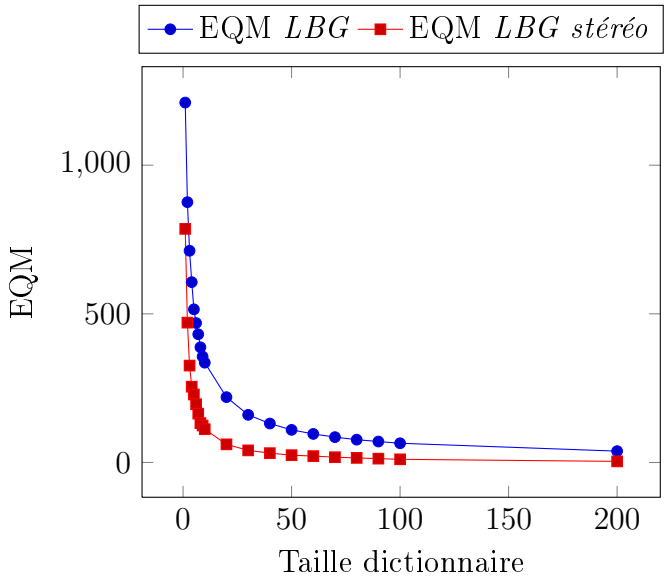

FIGURE 3.32: Evolution de l'EQM des dictionnaires *LBG* et *LBG stéréo* sur la séquence d'apprentissage fusionnée.

Les résultats montrent que les valeurs de l'EQM avec le dictionnaire *LBG stéréo* sont plus petites que les valeurs avec le dictionnaire *LBG*. Nous pouvons donc déduire à ce niveau que la méthode par *LBG stéréo* est plus performante pour représenter une séquence d'apprentissage dans l'espace LUV.

3.4.4 Comparaison des performances lors du codage/décodage de séquences stéréos

Tests objectifs de codage (quantification) de séquences stéréos, et basés sur le calcul des PSNR

A l'aide des dictionnaires créés avec les deux algorithmes *LBG* et *LBG stéréo*, nous pouvons

réaliser des tests basés sur le PSNR (cf. chapitre 2.4.3) sur les images fusionnées (cf. équation 3.2) ou sur les images *Side by Side*. Contrairement à la séquence fusionnée, la séquence *Side by Side* est créée par concaténation des deux images droite et gauche, celle-ci sera donc deux fois plus grande que la séquence fusionnée. Il est ainsi possible d'observer la quantification effectuée de chaque image (monoculaire).

La figure 3.33 représente le processus de test pour les images fusionnées (cf. équation 3.2).

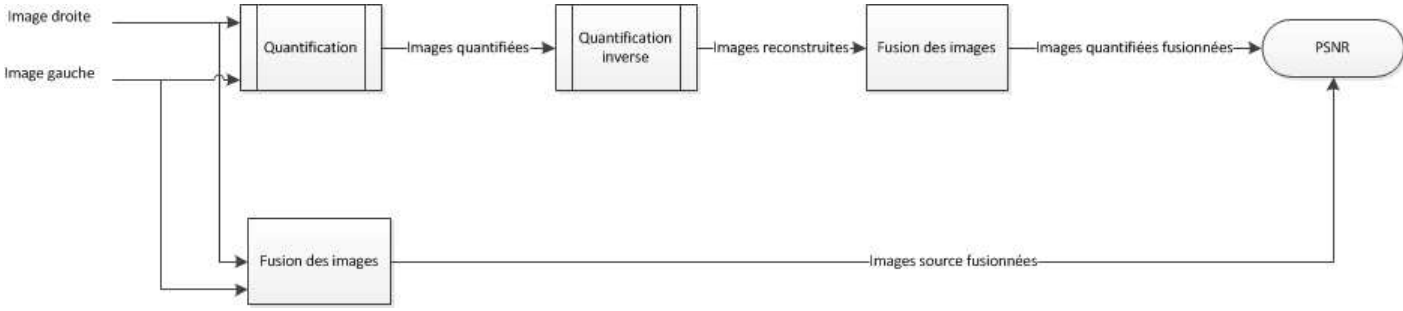

FIGURE 3.33: Processus de test de quantification d'images fusionnées.

La figure 3.34 montre les résultats pour les PSNR obtenus lors du codage (quantification) de la séquence fusionnée selon que l'on utilise un dictionnaire obtenu avec le *LBG* ou un dictionnaire obtenu avec le *LBG stéréo*. L'axe des abscisses représente le nombre de représentants du dictionnaire. L'axe des ordonnées représente le PSNR.

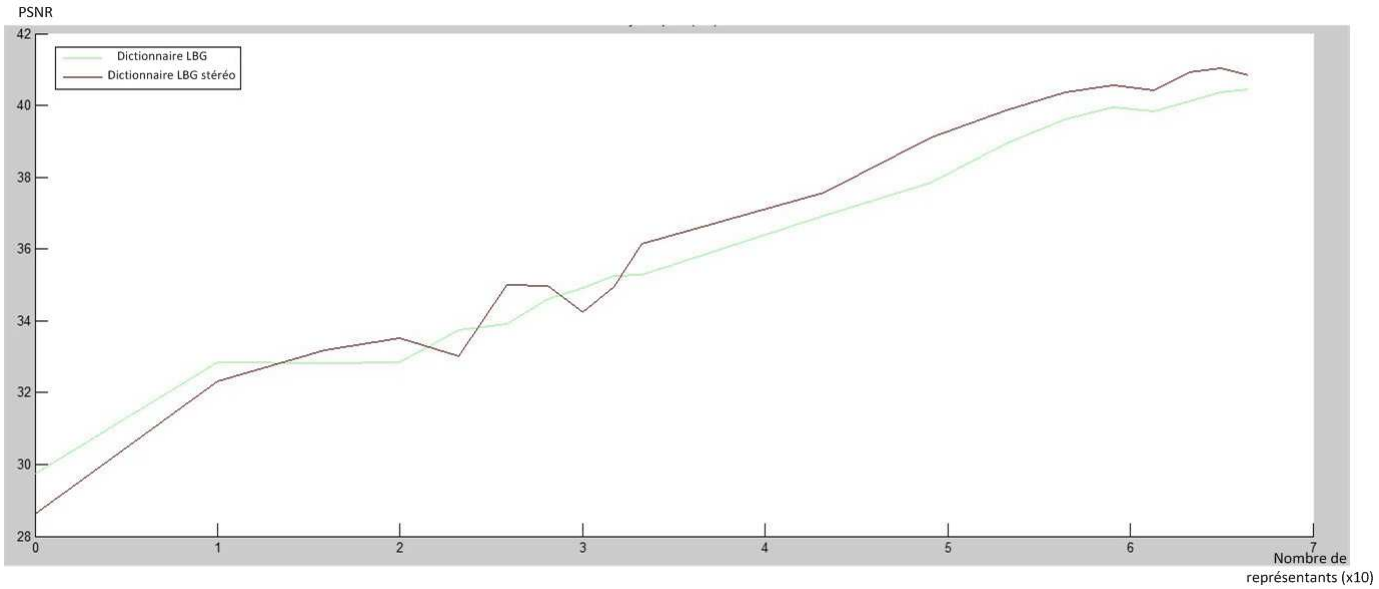

FIGURE 3.34: Comparaison des PSNR pour le codage de la séquence fusionnée (2 dictionnaires possibles : *LBG*, *LBG stéréo*).

Les résultats montrent que les courbes se suivent avec un résultat légèrement meilleur pour la quantification *LBG stéréo* pour des dictionnaires de taille supérieure. Nous pouvons donc déduire à ce stade que cette version de l'algorithme *LBG stéréo* pour la construction du dictionnaire n'apporte pas d'amélioration significative pour la représentation des images fusionnées.

La figure 3.35 représente le processus de test pour les images *Side by Side*.

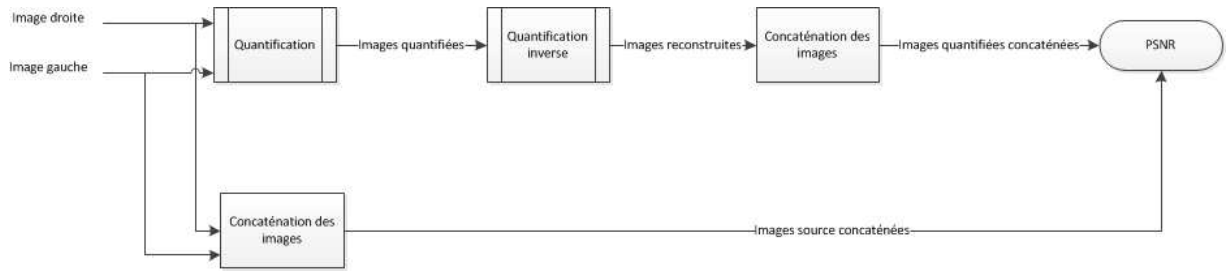

FIGURE 3.35: Processus de test PSNR sur des images *Side by Side*.

La figure 3.36 montre les résultats pour les PSNR obtenus lors du codage (quantification) d'images *Side by Side* selon que l'on utilise un dictionnaire obtenu avec le *LBG* ou un diction-

naire obtenu avec le *LBG stéréo*. L'axe des abscisses représente le nombre de représentants du dictionnaire. L'axe des ordonnées représente le PSNR.

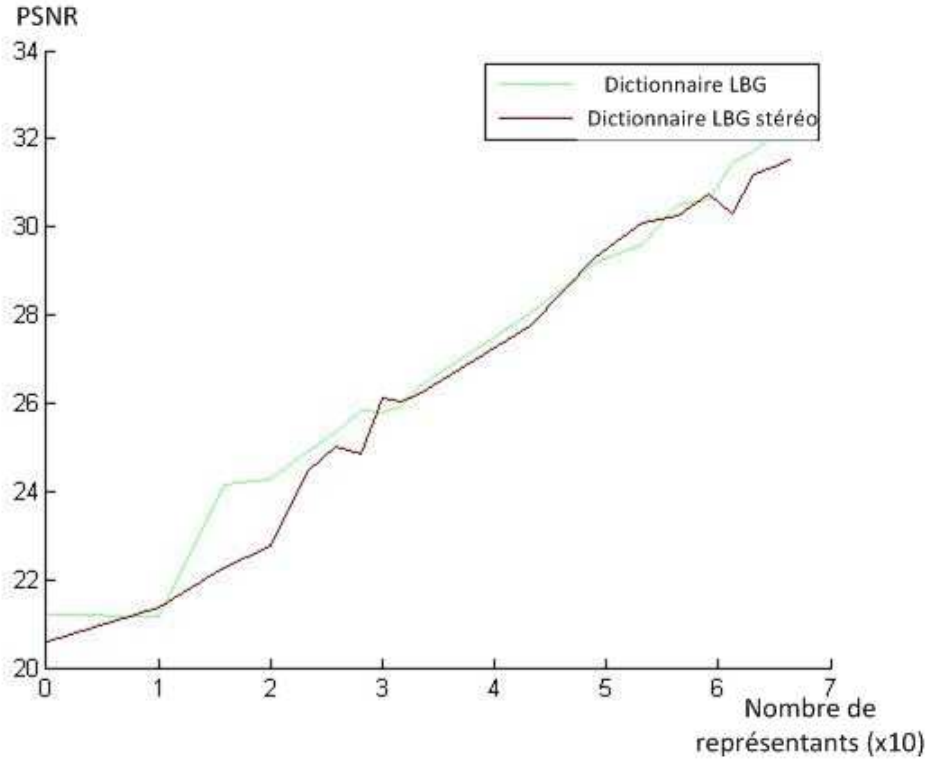

FIGURE 3.36: Comparaison des PSNR pour le codage des images *Side by Side* (2 dictionnaires possibles : *LBG*, *LBG stéréo*).

Les résultats montrent qu'il y a peu d'écart entre les courbes des images quantifiées selon la méthode *LBG* ou celle *LBG stéréo*. Nous pouvons donc déduire à ce stade que cette version de l'algorithme *LBG stéréo* pour la construction du dictionnaire n'apporte pas d'amélioration significative pour la représentation des images *Side by Side*.

Néanmoins, en comparant les deux résultats des deux tests, nous pouvons à ce stade en déduire que la version de l'algorithme *LBG stéréo* produit un dictionnaire qui représente mieux la séquence fusionnée que les images *Side by Side*.

Tests subjectifs de la qualité des images quantifiées/déquantifiées

A l'aide des paires d'images *Side by Side* préalablement quantifiées à l'aide d'un dictionnaire de taille 50 représentants, les deux types de dictionnaires (*LBG* versus *LBG stéréo*) sont respectivement utilisées. Nous pouvons observer la différence de qualité entre les images reconstruites (décodées) (cf. figure 3.37).

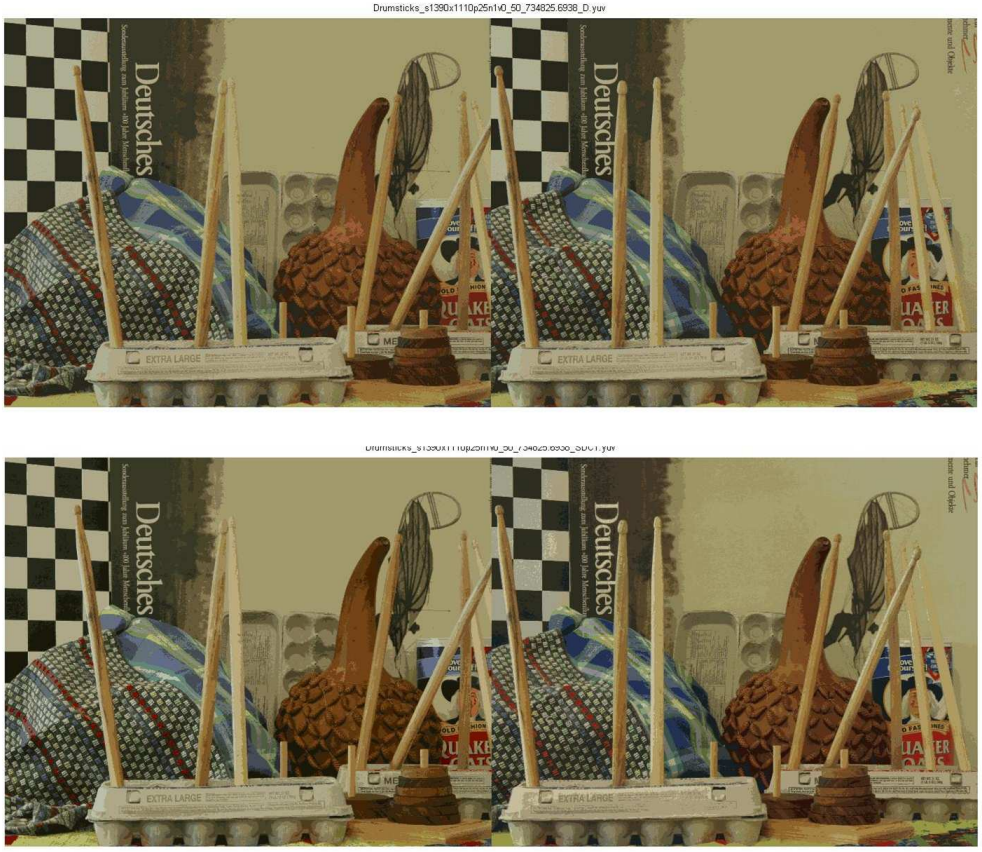

FIGURE 3.37: Images *Side by Side* quantifiées/déquantifiées : en bas avec un dictionnaire selon la méthode *LBG* et images en haut, un dictionnaire selon la méthode *LBG stéréo*.

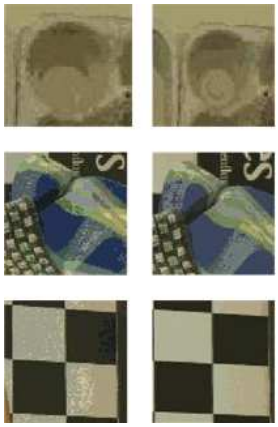

FIGURE 3.38: Agrandissement des zones des images *Side by Side* quantifiées/déquantifiées : à droite avec un dictionnaire selon la méthode *LBG* et à gauche selon la méthode *LBG stéréo*.

Après agrandissement de plusieurs zones (cf. figure 3.38) des deux images, nous pouvons observer que certaines parties de l'image quantifiée avec un dictionnaire *LBG* donne sensiblement de meilleurs résultats que l'image quantifiée avec le dictionnaire *LBG stéréo*. Sur l'ensemble des images, nous ne pouvons pas à ce stade déterminer la solution la plus efficace. Le résultat de cette observation est cohérent avec les résultats obtenus à l'aide des tests PSNR qui annonçaient une qualité globalement identique.

A l'aide des images fusionnées, nous pouvons observer la différence de qualité entre les images reconstruites et préalablement quantifiées à l'aide d'un dictionnaire de taille 50 (cf. figure 3.39). Les deux types de dictionnaires sont respectivement utilisés.

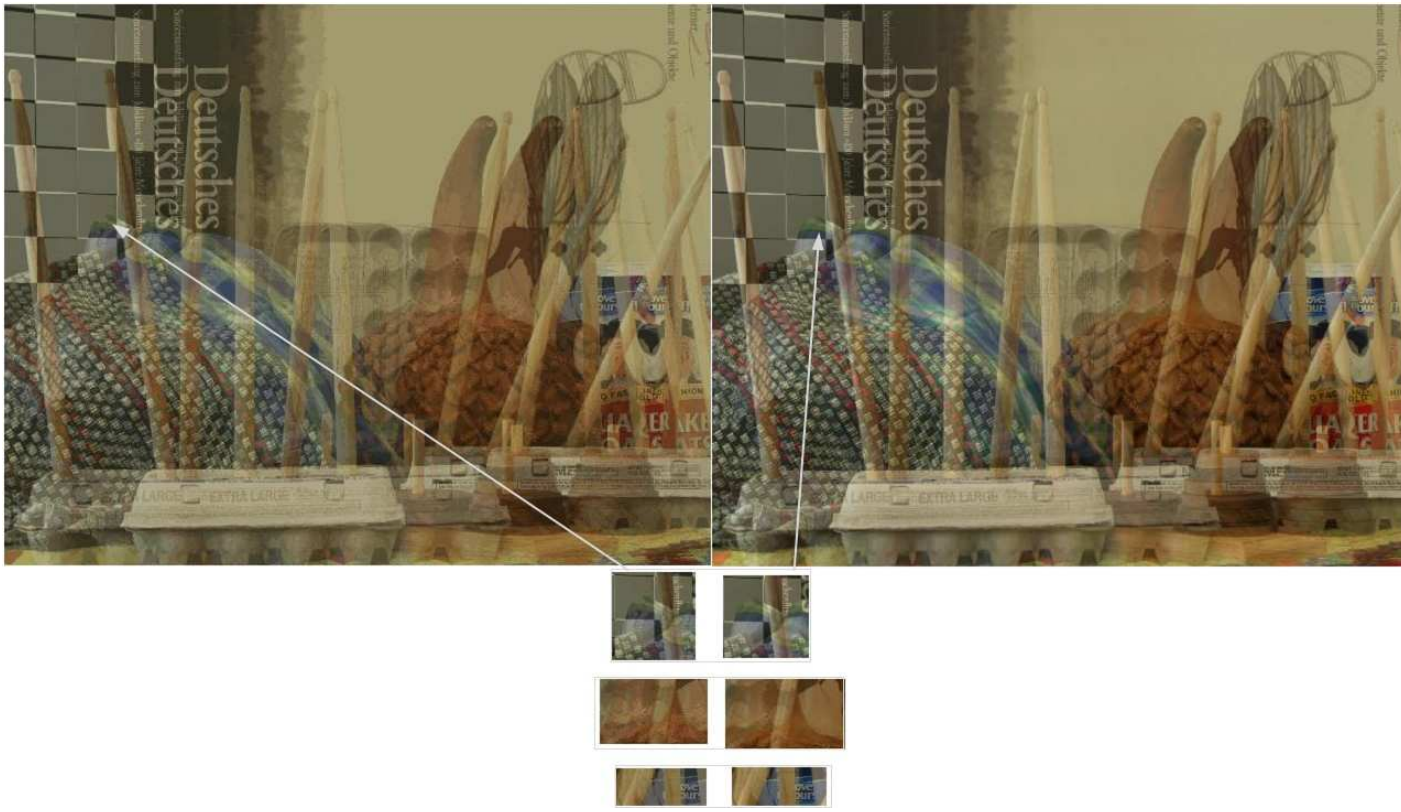

Figure 3.39: Images fusionnées quantifiées/déquantifiées : à gauche avec un dictionnaire *LBG*, à droite un dictionnaire *LBG stéréo*.

Après agrandissement de plusieurs zones des deux images, nous pouvons observer que certaines parties de l'image quantifiée avec le dictionnaire *LBG stéréo* donne sensiblement de meilleurs résultats que l'image quantifiée avec le dictionnaire *LBG*. Cependant, sur l'ensemble des images, nous ne pouvons pas à ce stade déterminer la solution la plus efficace. Le résultat de cette observation correspond aux résultats obtenus à l'aide des tests qui annonçaient une qualité globalement légèrement meilleure avec la solution basée *LBG stéréo*.

Chapitre 4

Environnement et gestion de projet

4.1 Gestion de projet

Mme MARTIN (2007) [27] a rapporté que la recherche et le développement expérimental englobent les travaux de création entrepris de façon systématique en vue d'accroître la somme des connaissances, y compris la connaissance de l'homme, de la culture et de la société, ainsi que l'utilisation de cette somme de connaissances pour de nouvelles applications. Elle rapporte qu'il est possible de scinder cette recherche en trois activités :

- La *recherche fondamentale* qui consiste en des travaux expérimentaux ou théoriques entrepris principalement en vue d'acquérir de nouvelles connaissances sur les fondements des phénomènes et des faits observables, sans envisager une application ou une utilisation particulière.

- La *recherche appliquée* qui consiste également en des travaux originaux entrepris en vue d'acquérir des connaissances nouvelles. Cependant, elle est surtout dirigée vers un but ou un objectif pratique déterminé.

- Le *développement expérimental* qui consiste en des travaux systématiques fondés sur des connaissances existantes obtenues par la recherche et/ou l'expérience pratique, en vue de lancer la fabrication de nouveaux matériaux, produits ou dispositifs, d'établir de nouveaux procédés, systèmes et services ou d'améliorer considérablement ceux qui existent déjà.

Elle rapporte qu'un projet de recherche présente 4 caractéristiques : la non-spécificité, les retards et délais et l'incertitude, mais ces caractéristiques tendent à diminuer au fur et à mesure qu'un projet passe à travers différentes étapes pour atteindre l'innovation finale.
Ce mémoire s'inscrit dans un contexte de recherche fondamentale qui doit tenir compte des caractéristiques décrites précédemment. Il est donc nécessaire d'utiliser une méthodologie de conduite de projet adaptée à ces contraintes. Pour ce faire, il a été décidé d'utiliser les méthodes dites "méthodes agiles", elles visent à réduire les phases de développement afin de

proposer de façon rapide des prototypes. On développe pour cela une version minimale, puis on y intègre des fonctionnalités par un processus itératif basé sur les retours du demandeur suite aux tests effectués.

L'origine des méthodes agiles est liée à l'instabilité de l'environnement technologique et au fait que le client est souvent dans l'incapacité de définir ses besoins de manière exhaustive dès le début du projet. Le terme « agile » fait ainsi référence à la capacité d'adaptation, aux changements de contexte et aux modifications de spécifications intervenant pendant le processus de développement. En 2001, 17 personnes mirent ainsi au point le manifeste agile dont la traduction est la suivante [13] :

- individus et interactions plutôt que processus et outils,
- développement logiciel plutôt que documentation exhaustive,
- collaboration avec le client plutôt que négociation contractuelle,
- ouverture au changement plutôt que suivi d'un plan rigide.

Grâce aux méthodes agiles, le client est pilote à part entière de son projet et obtient très vite une première mise en production de son logiciel.

Il existe plusieurs méthodes agiles :

- RAD (Rapide Application Déploiement), méthode basée sur 3 phases (cadrage, design et construction) ;
- DSDM (Dynamic Software Development Method), basée sur la méthode RAD il comble ses lacunes en prenant en compte l'ensemble du cycle de développement ;
- UP (Unified Process), est un processus de développement itératif et incrémental.

- RUP (Rational Unified Process) est une méthode de développement par itérations intégrant les équipes et un calendrier ;
- XP (eXtreme Programming) place le client au coeur du développement.

La méthode utilisée pour ce projet se rapproche de la méthode "UP". En effet, celle-ci est conçue dès le départ de façon très pragmatique : elle est adaptée au contexte de travail, aux besoins, aux possibilités de réutilisation de « briques » préexistantes. L'élaboration de l'architecture est d'abord grossière et indépendante des cas d'utilisation, puis un sous-ensemble des fonctions essentielles est identifié et l'architecture est reprise et détaillée suivant cet ensemble. De la spécification à la précision des cas, l'architecture évolue, incluant finalement de nouveaux cas, ainsi de suite, jusqu'à ce que l'architecture ait atteint un niveau de développement suffisamment élevé et stable pour donner lieu au développement d'un prototype qui sera présenté achevant ainsi une itération.
Une première version du codec ainsi été développée de façon simple, puis présentée lors de réunions hebdomadaires. Au fur et à mesure de la présentation des résultats, plusieurs versions du codeur ont été abandonnées, car les résultats ne correspondaient pas aux attentes. De nouvelles hypothèses ou pistes sont proposées et donnent lieux au développement de nouvelles versions de simulateur l'algorithme jusqu'à arriver à une version satisfaisante pour des cas simples. Ensuite, plusieurs parties sont modifiées pour orienter les résultats suivant plusieurs nouvelles hypothèses émises et les résultats sont présentés lors de réunions mensuelles. Trois versions du logiciel sont ainsi maintenues, ce qui permet de revenir facilement sur une version stable en cas d'abandon d'un développement.

Un planning prévisionnel a été créé en début de projet (cf. figure 4.1) et réactualisé en fonction de l'évolution du projet. Le dernier planning réalisé courant novembre est représenté à la figure 4.2.

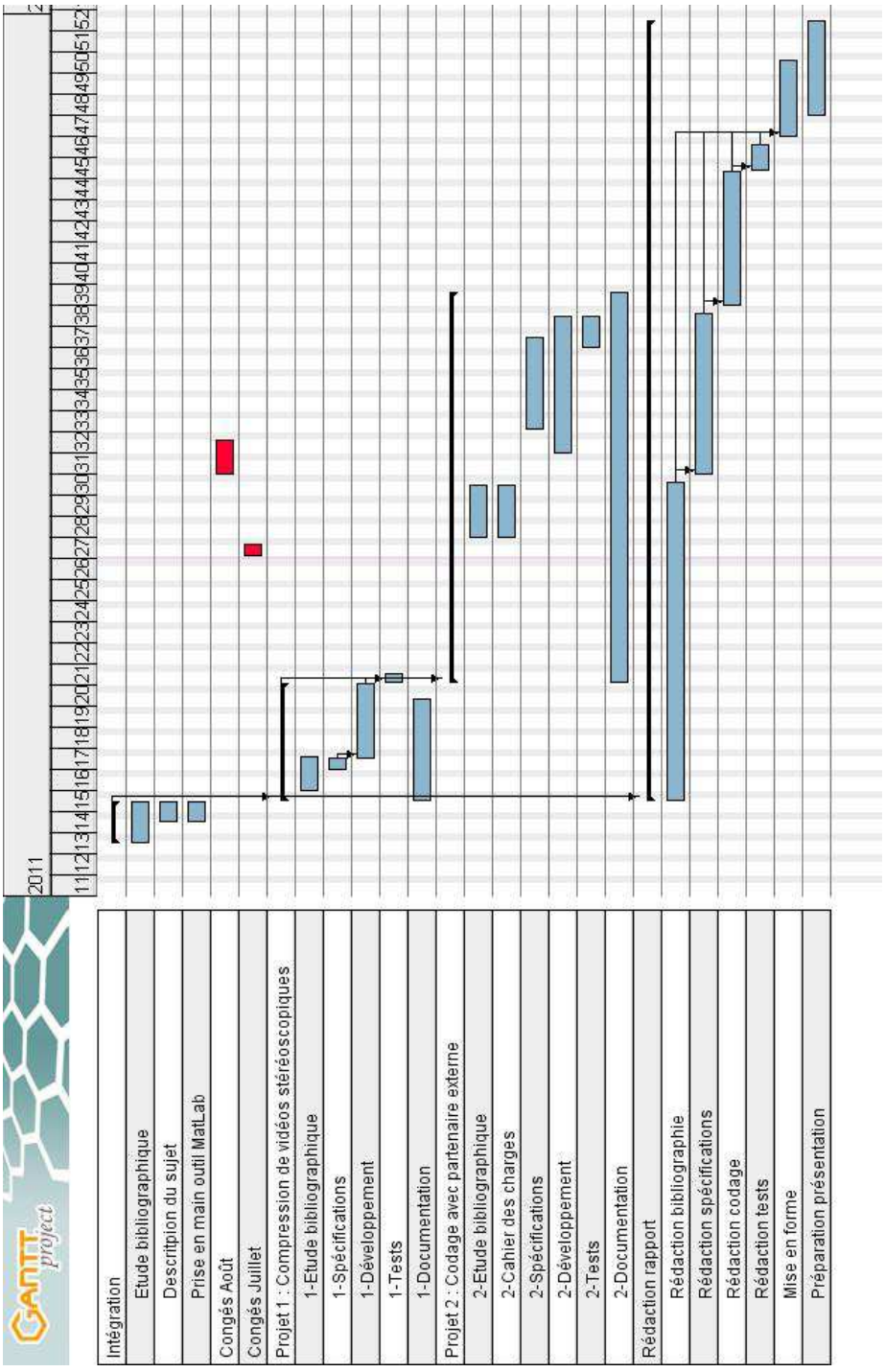

Figure 4.1: Gantt prévisionnel.

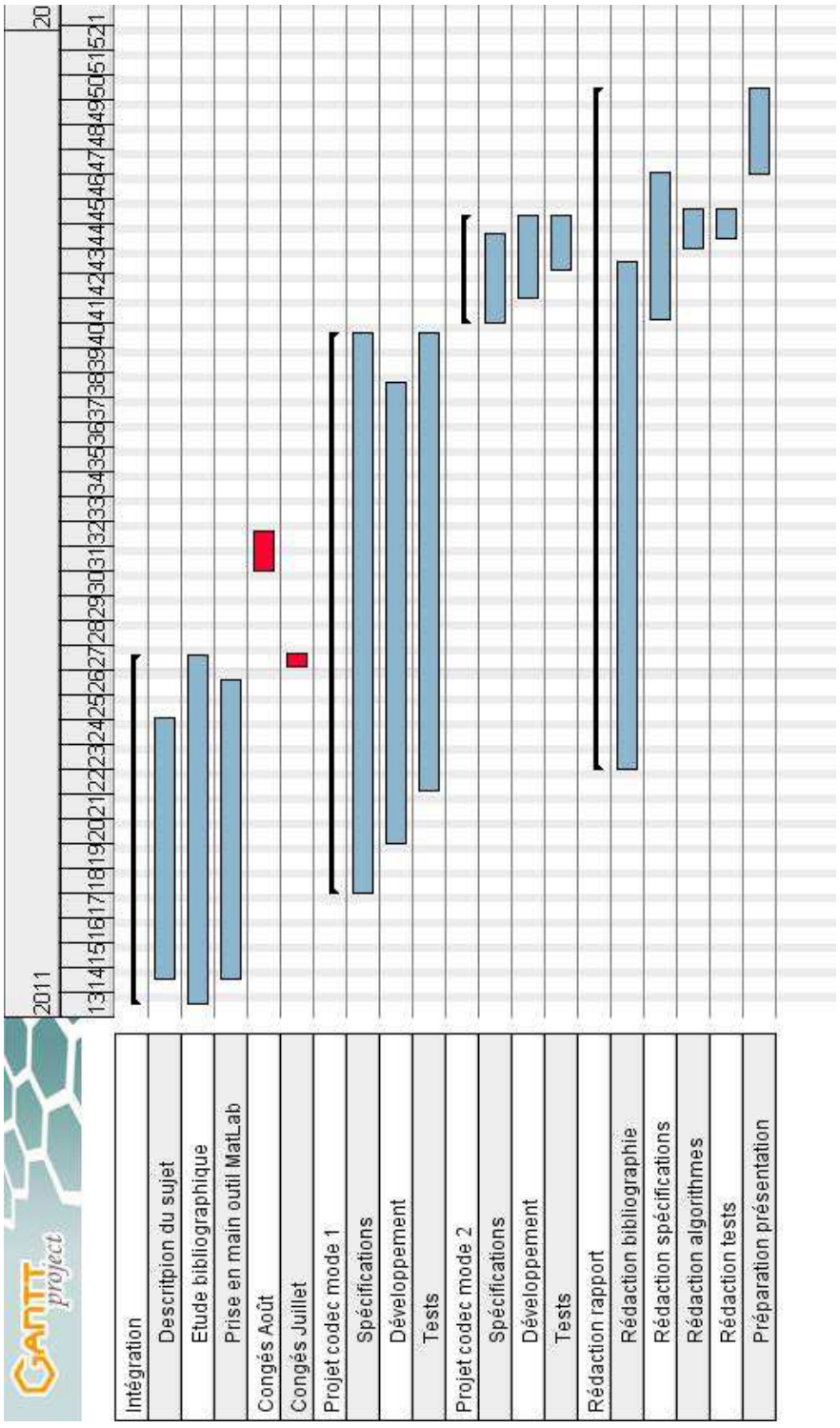

Figure 4.2: Gantt réalisé.

Nous pouvons observer la décision de ne pas coder de suite avec un partenaire externe mais plutôt d'améliorer l'algorithme existant. Ce changement d'orientation n'est pas une remise en question de l'ensemble du projet, mais une décision prise en fonction des résultats obtenus durant les étapes de développement. Cette "itération" est nécessaire pour pouvoir envisager une nouvelle étape d'application de l'algorithme à un projet de "recherche fondamentale" avant d'envisager sont utilisation à plus grande échelle. Cependant, la modélisation des algorithmes et les résultats obtenus seront utilisés pour une contribution à la rédaction d'un article scientifique.

4.2 Environnement technique

L'environnement technique est celui de l'équipe Images et Vidéocommunications (IVC) de L'Institut de recherche en communications et cybernétique de Nantes (IRCCyN).

Pour la réalisation du projet, il m'a été confié une station de travail graphique de type *DELL XPS* équipée d'un écran *DELL ALIENWARE* avec une carte *Nvidia Quadro4000* compatible avec l'affichage de la 3D.

Les prototypes ont été développés en utilisant le logiciel *MATLAB*. Ce logiciel est un puissant outil de calcul numérique, de programmation et de visualisation graphique. Il met à la disposition un environnement performant pour mener à bien des manipulations de matrices [53].

Si le projet doit aboutir sur une application de production, il sera nécessaire de le coder dans un autre langage plus rapide et plus convivial comme le langage C.

Le kit *NVidia3D* sera utilisé pour le visionnage stéréoscopique du résultat des développements sur la station de travail. Ce kit utilise des lunettes 3D actives pour le visionnage [20].

La rédaction du mémoire a été faite au format LaTeXet la présentation pour la soutenance sera au format Powerpoint.

Grâce au statut de stagiaire du laboratoire, il m'a été possible d'accéder à la Bibliothèque Universitaire du site de la Chantrerie pour mes recherches bibliographiques.

Le laboratoire dispose d'un accès internet à haut débit grâce au réseau Réseau National de Télécommunications pour la Technologie, l'Enseignement et la Recherche (RENATER) .

Chapitre 5

Conclusion et perspectives

Ce travail de mémoire a traité d'une nouvelle méthode pour coder des vidéos stéréo. La démarche exposée a montré qu'une approche exploitant la fusion binoculaire apporte de nouvelles pistes. La méthode proposée a permis de quantifier vectoriellement des images binoculaires au travers une approche exploitant deux dictionnaires aux couleurs complémentaires pouvant fusionner dans l'espace perçu 3D. Les dictionnaires ont été construis par adaptation de l'algorithme *LBG* qui est une référence dans le domaine de la QV. La combinaison des couleurs a été faite dans l'espace LUV avec de nombreuses hypothèses simplificatrices (disparité nulle, pas de contraintes pour la combinaison des couleurs). Les résultats obtenus sont prometteurs mais n'ont pas pour le moment permis d'établir définitivement la supériorité de la méthode.

La réalisation d'un codec "mode 1" pour prototyper a été une étape essentielle du travail de mémoire. Ce n'est qu'après une bonne compréhension du fonctionnement de l'algorithme sur des séquences réduites en taille de vecteurs qu'il a été possible de réaliser un codec "mode 2" traitant des vidéos réelles. Cependant, ce deuxième travail a mis en évidence des problèmes spécifiques liés notamment aux différentes transformations des images couleurs dans les différents espaces. De nouvelles pistes sont encore à étudier pour améliorer l'efficacité de l'algorithme de création des dictionnaires. Avec le codec "mode 2" nous avons aussi mis en œuvre deux méthodes de codage adaptées pour les séquences stéréo : une approche *Side by Side* (où les images gauche et droite sont codées séparément), et une méthode par fusion (où elles sont combinées). Enfin il a aussi fallu pour chacune des deux méthodes trouver des approches ad'hoc pour évaluer les erreurs de codage introduites.

A ce stade de l'algorithme, ce travail a donc permis de générer des dictionnaires complémentaires dans l'espace LUV (pouvant fusionner pour représenter les couleurs de l'image stéréo). Cependant, cette étude ne prend pas en compte d'autres problématiques perceptuelles telles que les limites de la fusion binoculaire. Le développement et l'intégration d'un

tel modèle doit pouvoir enrichir le processus de création des dictionnaires composés par l'algorithme *LBG stéréo*. De tels modèles donnent déjà lieu à des travaux de recherche au sein de l'équipe IVC.

Sur le plan personnel, cette expérience a été riche en découvertes, aussi bien techniquement, humainement, que sur la prise de conscience du travail de R&D d'un laboratoire universitaire. De plus, les résultats obtenus pourront être utilisés pour contribuer à une publication scientifique, cette perspective contribue à rendre ce travail encore plus intéressant.

Bibliographie

[1] JM Alliot. Perception de la couleur et colorimétrie - http ://www.photo-lovers.org/color.shtml.fr, 2011.

[2] C Auguste. L'anaglyphe, un procédé à la portée de tous ? - http ://www.photo-stereo.com/anaglyphes-photographie-relief.html, 2009.

[3] P Bas. *Compression d'Images Fixes et de Séquences Vidéo - http ://www.univ-sba.dz/rcam/liens/docs/courCom.pdf.* Grenoble, 2005.

[4] K Beck, M Beedle, A Van Bennekum, A Cockburn, W Cunningham, M Fowler, J Grenning, J Highsmith, A Hunt, R Jeffries, J Kern, B Marick, R Martin, C Mellor, S Schwaber, KJ Sutherland, and D Thomas. Manifeste pour le développement Agile de logiciels - http ://agilemanifesto.org/iso/fr/manifesto.html, 2001.

[5] F Benhamou. Présentation IRCCyN - http ://www.atlanstic.net/presentation/irccyn.html, 2011.

[6] R BRETON. Couleurs en liberté - http ://pages.infinit.net/graxx/CIE2.html, 2009.

[7] Brill, Hemmendinger, and Fairman. *How the CIE-1931 color matching functions were derived from Wright-Guild Data.* New Jersey, 1997.

[8] JC Burie, JM Ogier, and R Raveaux. *Sélection de composantes couleurs ou comment*

trouver un espace couleur discriminant ? Laboratoire L3I Université de La Rochelle, 2008.

[9] CE Carrier. La 3D, partie 1 : les yeux - http ://fillion.ca/blogue/nos-conseils/la-3d-partie-1-les-yeux, 2008.

[10] R Caubet. *FORMATION DOCTORALE EN INFORMATIQUE UNIVERSITE PAUL SABATIER DEA INFORMATIQUE de L'IMAGE et du LANGAGE (IIL) - www.irit.fr/˜Julien.Pinquier/Docs/dea_pinquier.doc.* Institut de Recherche en Informatique de Toulouse, Toulouse, 2001.

[11] C Charrier, A Saadane, and C Larabi. *Evaluation de la qualité : Acteurs, principe, approches : subjective et objective, application à la compression d'images - http ://www-lagis.univ-lille1.fr/ehinc2005/presentations/saadane_chaker_larabi_qualite.pdf.* EHINC, Lille, 2005.

[12] H Cherifi and C Charrier. *Optimisation du dictionnaire pour la quantification vectorielle d'images couleur.* Colloque GRETSI, GRENOBLE, 1997.

[13] Chris. Historique de la 3D - http ://www.web-libre.org/breves/cinema,18646.html, 2011.

[14] M Colinet. Concevoir un produit multimédia avec des objets particuliers - http ://www.det.fundp.ac.be/cefis/publications/monique/image-son-video-5-74.pdf, 2001.

[15] P Courtellemont. *Traitement des images couleur, Présentation DEA Image et calculs.* Laboratoire L3I Université de La Rochelle, 2001.

[16] Dominguez. *Recherche sur le relief interactif en temps réel : " perception ", horloge en relief, Master A.T.I Arts et Technologies de l'Image.* Paris 8 Vincennes-Saint Denis, 2008.

[17] PA Dorange. Comprendre le codage couleur YUV - http ://www.garage-video.com/spec/YUV.html, 2001.

[18] M Faidley, D Shasha, and Poultney C. Enseignement / Quantification vectorielle - http ://interstices.info/jcms/c_24839/jouez-avec-les-diagrammes-de-voronoi, 2007.

[19] C. Fernandez-Maloigne, P. Bonton, and A. Trémeau. *IMAGE NUMÉRIQUE COULEUR : De l'acquisition au traitement*. DUNOD, 2004.

[20] S Gateau. 3D Vision Technology Develop, Design, Play in 3D Stereo - http ://www.slideshare.net/NVIDIA/3d-vision-technology-develop-design-play-in-3d-stereo, 2009.

[21] P Hsiao Hui. *VRML : Etude, mise en oeuvre et applications*. Cnam Paris, 2001.

[22] P Hsiao Hui. *The Evolution and prospect of 3D Display technology, Master in Business Design Academic*. Paris 8 Vincennes-Saint Denis, 2009.

[23] Intel. Image Color Conversion - http ://software.intel.com/sites/products/documentation/hpc/ipp/ippi/ippi_ch6/ch6_color_models.html.

[24] P Jadoul. La Vision - http ://users.swing.be/Ph.Jadoul/home%20%20files/Vision/Vision.htm.

[25] G Marcellier. Ecrans 3D Alioscopy - http ://www.alioscopy.eu/fr/ecrans3D.php, 2011.

[26] G Marcellier. Le relief à l'oeil nu ! - http ://www.alioscopy.eu/fr/accueil.php, 2011.

[27] A Martin. *De la sous-traitance à la coopération : la gestion de la R&D dans les entreprises françaises - http ://lem.cnrs.fr/portals/2/actus/DP_200714.pdf*. Lille, 2007.

[28] D Metz. Blog couleur - http ://www.blog-couleur.com/ ?Qu-est-ce-que-la-luminance, 2008.

[29] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *The Not So Short Introduction to LATEX 2 - http ://tobi.oetiker.ch/lshort/lshort-a5book.pdf.* Citeseer, 2011.

[30] OPI. Système colorimétrique RGB et CIE XYZ - http ://www.optique-ingenieur.org/fr/cours/OPI_fr_M07_C02/co/Contenu_07.html, 2011.

[31] S Pakin. *The Comprehensive LaTeX Symbol List - http ://tobi.oetiker.ch/lshort/lshort.pdf.* 2011.

[32] S Paris. *Le multimédia et la compression.* Lavoisier, 2009.

[33] C Petit-Jean. Enseignement / Quantification vectorielle - http ://carolinepetit-jean.free.fr/enseignements/quantif_vect.php, 2011.

[34] C Poynton. Colour FAQ, 2009.

[35] IEG Richardson. *H.264 and MPEG-4 video compression : video coding for Next-generation multimedia.* John Wiley & Sons, 2003.

[36] V Ricordel. *Etudes de schémas de quantification vectorielle algébrique et arborescente. Application à la compression de séquences d'images numériques.* PhD thesis, Rennes1, 1996.

[37] V Ricordel. Présentation IVC - http ://www.irccyn.ec-nantes.fr/spip.php ?article324, 2011.

[38] V Ricordel. Présentation projet PERSEE - http ://persee.irccyn.ec-nantes.fr/, 2011.

[39] V Ricordel, C Labit, and P Fiche. *Etudes d'algorithmes de quantification vectorielle arborescente pour la compression d'images fixes.* Rapport de Recherche INRIA, N 2241, 1995.

[40] D Romeuf. Physiologie sensorielle, oeil-cerveau-couleurs - http ://www.david-romeuf.fr/3D/Anaglyphes/BonCoupleEL/BonCoupleEcranLunettesAnaglyphe.html, 2008.

[41] D Scharstein. Middlebury Stereo Vision Page - http ://vision.middlebury.edu/stereo/, 2009.

[42] Infoclick Solution informatique. Le codage de la couleur - http ://www.infoclick.fr/ccm/video/couleur.htm#RGB, 2004.

[43] Webexpert. L'oeil Humain - http ://membres.multimania.fr/imgnum/theorie/1.html, 1999.

[44] Wikipedia. Espace colorimétrique CIE LUV - http ://fr.wikipedia.org/wiki/CIE_LUV.

[45] Wikipedia. Notions sur la vision des couleurs - http ://fr.wikibooks.org/wiki/Photographie/Photom%25C3%25A9trie/Notions_sur_la_vision_d 2010.

[46] Wikipedia. CIE RVB - http ://fr.wikipedia.org/wiki/CIE_RVB, 2011.

[47] Wikipedia. La Rétine - http ://www.fr.wikipedia.org/wiki/R%C3%A9tine, 2011.

[48] Wikipedia. La stéréoscopie - http ://fr.wikipedia.org/wiki/St%C3%A9r%C3%A9oscopie, 2011.

[49] Wikipedia. Lois de Grassmann - http ://fr.wikipedia.org/wiki/Lois_de_Grassmann, 2011.

[50] Wikipedia. Méthodes agiles - http ://fr.wikipedia.org/wiki/M%25C3%25A9thode_agile, 2011.

[51] Wikipedia. Peak Signal to Noise Ratio - http ://fr.wikipedia.org/wiki/Peak_Signal_to_Noise_Rat 2011.

[52] Wikipedia. Saturation (couleurs) - http ://fr.wikipedia.org/wiki/Saturation_(couleurs), 2011.

[53] R Younes. *Initiation sur MatLab - http ://www.ryounes.net/cours/Initiation.pdf*. Université Libanaise, 2005.

Annexes

1 - Le laboratoire IRCCyN et l'équipe IVC

Ce mémoire a été réalisé au sein du laboratoire IRCCyN [5] qui est une UMR (Unité Mixte de Recherche) du CNRS (Centre National de la Recherche Scientifique), rattachée au département scientifique des Sciences et Technologies de l'Information et de l'ingénierie, et dont les tutelles locales sont l'Ecole Centrale de Nantes, l'Université de Nantes et l'Ecole des Mines de Nantes.

Plus précisément, j'ai intégré l'équipe IVC qui est l'une des équipes de recherche de l'IRCCyN, le laboratoire a pour thématiques :

- L'automatique (équipe Commande)
- Le traitement du signal et des images (équipe ADTSI)
- **La vidéo communication (équipe IVC)**
- La robotique (équipe Robotique)
- La conception mécanique assistée par ordinateur (équipe MCM)
- La modélisation et l'optimisation de processus de production (équipe MO2P)
- L'ingénierie virtuelle pour l'amélioration des performances industrielles (équipe IVGI)
- Les systèmes temps Réel (équipe Temps Réel)

- La modélisation et la versification des systèmes embarques (équipe MoVES)
- Les systèmes logistiques et de production (équipe SLP)
- Les systèmes a évènements discrets (équipe ACSED)
- La psychologie cognitive et l'ergonomie (équipe PsyCoTec)

Le laboratoire rassemblait en janvier 2010 un peu plus de 260 personnes avec 103 chercheurs et enseignants-chercheurs (dont 14 du CNRS), 17 ingénieurs, techniciens et administratifs ainsi que 142 chercheurs non-permanents (dont 111 doctorants, 28 CDD et 3 Postdoc) sans compter de nombreux professeurs invites et divers stagiaires. L'équipe IVC [37] est constituée de huit personnes permanentes, d'ingénieurs, de doctorants, de Postdoc, de chercheurs, d'invités et de stagiaires. Durant la période de stage cela représente environ trente personnes. Son axe de recherche s'articule autour de la perception, communication et Représentation de l'information et se décline en cinq thèmes de recherche :

- Représentation et perception modèles psychovisuels
- Représentation Mojette et géométrie discrète
- Représentation et communication vidéo-multimédia
- Représentation et communication réseaux
- Représentation et écrits et documents

2 - Le projet PERSEE

Le projet Persée est réalisé en partenariat par quatre équipes de recherche :

- IRCCyN - Equipe IVC.
- L'Institut National de Recherche en Information et Automatique (INRIA) - Rennes - Equipe Traitement, modélisation et communication d'images numériques (TEMICS).
- L'Institut d'Electronique et de Télécommunications de Rennes (IETR) - Rennes - Equipe Image
- Le laboratoire traitement et Communication de l'Information (LTCI) - Paris - Groupe Multimédia

Ce projet s'inscrit donc dans un contexte d'émergence :

- De nouveaux formats pour la vidéo 3D (Multiview, Video-plus Depth (MVD))
- De nouveaux standards et codecs (H.264, 3D Vidéo Coding (3DVC))
- De nouveaux formats de vidéo immersive (3D Télévision (3DTV) Free ViewPoint TV (FTV))

Un constat est fait, il faut améliorer la qualité perçue des vidéos codées/décodées, pour cela le projet vise à produire de nouveaux outils et méthodes. Exactement ce projet a pour objectifs :

- D'aller vers une compression basée qualité perceptuelle des contenus 2D et 3D
- De produire des méthodes et outils pour une représentation adaptée au 2D et à la 3D
- L'intégration dans une plateforme logicielle commune de codage 2D et 3D

Dans un contexte plus particulier, c'est la contribution pour le standard en cours 3DVC et l'amélioration de la qualité perçue pour un codage basé qualité perceptuelle.

Table des figures

Conception d'outils avancés pour le codage de la vidéo 3D : Fusion d'images couleur binoculaires perçues

Mémoire d'Ingénieur CNAM, Centre Régional Associé de Pays de Loire

---

RESUME

Pour la compression d'images couleur par la technique de quantification vectorielle, l'objectif est d'obtenir un dictionnaire représentant au mieux les images. Dans le cas de vidéos stéréo, deux images sont projetées pour les deux yeux, mais oblige à avoir un dictionnaire des représentants pour chacune. Une des pistes proposée pour réduire la taille de ces dictionnaires est d'explorer les possibilités de la fusion binoculaire pour créer plus de couleurs fusionnées et des dictionnaires compacts. Pour la création de ces dictionnaires complémentaires (dont les couleurs peuvent fusionner), nous allons modifier l'algorithme *LBG*. La méthode introduite a de nombreuses conséquences : sur le schéma de codage et de décodage des images stéréo, l'évaluation des erreurs, la prise en compte de facteurs perceptuels.

mots clés : Image stéréo, vidéo stéréo, fusion binoculaire, quantification vectorielle, *LBG*, colorimétrie.

ABSTRACT

For the compression of color images by the technique of Vector Quantization, the goal is to compute a dictionary which represents at best the images. In the case of stereo videos, two images are used for both eyes, but it requires to have a representative dictionary for each image. One of the idea introduced in order to reduce the size of these dictionaries is to explore the possibilities of the binocular fusion, it permits to create more merged colors from smaller dictionaries. For the design of complementary dictionaries (whose colors can be merged), we are going to modify the *LBG* algorithm. The new method has many consequences: on the coding and decoding scheme of stereo images, on the error assessments, on the taking into account of perceptual factors.

key words : Stereo image, stereo video, binocular fusion, vector quantization, *LBG*, colorimetry.