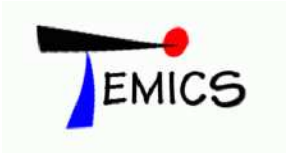
Compression de séquences vidéo et choix des images de références

Rapport de stage de DEA Informatique
Université de Rennes 1 - IFSIC
Juin 2003

Nicolas Dumoulin
Encadré par :
Henri Nicolas



Remerciements

Je tiens à remercier Henri Nicolas, pour m'avoir encadré et guidé durant ce stage.

Je remercie également Fabien Catteau et Érice Morillon, respectivement doctorant et stagiaire dans le laboratoire TEMICS pour leur aide.

Table des matières

Introduction

Ce document a pour objectif de rendre état de mon stage de DEA Informatique, réalisé dans le laboratoire TEMICS (TraitemEnt, Modélisation d'Images et Communications) de l'IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires). Le but de ce stage est de réaliser des expérimentations sur le choix des images de références dans le processus de compression de séquences vidéo.

La compression des séquences vidéo est nécessaire pour en permettre le stockage de manière économique, ainsi que pour pouvoir transmettre les données vidéos numériques à travers un réseau à bande passante limitée ou depuis un média ayant une limite de taux de transfert.

Au cours des dernières années, l'intérêt pour le multimédia, et en particulier pour la diffusion de contenus audiovisuel, a entraîné de nombreuses recherches dans le domaine du codage du signal vidéo, qui ont abouti à plusieurs normes (H.263 [1], MPEG-4 [2] et bientôt H.26L [3] pour ne citer que les plus récents). Ces normes consistent finalement en des boîtes à outils de traitement du signal vidéo, qui permettent d'adapter le traitement apporté en fonction du contexte et du résultat souhaité (rapport débit/distorsion). Les dernières recherches visent donc à améliorer ces outils, et à en apporter de nouveaux.

Ce rapport présentera tout d'abord les techniques utilisées dans le processus de compression vidéo. Ensuite, nous verrons les logiciels de traitement des séquences vidéos utilisés durant ce stage. Enfin, j'expliquerai la manière dont ont été menées nos expériences, et leur résultats.

Chapitre 1

La compression de séquences vidéo

1.1 La représentation d'une séquence vidéo

Une séquence vidéo brute est une suite d'images fixes, qui peut être caractérisée par trois principaux paramètres : sa résolution en luminance, sa résolution spatiale et sa résolution temporelle.

La résolution en luminance détermine le nombre de nuances ou de couleurs possibles pour un pixel. Celle-ci est généralement de 8 bits pour les niveaux de gris et de 24 bits pour les séquences en couleurs. La résolution spatiale, quant à elle, définit le nombre de lignes et de colonnes de la matrice de pixels. Enfin, la résolution temporelle est le nombre d'images par seconde.

La valeur de ces trois paramètres détermine l'espace mémoire nécessaire pour stocker chaque image de la séquence. Cet espace mémoire est caractérisé par le débit, qui est le coût de stockage pour une seconde (capacité mémoire nécessaire pour stocker une seconde de vidéo). Par exemple, une séquence ayant une résolution de 720 par 576 pixels, un codage des couleurs sur 24 bits, et une fréquence de 25 images par seconde, nécessitera un débit de 137 Mb/s (mégabits par seconde). Le débit d'une séquence vidéo brute est très élevé comparé aux débits et à l'espace offerts par les moyens de stockage et de transferts actuels.

1.1.1 Le format YUV 4 :1 :1

Dans un premier temps, une solution pour réduire ce débit est de sous-échantillonner les composantes de chrominance de la séquence. L'œil humain étant plus sensible aux variations de luminance que de chrominance, on uti-

lise l'espace de représentation de couleurs YUV, en sous-échantillonnant les composantes U et V.

Historiquement et techniquement, le format YUV est utilisé pour la transmission des signaux vidéos analogiques couleurs. Dans ce sigle, Y représente la luminance, U et V les composantes Rouge et Bleu (chrominances). Ces 3 informations permettent de restituer au final les composantes RGB [1] et apporte l'avantage de permettre une compression facile des couleurs et de fiabiliser le transport du signal vidéo. Mais surtout la partie Y (luminescence) permet une parfaite compatibilité avec les anciens équipements en noir et blanc, ce qui a permis le déploiement de la diffusion couleur alors que la totalité des foyers étaient alors équipés de téléviseurs noir et blanc.

Dans le format YUV 4 :1 :1, les composantes de chrominance sont réduits à la moitié de la résolution verticale et horizontale de la composante de luminance, soit 4 composantes de chrominance pour une composante U, et une composante V (voir figure 1.1).

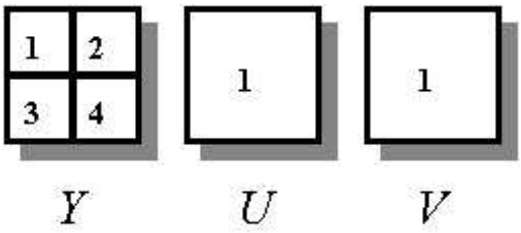


FIG. 1.1 – Le format YUV 4 :1 :1.

Ce sous-échantillonnage correspond à une réduction de la redondance psycho-visuelle. Dans la suite, nous étudierons les techniques permettant de réduire les redondances spatiales et temporelles.

1.2 La compression de données

Dans cette section nous allons nous intéresser aux techniques utilisées pour compresser le signal vidéo en utilisant la redondance spatiale. Le but est de réduire le débit de la séquence vidéo à compresser, tout en minimisant les erreurs visibles (voir section 1.3.2 page 14).

La compression de données permet de réduire la quantité d'information, en modifiant son mode de représentation. Pour cela, il existe deux principales techniques, la compression sans pertes et la compression avec pertes.

[1] RGB est l'abréviation des couleurs de base (*Red-Green-Blue* soit Rouge-Vert-Bleu) constituant le codage des pixels utilisé en imagerie.

(a) originale (786 ko)

(b) compression JPEG facteur 88 (56 ko)

(c) compression JPEG facteur 14 (9 ko)

FIG. 1.2 – Introduction des pertes dans le processus de compression.

La première permet de retrouver l'information initiale après décompression, tandis que la deuxième n'en restituera qu'une approximation. Dans le cas de la compression d'images naturelles (par opposition aux images de synthèse), la compression sans pertes est insuffisante, et l'introduction de pertes dans le processus de compression permet d'obtenir de meilleurs résultats sans empêcher l'interprétation du contenu visuel (voir figure 1.2).

Les normes vidéo actuelles utilisent un système de codage hybride avec compensation du mouvement basé sur des blocs et réduction de l'entropie par transformée (voir figure 1.3 page suivante), que nous allons étudier maintenant.

1.2.1 La réduction de l'entropie

L'entropie ($H(X)$) d'une image est la quantité moyenne d'informations apportée par chaque nouveau pixel transmis, ou encore, l'incertitude moyenne sur le prochain pixel qui va être traité [4].

$$H(X) = \sum_{i=1}^{N} -p_i log_2(p_i)$$

L'entropie sera maximale si la source obéit à une loi uniforme, alors qu'elle sera plus faible si la distribution est concentrée autour de quelques valeurs seulement. Le but de la réduction de l'entropie est par conséquent de transformer la distribution des valeurs de l'information à transmettre.

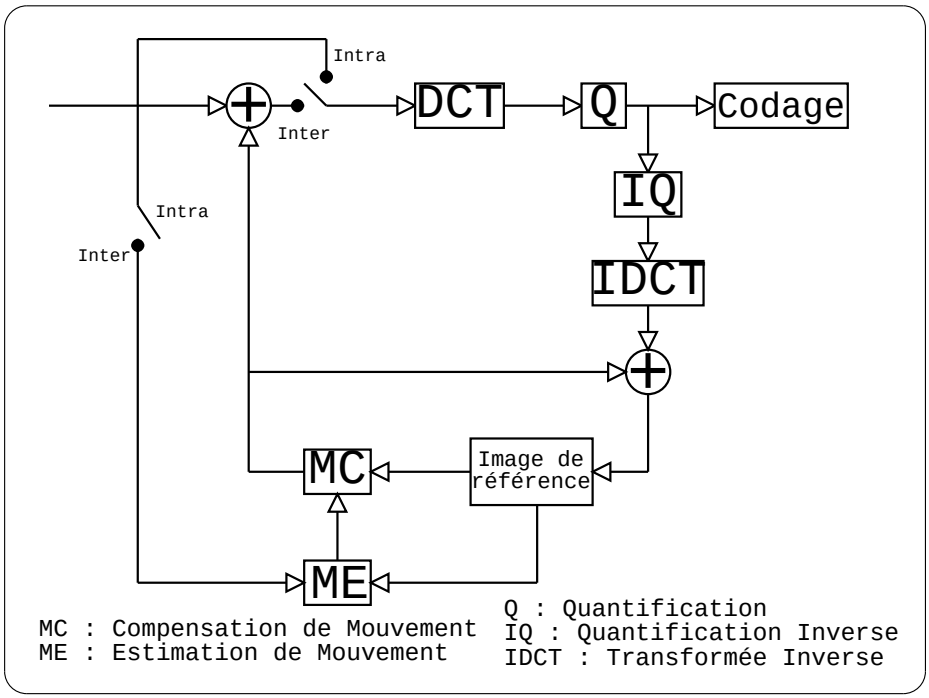


FIG. 1.3 – Schéma général d'un codeur vidéo.

La réduction de l'entropie est composée de trois étapes, que sont la décorrélation, la quantification, et le codage entropique. La décorrélation est opérée par un codage par transformée. Parmi les transformations utilisées pour la compression d'images, on peut citer la transformation de Kahrunen-Loève (KLT), la transformée en cosinus discrète (DCT) et les transformées en ondelettes discrètes (DWT).

La KLT est la transformée optimale [5], mais comporte l'inconvénient d'être dépendante des données, puisque composée des vecteurs propres de la matrice de covariance de l'image. Cette dépendance induit un coût de calcul élevé, que ne présente pas les deux autres transformées.

Dans le cas des images naturelles, les pixels voisins présentent une forte corrélation, de ce fait, les capacités de la DCT sont très proches de celle de la KLT [5] tout en réduisant la complexité de calcul.

La DWT a l'avantage de produire une matrice contenant beaucoup de zéros, ce qui a la propriété d'alléger les calculs lors d'opérations matricielles. Elle peut donc être appliquée à l'image entière, contrairement à la DCT qui est appliquée par blocs pour limiter les calculs.

L'application d'une transformée sur une image naturelle permet d'obtenir des composantes dont la variance est concentrée sur les composantes de basses

fréquences.

L'image ainsi transformée est ensuite quantifiée puis compressée à l'aide d'un codeur entropique.

1.3 Codage vidéo par estimation et compensation de mouvements

La technique que nous allons maintenant étudier permet d'exploiter les redondances temporelles, fortement présentes dans une séquence vidéo naturelle. La compression par compensation de mouvement consiste pour une image, à ne coder que les erreurs résultantes de l'estimation de mouvement faite à partir d'une image de référence.

L'estimation de mouvement est faite par mise en correspondance de blocs des images (ou *bloc matching*), afin d'obtenir les vecteurs de mouvement correspondant. Cette estimation n'est qu'une approximation, car la mise en correspondance n'est pas complète. Les erreurs résiduelles, qui consistent en la différence entre l'image prédite et l'image compensée, sont donc ajoutées aux vecteurs de mouvement. Cette compensation et ces erreurs sont ensuite transformées, quantifiées et codées, avant d'être transmises. En plus d'être codées et transmises, elles subissent également une quantification inverse et une transformation inverse.

Le but de décoder les images à l'intérieur du processus d'encodage, est d'effectuer les prédictions à partir des images décodées, afin de travailler avec les mêmes informations que le décodeur cible.

1.3.1 La structuration du flux vidéo

Ce codage prédictif rend les images interdépendantes, puisque lors du décodage, il est nécessaire d'avoir l'image précédente pour décoder l'image en cours. Cette dépendance est gênante si l'on souhaite naviguer de manière aléatoire, le décodage de tout le début de la séquence étant nécessaire, ou si une image est perdue lors d'un transfert, le décodage du reste de la séquence étant impossible.

C'est pourquoi le processus de compression comprend deux modes : le mode *intra* et le mode *inter*. Le mode *intra* correspond à une compression spatiale sans compensation de mouvement (transformée, quantification et codage), tandis que le mode *inter* utilise la compression temporelle. Les commutateurs présents dans la figure 1.3 page ci-contre représentent le choix du mode de codage.

Les images sont regroupées en GOP (*Group Of Pictures*) composés de trois types d'image. Les images «I» sont codées en mode *intra*, et sont donc indépendantes mais demandent plus de ressources. Les images «P» sont codées par compensation de mouvement à partir de la dernière image I ou P. Les images «B» sont prédites bidirectionnellement à partir d'images I ou P.

Un GOP débute toujours par une image I, qui constitue l'image-clé du GOP. Cette image, est ensuite suivie d'image P, intercalées par des image de type B (voir figure 1.4). La fréquence des images de type P, et la taille du GOP sont des paramètres qui influerons sur le taux de compression, et sur le temps de décodage. En effet, plus le GOP sera long, et plus la dépendance entre les images sera forte, mais moins il y aura d'images I qui freine sensiblement le taux de compression. L'introduction des images induit un délai de décodage additionnel, puisque les images P postérieures doivent être décodées avant que les images B antérieures puissent être décodées.

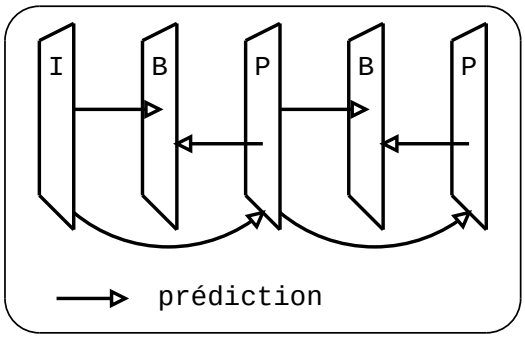


FIG. 1.4 – Exemple d'agencement d'un GOP.

Le choix du mode de codage peut être fait sur les images entières (déterminé par la structure de GOP utilisé) ou sur des blocs d'images. En effet, lors du codage d'un bloc d'une image, le processus de compensation de mouvement peut s'avérer inefficace, dans ce cas, le bloc est codé en utilisant le mode *inter*.

1.3.2 Appréciation des erreurs de compression

La quantification est l'opération qui introduit les pertes. Ces pertes sont mesurées par l'erreur quadratique moyenne (EQM) qui est calculée à partir de l'image originale et de l'image reconstruite.

$$\mathrm{EQM} = \frac{1}{\mathrm{N}} \sum_{i=1}^{N} (x_i - \overline{x_i})^2$$

De cette EQM, on dérive une autre grandeur, plus significative, appelée *peak signal to noise ratio* (PSNR). Cette grandeur est un gain qui s'exprime en décibels, et s'obtient ainsi :

$$\mathrm{PSNR} = 10 \log_{10} \frac{255^2}{\mathrm{EQM}}$$

Cette mesure de la distorsion couplée avec le débit de la séquence résultante permet d'apprécier le compromis entre le gain de la compression et la qualité de restitution. Ce compromis est visible sur la courbe de la figure 1.5. Le but étant, en général, d'obtenir la meilleure fidélité (ou la plus grande distorsion) compte tenu de la capacité du canal de transmission, qui détermine la contrainte de débit. Cette optimisation peut être faite à l'aide des techniques de minimisation de Lagrange fondées sur la théorie du débit-distorsion [6].

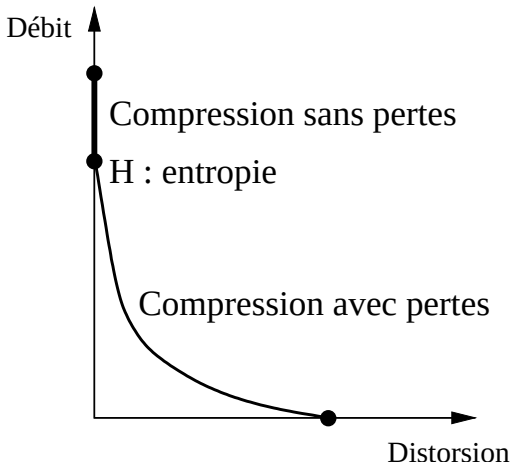


FIG. 1.5 – Allure typique d'une courbe du rapport débit / distorsion.

1.4 Les images de références

On apelle *image de référence*, l'image qui va être utilisée pour prédire une image et estimer le mouvement à compenser. Cette image peut être simplement une image précédemment codée dans la séquence. Dans les dernières normes, de nouvelles images sont utilisées : les objets mosaïques.

Un objet mosaïque contient l'ensemble de l'information disponible sur un objet physique donné. Il est souvent utilisé sous la forme d'objet mosaïque plan (voir figure 1.6 page suivante) appelé MOP (*Mosaic Object Plane*) qui est

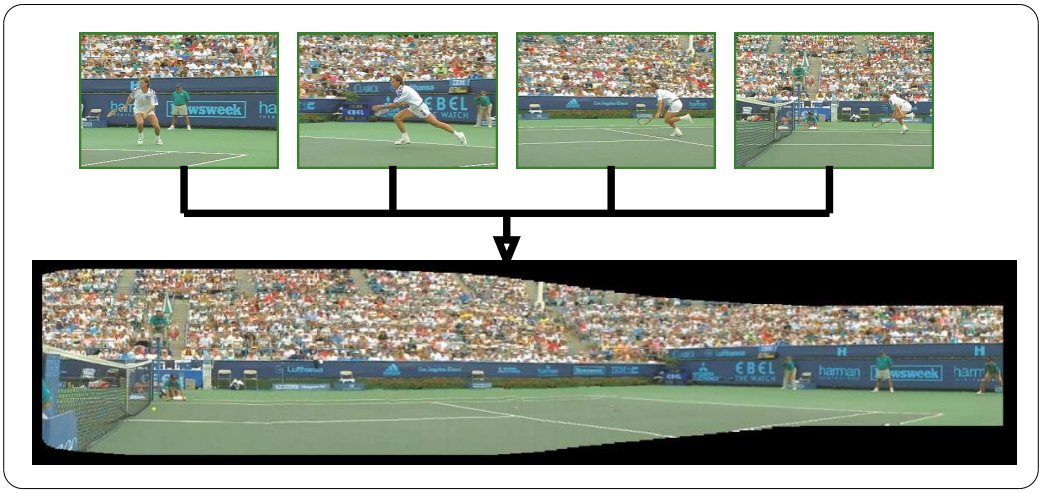


FIG. 1.6 – Exemple d'image mosaïque construite à partir d'un ensemble d'images.

construit à partir d'une séquence de vues de l'objet considéré. Le principe de construction consiste à recaler les images (ou portions d'images) traitées dans le référentiel de l'objet mosaïque [7]. Ces images peuvent être utilisées pour représenter sur une seule image la totalité d'une scène [8]. Cette mosaïque est généralement plus grande grande qu'une image de la séquence.

Le standard MPEG-4 [2] a ajouté le mode de codage *sprite* aux modes *intra* et *inter*, pour construire de tels objets. En revanche, rien n'est spécifié dans ce standard en ce qui concerne le processus de construction.

1.5 les normes de compression existantes

Les dernières normes actuellement développées sont le H.263 [1] et le MPEG-4 [2]. La première de ces normes a été créée par le *Video Coding Expert Group* (groupe fondé en 1997 par l'ITU-T, l'*International Telecom Union*). La seconde a été spécifiée en 1999 par le *Moving Picture Expert Group*, groupe créé en 1988 par l'ISO/IEC JTC.

MPEG-4 a été le premier à apporter la notion d'objet vidéo, et donc celle d'objet mosaïque, tout en reprenant les fonctionnalités de MPEG-2. Le standard H.263 est surtout dédié aux applications bas débits.

H.26L (voir section 2.1 page 19), qui devrait être très prochainement normalisé proposera une large gamme de résolution, et améliorera les précédents codeurs en conservant le même schéma de compression.

1.6 Problématique du stage

Classiquement, l'image de référence utilisée lors de l'encodage d'une image prédite, n'est pas choisie, c'est l'image de type P ou I précédente qui est utilisée. Or, dans certains cas, le choix d'une image situé plus loin dans la séquence semblerait plus approprié.

Prenons comme premier exemple le cas d'un changement de plan répétitif. Si un changement de plan survient entre deux images prédites, la prédiction de la première image du second plan sera faite à partir d'une image du plan précédent, alors qu'il semblerait plus intéressant d'utiliser une image antérieure du même plan (si une telle image existe). Ce problème s'illustre parfaitement sur la figure 1.7, où l'image 22 fait référence à une image du plan B, alors que des images du plan A ont déjà été codées ultérieurement.

Le deuxiéme exemple concerne l'entrée d'un objet dans une scène. Dans ce cas, il semblerait intéressant de choisir comme image de référence l'image où l'objet est complètement rentré dans le champ de vision. Ainsi, les prédictions seraient plus efficaces pour les parties visibles de l'objet pour chaque image où il apparaît.

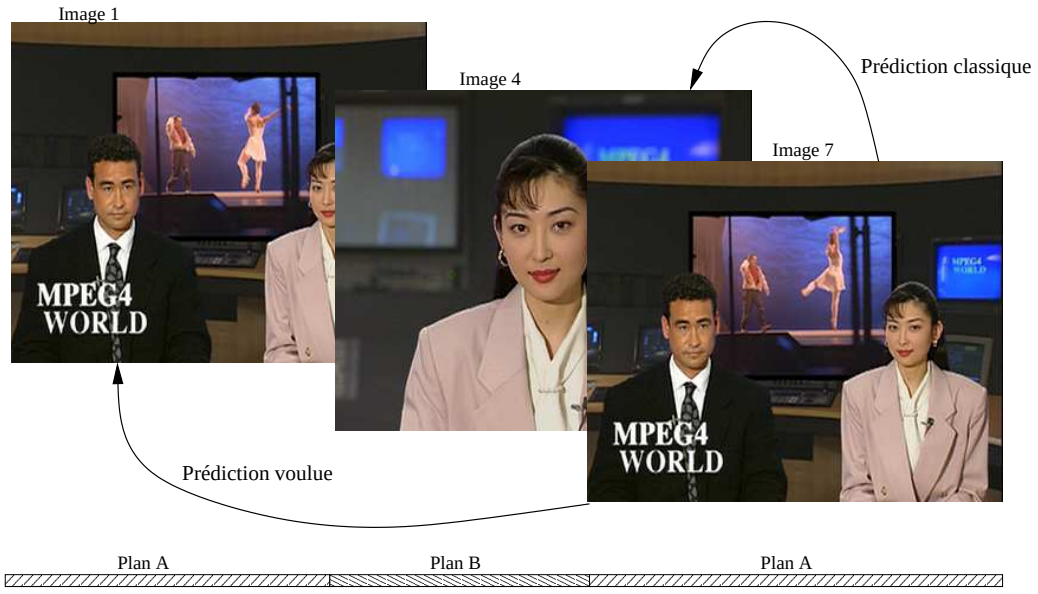


FIG. 1.7 – Prédiction lors d'un changement de plan.

Le but de ce stage est donc de faire des expériences sur l'utilisation d'images de références pertinentes, en essayant de trouver des critères pour déterminer les images. à utiliser comme images de références.

Chapitre 2

La manipulation des séquences vidéo

2.1 Le format H.26L

2.1.1 Généralités

Le format H26L est issu d'une spécification commune du VCEG (*Video Coding Expert Group*) et du groupe MPEG (*Moving Picture Expert Group*), regroupés en une équipe (*Joint Video Team*) créée en 2002. Ce regroupement a pour but de regrouper les efforts fournis par ces deux groupements d'experts afin d'aboutir à de nouveaux standards plus performants, en réunissant les travaux antérieurs de chacun d'eux. H26L vient donc remplacer deux projets initiés par les deux parties, qui sont *MPEG-4 Part 10 (Advanced Video Coding, AVC)* pour le groupe MPEG et H264 pour le VCEG. L'élaboration du standard H.26L a déjà abouti à la rédaction d'un brouillon final [3], et au développement d'un logiciel de référence : le *Joint Model* [9].

Cette norme vise aussi bien les applications temps réel (visioconférence) que les applications de stockage. Les principales caractéristiques [3, 10, 11] de ce standard émergeant sont l'utilisation de macroblocs de taille variable (de 4x4 à 16x16), la gestion des paquets perdus et des erreurs de bits, l'adaptation au débit disponible d'un réseau (*Network Adaptation Layer*), la possibilité d'utiliser un codeur arithmétique (*CABAC : Context-Based Adaptative Binary Arithmetic Coding*) et l'utilisation de plusieurs images de références pour les images de type P et B.

L'adaptation aux conditions du réseau est permis grâce à l'introduction d'un nouveau type d'image, le type SP. Ces images qui viennent remplacer des images P, permettent la transition entre deux flux différents (qui peuvent être la même séquence codée à différents débits).

Le codage en mode *inter* peut utiliser jusqu'à 5 images de références, ce qui augmente la qualité visuelle du rendu à débit constant [11]. En revanche, cela nécessite plus de calcul et de mémoire pour le codage et le decodage.

2.1.2 Implémentations existantes

Le JVT a développé un logiciel de référence afin d'expérimenter et de valider les fonctionnalités de H26L. Ce logiciel se nomme JM (*Joint Model*), et en est à sa version 6.1, dont les sources sont librement téléchargeables sur la page web des logiciels du JVT [9]. Ce logiciel est fonctionnel, mais a reçu de nombreuses critiques quand à la lisibilité et la réutilisabilité de ses sources [12]. L'illisibilité du code vient du fait qu'il est entièrement écrit en langage C orienté efficacité (peu de fonctions, réutilisation de variables, ...), et que la plupart des données utilisées sont stockées dans des variables globales. De plus aucune documentation n'est fournie sur la structuration du modèle et sur la signification des variables utlisées ce qui complique sa compréhension.

C'est pour cette raison qu'un projet alternatif a été initié par quelques développeurs, sous le nom de Hdot264. Ce projet est hébergé sur le site de *Sourceforge* [13], et ne propose pas encore d'implémentation fonctionnelle à ce jour. Hdot264 a pour objectifs de proposer une implémentation de base facilement réutilisable (conception par objets), et également plus performante en terme de rapidité d'exécution. En effet, le logiciel du JVT se révèle être très lent à l'exécution (en moyenne une dizaine de secondes par image au format QCIF : 352x288, sur un PC à 1.8GHz).

2.2 Les outils utilisés

Dans cette section, nous allons survoler les outils applicatifs qui ont servi à la manipulation des séquences, et au traitement des résultats. L'utilisation d'interpréteurs comme Bash, Gawk et Gnuplot, à permis de réaliser des scripts automatisant les expériences menées ainsi que le formatage des résultats (génération de courbes).

seqview

Ce logiciel permet de visualiser de maniere interactive des séquences de formats variés. Parmis ceux-ci, on retrouve le YUV et la séquence d'images non compressées ppm, mais également d'autres formats compressés spécifiques aux travaux du laboratoire.

yuv2ppm et ppm2yuv

La première de ces deux applications (yuv2ppm) permet d'extraire une (ou des) image(s) d'une séquence vidéo. ppm2yuv permet l'opération inverse, c'est-à-dire reconstruire une séquence à partir de plusieurs images. Ces deux outils sont notament pratiques pour éditer une séquence.

BASH : Bourne Again SHell

Bash est un interpréteur de langage de commande compatible avec SH, qui exécute des commandes lues depuis l'entrée standard, ou depuis un fichier.

gawk

Gawk est un interpréteur qui permet de filtrer un flux ASCII, pour en extraire des motifs, et d'effectuer des opérations complexes sur les chaines de caracteres.

gnuplot

Gnuplot est un programme qui permet de tracer des courbes à partir de commandes interprétées, et de fichiers de données.

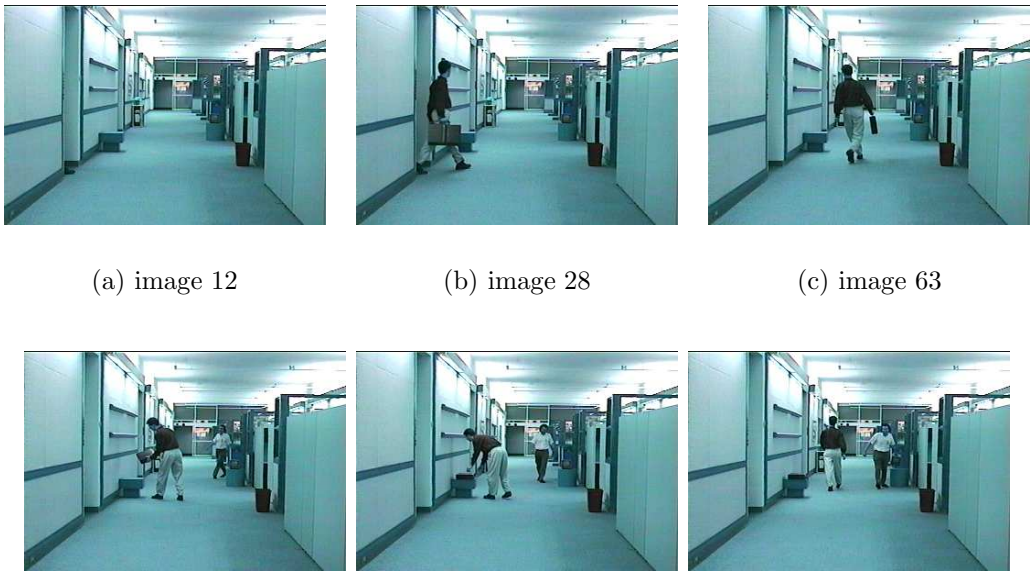
2.3 Les outils développés

Les expérimentations menées ont nécessité plusieurs étapes (détails au chapitre 3 page 23) dont celle de l'encodage qui était très longue (plusieurs heures), il a donc été nécessaire d'automatiser ces opérations. Cette automatisation a été possible par l'utilisation de scripts écrits en Bash et Gawk.

2.4 Les séquences étudiés

Une des séquences que nous avons le plus étudiée est la séquence *Hall* (échantillons sur la figure 2.1 page suivante). La prise de vue est fixe, et a lieu dans un couloir, où deux hommes apparaissent, déambulent dans le couloir, puis disparaissent. Cette séquence est intéressante car le début est sans mouvement, puis les individus apparaissent un par un

La séquence *News* visible sur la figure 1.7 page 17, a été construite afin d'obtenir des changements de plan. Cette séquence comporte 8 images dont un changment de plan à la troisième image, et un retour sur le premier plan à la cinquième.



(a) image 12 (b) image 28 (c) image 63

(d) image 96 (e) image 131 (f) image 178

FIG. 2.1 – La séquence *Hall*.

Chapitre 3

Expérimentations

Tout d'abord, nous avons choisi de nous limiter à l'utilisation des images de type I et P, et donc d'écarter les images de type B.

Pour effectuer nos expérimentations, nous souhaitions pouvoir utiliser plusieurs images de références, pour exploiter au mieux les connaissances a priori de la séquences étudiée. C'est pour cela que nous nous sommes orientés vers l'encodeur de référence H.26L (*H.26L Reference Software*).

3.1 Modification de l'encodeur de référence

La version que nous avons utilisée est la 6.1, elle est fournie avec les fichiers d'environnement destinés à être utilisés avec Visual C++, et un Makefile pour plateforme Unix.

L'encodeur fonctionne avec un fichier de configuration qui spécifie les options à utiliser, un exemple de ce fichier se trouve en annexe (voir page 33). En plus de ce fichier de configuration, des options peuvent être passées en ligne de commande à l'encodeur. Cependant, la vérification de la validité des options est vérifiée après la lecture du fichier de configuration, mais avant la lecture de celles fournies en ligne de commande. Cette validation des options comprend la verification de la possibilité d'accéder aux fichiers à utiliser, la cohérence entre paramètres, et le respect des bornes. Pour remédier à ce problème, il suffit de déplacer l'appel à la fonction `PatchImp()` réalisé à la fin de la fonction `ParseContent(char* , int)` dans le fichier `configfile.c` à la fin de la fonction `Configure(int , char*)` située dans le même fichier.

Afin de faire nos expériences, il fallait modifier le code source pour pouvoir forcer l'utilisation une image spécifique comme image de référence. J'ai donc commencé par analyser la structure du code, et à en dégager les principales fonctions et structures. Les résultats de cette analyse sont présentés dans la

section suivante.

3.1.1 Analyse du code source de l'encodeur de référence

La structure `ImageParameter` (déclaré dans le fichier `global.h`) permet de stocker l'image en cours de codage. Le champ `number` de la variable globale `img` (de type `ImageParameter`) renseigne sur l'indice de l'image en cours de codage. Ce champ ne sera utilisé que pour les images de type I ou P.

La fonction `encode_one_frame` est la fonction principale d'encodage qui est appelée pour chaque image de la séquence (quelque soit leur type). Par la suite plusieurs fonctions sont appelées de manière imbriquée : `frame_picture`, `code_a_picture`, `encode_one_slice` , `encode_one_macroblock` et `PartitionMotionSearch`. La dernière de celles-ci (`PartitionMotionSearch`) sert à chercher les vecteurs de mouvement en utilisant plusieurs images de références. Cette recherche est effectuée en utilisant toutes les images de références disponibles.

Les images de références sont indexées par la variables `ref` de type entière. Cette indicage est fait de manière relative par rapport à l'indice de l'image codée, selon la formule suivante :

$$\text{indice de l'image de référence} = \text{indice de l'image codée} - \texttt{ref}$$

3.1.2 Modification apportées

Après cette analyse qui fut assez longue en raison de l'opacité du code source, j'ai tenté de contraindre l'utilisation d'images de références pour le codage des images de type P. Malgré plusieurs tentatives, je ne suis pas parvenu à faire fonctionner l'encodeur correctement. Face à ces difficultés qui ont fortement ralentit le déroulement de mon stage, il fallut trouver une solution alternative afin de mener à bine nos expérimentations.

3.2 Reconstruction de séquences

Puisque la modification de l'encodeur n'était pas possible, nous avons décidé de modifier les séquences en entrée pour pouvoir choisir les images qui seraient utilisées comme images de références, ou comme images-clé des GOPs (images I). Nous avons donc découpé les séquences en GOPs à l'aide des deux outils `yuv2ppm` et `ppm2yuv` (voir 2.2 page 20). Lors de la reconstruction du

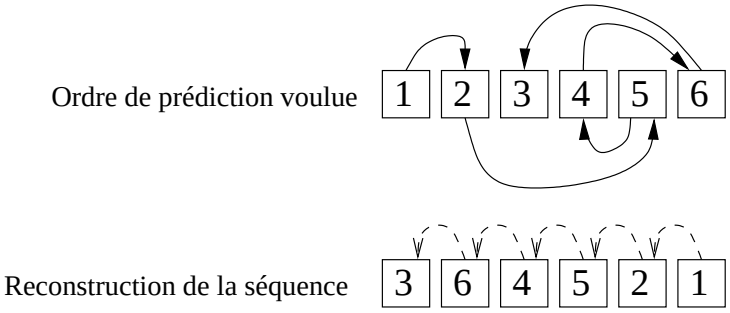


FIG. 3.1 – Reconstruction d'une séquence.

GOP, nous pouvions alors modifier l'ordre des images (figure 3.1) afin de voir l'influence des images de références choisies sur la qualité de la compression.

Le script `buildSequence.sh` fourni en annexe permet de reconstruire une séquence en spécifiant les indices des images désirées.

3.3 Recherche de la meilleure image it intra

Etant donné que la reconstruction des séquences permettait de choisir l'image it intra d'un GOP, nous avons voulu observer l'influence du choix de l'image *intra* sur la qualité de la compression (au sens débit/distorsion). Le script `bestIFrame.sh` fourni en annexe permet pour un GOP donné, de tester l'encodage avec chaque image du GOP comme image *intra*, et de fournir un graphique représentant les coûts de stockage des images P et les PSNR moyens des images décodées de chaque encodage.

On peut voir sur la figure 3.2 page suivante le résultat de l'exécution de ce script sur la séquence Hall pour le GOP comprenant les images 18 à 30. On remarque un gain au niveau du stockage pour l'encodage de la séquence en utilisant la deuxième image (l'image 19) comme image *intra*. En revanche les images suivantes n'obtiennent pas de bon résultats. L'utilisation de l'image 21 produit également un résultat équivalent (stockage un peu plus élevé, mais distorsion plus faible).

Pour obtenir de meilleurs résultats, nous allons alors réordonnancer complètement la séquence.

3.4 Ordonnancement des images

La démarche consiste à rassembler dans un premier temps les images similaires en se basant sur l'EQM. Une nouvelle séquence est donc recons-

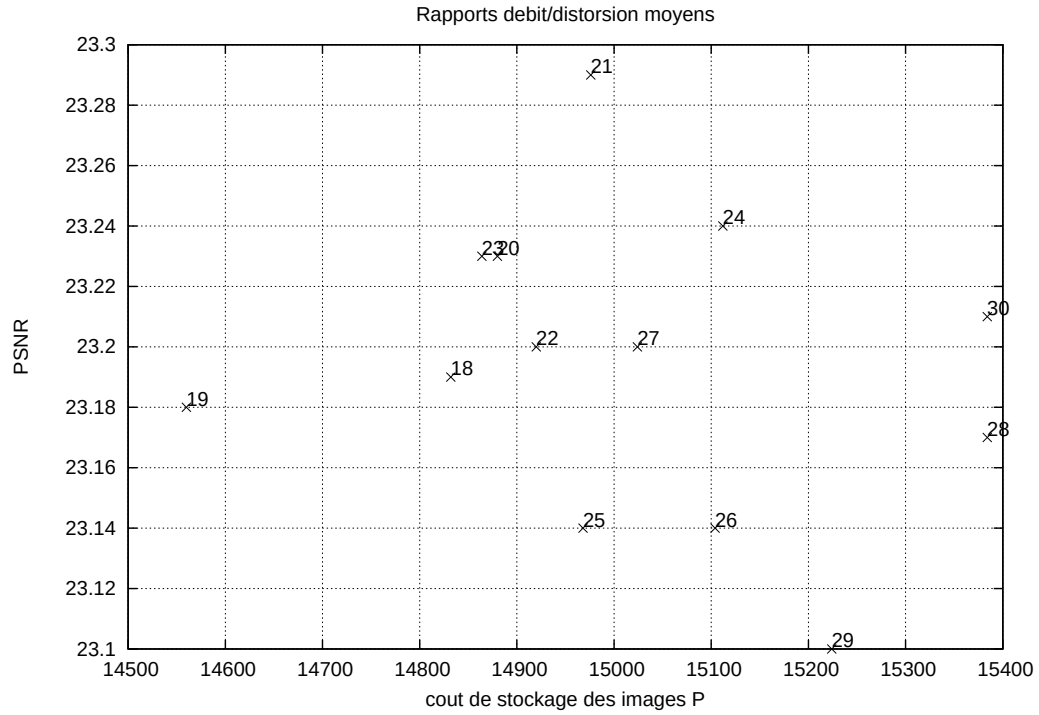


FIG. 3.2 – Rapports débit/distorsion pour les différentes images intra du GOP 18-30 de la séquence *Hall*.

truite image par image, en comparant chaque image aux suivantes, afin de selectionner la plus proche au sens de l'EQM. Cette EQM est calculée avec les images codées et décodées en mode *intra*, afin de prendre ne compte les pertes introduite dans la compression. Cette opération est réalisée par le script `ordonnancement.sh` (voir annexes). Puis, une recherche de la meilleure image *intra* est effectuée pour chaque GOP de la séquence.

Une fois toutes ces informations calculées, il ne reste plus qu'à construire la séquence à encoder à partir de ces informations. L'ensemble de ces opérations est effectuée par le script `encod_seq_reordonnancee.sh` (voir annexes).

Voici les résultats de cette méthode sur la séquence *News* :

```
encodage avec réordonnancement et choix des meilleures images intra :
   réordonnancement de la séquence : 0 1 6 7 5 4 3 2
   meilleure image intra :
     GOP 1 : image 1
     GOP 2 : image 1
   débit :      1045.32   kbit/s
   PSNR moyen : 38.051030 dB

encodage de la séquence originale :
   débit :      1367.73   kbit/s
   PSNR moyen : 38.036927 dB
```

L'ordonnancement de la séquence a été performant puisque les images ont été regroupées par plan (voir section 2.4 page 21). Le débit obtenu avec le réordonnancement de la séquence est bien meilleur que celui obtenu avec la séquence originale (-24%), pour la même qualité visuelle.

Conclusion

Ce stage m'a permis de découvrir le domaine du traitement du signal vidéo et les dernières techniques utilisées dans le processus de compression vidéo.

L'idée de ce stage était d'effectuer des tests sur la modification de la séquence d'encodage des images d'une séquence vidéo, dans le but d'apporter un gain sur le résultat obtenu. Pour cela, nous avons choisi de nous baser sur le logiciel de référence du standard H.26L.

L'analyse infructueuse du code source de l'encodeur de référence a fortement compromis l'avancement des expérimentations. Néanmoins, des scripts permettant des comparaisons entre un processus d'encodage classique et un encodage modifié ont été développés. En raison du temps nécessaire à l'encodage, les test de ces scripts furent assez longs.

Certains tests ont permis de mettre en évidence le gain possible pour certaines séquences par la modification des images de références utilisées. En revanche, aucune méthode performante n'a été mise en avant,en ce qui concerne l'extraction de connaissances sur la séquence, même si la comparaison par EQM reste une solution viable.

Bibliographie

[1] UIT-T. Recommandation h.263, version 2 : Codage vidéo pour communications à faible débit, 1998.

[2] Rob Koenen. Overview of the mpeg-4 standard iso/iec jtc1/sc29/wg 11 n4030. Mars 2001.

[3] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG. Joint Final Commitee Draft (JFCD) of Joint Video Specification (ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC).

[4] Claude Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27(3) :379–423, juillet 1948.

[5] Majid Rabbani and Paul W. Jones. *Digital Image Compression Techniques*, volume 7 of *Tutorial texts in optical engineering*. SPIE Optical Engineering Press, Bellingham, WA, USA, 1991.

[6] Berger T. *Rate Distorsion Theory*. NJ : Prentice Hall, Inc., 1971.

[7] Henri Nicolas. *Contribution à la création et à la manipulation des objets vidéo*. PhD thesis, Université de Rennes 1, Institut de Formation Supérieure en Informatique et Communication, 2001.

[8] M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu. Efficient representations of video sequences and their applications. *Signal Processing : Image Communication*, 8 :327–351, 1996.

[9] JVT Software Page, JM/TML Software Coordination. http ://bs.hhi.de/∼suehring/tml/.

[10] Guy Côté and Lowell Winger. Progrès récents dans le domaine de la compression vidéo. *IEEE Canadian Review*, 2002.

[11] UB Video Inc. Emerging H.26L Standard, Overview and TMS320C64x Digital Media Platform Implementation.

[12] Newsgroup gmane.comp.video.h264.devel.

[13] Hdot264 video codec. http ://sourceforge.net/projects/hdot264/.

Annexes

Dans ces annexes, sont regroupés différents scripts qui ont été développés pour réaliser les expérimentations, mais également le code source d'un outil développé pour calculer le PSNR entre deux séquences (`computePSNR.cpp`), et un exemple de fichier de configuration de l'encodeur de référence H.26L (`encoder.cfg`).