Compression vidéo fondée sur l'apparence

Karl Schwerdt

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Spécialité : « IMAGERIE, VISION ET ROBOTIQUE »

préparée au laboratoire GRAVIR – IMAG au sein du projet PRIMA
INRIA Rhône-Alpes, 655, ave. de l'Europe, 38330 Montbonnot Saint Martin

dans le cadre de l'école doctorale
« MATHÉMATIQUES, SCIENCES ET TECHNOLOGIE DE L'INFORMATION »

présentée et soutenue publiquement

par

Karl SCHWERDT

le 18 mai 2001

(Version préliminaire)

Titre :

COMPRESSION VIDÉO FONDÉE SUR L'APPARENCE

Directeur de thèse :
James L. CROWLEY

JURY

Président	M. Claude PUECH (UJF Grenoble)
Rapporteurs	M. Murat KUNT (EPF Lausanne)
	M. Claude LABIT (INRIA Rennes)
Examinateurs	M. James L. CROWLEY (INP Grenoble)
	M. Gerhard RIGOLL (Universität Duisburg)

Table des matières

Liste des tableaux

Table des figures

Première partie

COMPRESSION VIDÉO FONDÉE SUR L'APPARENCE (extraits)

Prologue

apparence n. f. Ce qui se présente immédiatement à la vue ou à l'esprit : *Une maison de belle apparence. Toutes les apparences sont contre l'accusé.* || Aspect extérieur, qui ne répond pas à la réalité ; semblant : *Cacher sous une apparence bonasse une dureté intraitable.* || — SYN. : *air, dehors, extérieur, façade, phénomene, probabilité, semblant, vraisemblance.* • *Sauver les apparences,* ne rien laisser paraître qui puisse nuire à la réputation ou blesser les bienséances. • LOC. ADV. *En apparence,* extérieurement, à en juger d'après ce que l'on voit.

Larousse [Lar92].

Chapitre 1

Introduction

Depuis sa création en 1990 avec la définition de *HyperText Transport Protocol (HTTP)* [HTT00], le World Wide Web a révolutionné presque chaque aspect de la communication et du calcul personnel et professionnel. En même temps, le *Global System for Mobile Communications (GSM)* [GSM01] est devenu le protocole le plus important pour le téléphone mobile. L'*Universal Mobile Telecommunications System (UMTS)* fait partie des systèmes de communication mobile de troisième génération (3G) de l'International Telecommunications Union (ITU) [UMT01]. En 2000, les compagnies de télécommunication ont dépensé le montant sans précédent de 305 milliards € pour obtenir des licences d'UMTS pour les services mobiles de communication 3G en Europe [BBF+01]. UMTS fournira de l'information et des services de commerce et de divertissement aux utilisateurs mobiles par l'intermédiaire de réseaux fixes, radio, et satellites, soutenant de cette façon l'intégration des télécommunications, des technologies d'information, des médias et des services de contenu. Ceci correspond à une connexion Internet mobile permanente avec des services de voix, de vidéo, de communication de données, et d'autres services d'information intégrés. La figure 1.2 illustre les changements des habitudes de communication (mobile) pour un tel genre de systèmes.

FIG. *1.1: La mine d'or de communications globales : Terminal UMTS (source [BBF+01])*

La largeur de bande visée de transmission pour UMTS est 2 mégabits/seconde, mais pour son début en 2002, cette cadence a été réduite à 384 kilobits/seconde en raison de problèmes techniques. L'obstacle technique principal est l'alimentation d'énergie pour les appareils d'utilisateurs. Bien qu'il n'ait pas répondu à toutes les espérances, le *Wireless Application Protocol (WAP)* [WAP00] pour les téléphones mobiles était une première étape pour rendre invisible l'emplacement physique d'un terminal d'utilisateur. Dès lors qu'il fonctionnera comme annoncé, UMTS, en effet, rendra sans importance l'emplacement physique réel d'un utilisateur et des informations qui lui sont fournies. Un terminal UMTS avec une connexion à l'Internet est tout ce qui est nécessaire pour rendre n'importe quelle information disponible partout sur cette planète. La figure 1.3 démontre quelques exemples d'études de projets des terminaux UMTS.

Cet environnement technologique crée une situation dans laquelle il est nécessaire de transmettre une quantité de données par les lignes de communication avec une largeur de bande très limitée, qu'elles soient par radio ou par fil. Ceci mène aux demandes techniques suivantes :

⇒ Représentation de données efficace, élimination d'information redondante

⇒ Protection de la sécurité et de l'espace privé pendant la communication et la transmission ou l'accès aux données

⇒ Accès omniprésent et instantané/permanent à l'information

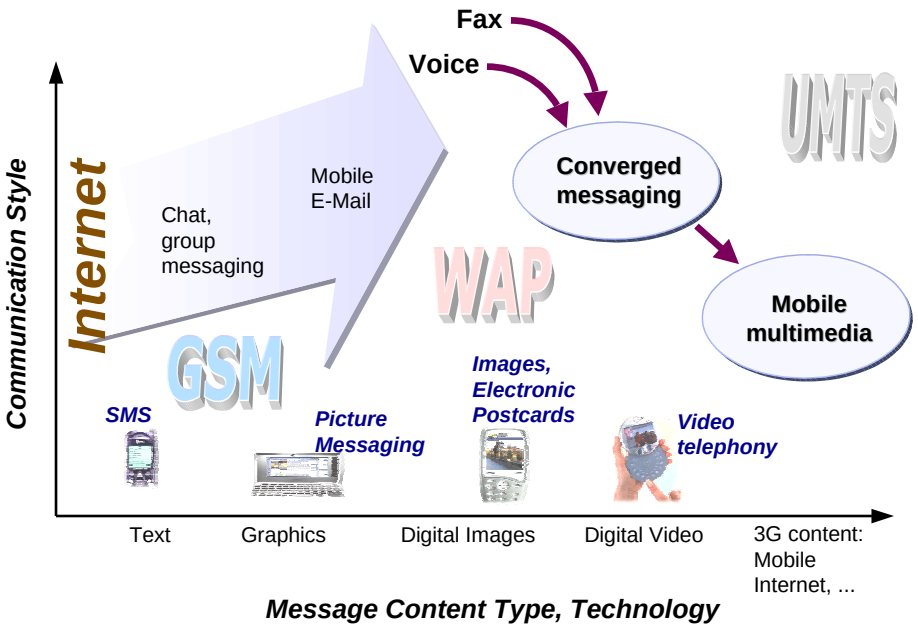


FIG. *1.2: Évolution des styles de communication et des contenus de messages. Nous notons de gauche à droite le développement du texte, d'après graphiques, images numériques, et vidéo numérique à l'intégration de contenu mobile multimédia. Verticalement nous voyons l'intégration de services fax, voix, et Internet* (WWW, E-Mail, Chat) *séparés dans un seul service.*

Du côté du hardware, la puissance de calcul exponentiellement croissante et les nouveaux périphériques et techniques telles que les caméras pilotables avec suivi intégré, les écrans à contact, les capteur-gants, et les logiciels de reconnaissance de la parole ouvrent de nouvelles voies d'interaction homme-machine. La conséquence est le développement de demande complexes de multimédia, de système de vidéoconférences, de la vidéotéléphonie, d'échange de données vidéo, et d'autres de communication vidéo. Par contre, les capacités améliorées des environnements de calcul actuels (et à venir) vont au-delà de la création de nouvelles applications. Les publications récentes comme [CCB00, Pen00] démontrent que les systèmes soutenus par ordinateur deviennent maintenant capables de *percevoir* leur environnement. Une fois réussi, ceci changera la façon

dont on interagit avec les ordinateurs. Alors que ces nouvelles applications sont développées, la puissance de calcul insuffisante reste un problème critique à cause des largeurs de bande de transmission et de l'espace mémoire limités.

Ericsson Motorola Siemens SX45 Nokia C3 Nokia F3

FIG. *1.3: Études de conception de terminaux UMTS [GSM01]*

Une largeur de bande et une puissance de calcul croissante feront de la vidéo une propriété de base des appareils et des services de communication. Cependant, même les canaux de communication à grande vitesse les plus modernes exigent la compression vidéo. Les formats d'images de séquences vidéo les plus communs sont ceux utilisés pour l'émission de télévision, c'est–à–dire, NTSC[1], PAL[2], SECAM[3], et CIF[4]. Le tableau 1.1 contient les tailles d'images correspondantes [ITU96, ITU93, ITU94].

TAB. *1.1: Formats d'images standards importants*

Standard	pixels / ligne	lignes / image	images / seconde	codage de pixels
NTSC	858 (720 digital active)	525	30	YC_RC_B (ITU-R BT.601-4)
PAL	864 (720 digital active)	625	25	YC_RC_B (ITU-R BT.601-4)
CIF	352	288	29.97	YC_RC_B (ITU-R BT.601-4)

Afin de démontrer la quantité de données que nécessite l'information vidéo, considérons deux exemples.

[1]NTSC = *National Television Standards Committee*, approuvé par la Commission fédérale de transmissions (FCC) des Etats-Unis ; standard visuel analogique officiel aux Etats-Unis, Canada, Mexique, Japon, Taïwan, Corée, et quelques régions d'Amérique centrale et sud. Une autre définition est « Never Twice the Same Color ».

[2]PAL = *Phase shift on Alternate Lines* ; standard vidéo analogique le plus répandu, utilisée au Royaume-Uni, en Allemagne, en Espagne, au Portugal, en Italie, en Chine, en Inde, dans la majeure partie de l'Afrique et du Moyen-Orient

[3]SECAM = *Système sÉquentiel Couleur À Mémoire* ; utilisé en France, en Russie, en Europe de l'Est, et quelques parties du Moyen-Orient. Une autre définition est « Surtout Éviter la Compatibilité Avec le Monde »

[4]CIF = *Common Intermediate Format* ; conçu par l'ITU-T comme format intermédiaire entre PAL et NTSC, en particulier pour la compression vidéo

Exemple 1 *Une seconde d'une séquence vidéo* non-comprimée *correspond à*

$$864\,\frac{pixels}{ligne}\cdot 625\,\frac{lignes}{image}\cdot 25\,\frac{images}{sec}\cdot 16\,\frac{bits}{pixel}\cdot 1\,sec\;=\;216\,mégabits\;=\;27\,mégabytes,$$

admettant un codage de 8 bits pour la luminance et la chrominance et un sous-échantillonnage de 1 : 4 des deux valeurs de chrominance. C'est–à–dire, une séquence vidéo de 90 minutes représente

$$27\,\frac{mégabytes}{sec}\cdot 60\,\frac{sec}{min}\cdot 90\,min\;=\;145.8\,gigabytes.$$

Ceci correspond à 225 (!) CD-ROM d'une capacité de 650 mégabytes.

Un réseau ethernet de 100 mégabits/sec ne serait pas suffisant pour traiter cette quantité de données dans une application temps-réel. Ce n'est donc pas concevable de transmettre des données non-comprimées par l'Internet. Même si on réduit fortement la taille des images, par exemple au format QCIF, cela crée rapidement une quantité de données considérable.

Exemple 2 *Une seconde d'une séquence vidéo non-comprimée avec des images en format QCIF correspond à*

$$176\,\frac{pixels}{line}\cdot 144\,\frac{lines}{img.}\cdot 29.97\,\frac{img.}{sec}\cdot 16\,\frac{bits}{pixel}\cdot 1\,sec\;=\;12.2\,mégabits\;=\;1.5\,mégabytes.$$

Ceci admet un codage 8 bits pour les valeurs de luminance et de chrominance et un sous-échantillonnage de 1 : 4 des deux valeurs de chrominance. Un E-Mail vidéo de 5 minutes en format QCIF et non-comprimé aurait

$$1.52\,\frac{mégabytes}{sec}\cdot 60\,\frac{sec}{min}\cdot 5\,min\;=\;455.7\,mégabytes,$$

ce qui est beaucoup pour une boite aux lettres électroniques et cela boucherait rapidement toute connexion internet, même à large bande.

Nous verrons dans le chapitre 4, qu'il n'y a aucun algorithme universel de compression qui comprime efficacement toutes les sortes de données. L'exécution des algorithmes de compression dépend toujours de l'application et du modèle de données utilisés. Nous choisissons donc d'abord notre domaine d'application et un modèle approprié pour nos données, faisons nos mesures et déterminons enfin à quel point notre algorithme fonctionne. Comme indiqué au début de ce chapitre, le domaine d'application choisi pour ce travail est la communication vidéo « point-to-point ».

Pour le domaine d'application de la compression vidéo, il y a déjà une variété de techniques. Cependant nous justifierons dans la prochaine section notre propre approche, qui est fondamentalement différente des approches existantes du problème. En dehors des algorithmes propriétaires et commerciaux de compression vidéo tels que Cinepak [CTI01], RealVideo [Rea01] streaming, et DVD [DVD01], un certain nombre de standards techniques internationaux existent tels que H.261 [ITU93], et H.263 [ITU96], édités par le *secteur de télécommunication de l'ITU (ITU-T),*

et MPEG-1 [ISO93] et MPEG-2 [ISO96a] de l'*International Organization for Standardization (ISO)*. Ces standards sont employés dans la plupart des produits commerciaux en utilisant la technologie de compression vidéo telle que la vidéotéléphonie, la communication vidéo, l'archivage de films et la télévision numérique. Tout en étant très efficaces, ces standards ont leurs limites de rentabilité imposant des compensations entre la qualité de reconstruction et le taux de compression. Ceci a naturellement mené, grâce à des efforts de recherche, à repousser ces limites plus loin.

D'ailleurs, les appareils modernes comme pour UMTS exigent une intégration de médias différents et, finalement, des possibilités de l'interaction avec leur contenus. Les développements actuels tels que MPEG-4 et MPEG-7 d'ISO répondent à ces demandes. En utilisant la même méthode en trois étapes pour la compression vidéo que MPEG-1 et MPEG-2, ces nouveaux standards fournissent un cadre pour la fusion des contenus vidéo, audio, multimédia, et pour l'interaction avec ces contenus. Les outils d'accès au contenu d'images viennent de la vision par ordinateur. On pourrait dire que la vision par ordinateur donne aux ordinateurs la capacité de voir par l'extraction d'information d'images. L'interaction de la vision par ordinateur et de la compression vidéo est donc une conséquence normale de la nécessité de satisfaire de nouvelles demandes technologiques. C'est le postulat de cette thèse.

1.1 Sujet de recherche et approche

Les standards actuels d'ITU-T et d'ISO pour le codage vidéo se fondent principalement sur la relation statistique entre des images et des pixels. Intuitivement, l'utilisation d'information supplémentaire accessible *a priori*, sagement appliqué, devrait apporter une augmentation du taux de compression. Pour le cas de la communication « *point-to-point* » entre deux personnes, ceci correspond en principe à un scénario de visage et d'épaules (*head-and-shoulders* ci-dessous), qui est également utilisé pour des approches de compression vidéo basées sur des modèles [HL96]. Nous acceptons ce scénario et le paradigme d'extraire de l'information de l'image afin d'améliorer la compression vidéo au delà de la performance des standards actuels. Cependant, nous croyons que les méthodes pour l'extraction de contenu d'image basées sur l'apparence sont plus flexibles face à un contenu d'image changeant et plus faciles à utiliser que des modèles géométriques 3D.

Nous affirmons qu'on n'a pas besoin de toutes les informations *possibles* d'un contenu d'image. On devrait plutôt se concentrer sur ce qu'on *voit* dans l'image. Cette distinction est importante, parce que la façon dont on regarde les données détermine l'approche pour leur traitement. Puisque l'étape de codage décisive de notre méthode de compression est le calcul d'un espace de base orthonormal à partir d'un ensemble d'images sélectionnées, nous appelons notre technique *Codage de Bases Orthonormales (CBO)*, en anglais *Orthonormal Basis Coding (OBC)*. CBO crée une représentation optimale des données d'entrée en ce qui concerne le compactage d'énergie, et fournit une représentation permettant la manipulation du contenu avec d'autres méthodes basées sur l'apparence de la vision par ordinateur. La figure 1.4 localise CBO à l'intersection de la compression vidéo, de la vision par ordinateur, et de l'identification de configuration.

Les exemples d'application pour CBO sont, par exemple, le courrier électronique vidéo (*Video E-Mail*), où une séquence est enregistrée pour être transmise comme un E-Mail ; la compression vidéo ou de *films*, bien qu'un algorithme conçu pour un scénario *head-and-shoulders* n'est pas forcement approprié pour le codage vidéo de plein mouvement ; un visage parlant sur des pages web (*talking head on web pages*) pour guider l'utilisateur par la sélection de produits ; cela est juste *un* exemple pour la vidéo interactive, un outil pour la convenance d'utilisateurs dont les entreprises qui vendent par le web ou par la télé pourraient profiter ; et finalement le *vidéotéléphone* et la *vidéoconférence* pour une version en ligne de CBO.

Les méthodes basées sur l'apparence exigent que l'entrée de données vidéo soit normalisée à un objet principal. Nous présentons dans cette dissertation un nouveau suivi de visage de basse complexité basé sur la détection et le suivi de couleur. Ce suivi rapporte une stabilisation de données d'entrée suffisamment précises pour permettre l'utilisation de CBO. Cependant, comme les résultat le suggèrent, une amélioration du suivi, probablement par d'autres modules que la détection de couleur, pourrait encore contribuer à l'efficacité de CBO.

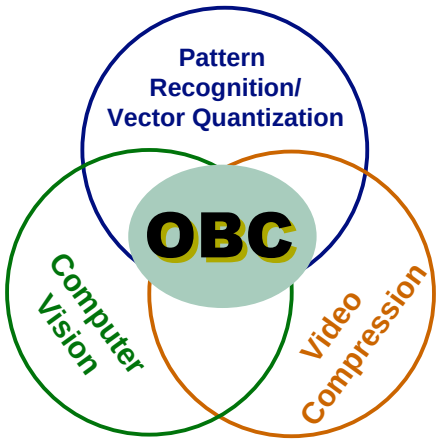


FIG. *1.4: Localisation de CBO à l'intersection de la compression vidéo, de la vision par ordinateur, et de l'identification de configuration.*

Un problème particulier rencontré lors de l'évaluation de CBO est que la dégradation d'image liée à la compression est essentiellement différente de celles imposée par d'autres méthodes de compression utilisant des bases fixes telles que MPEG. Les mesures de qualité de reconstruction utilisées actuellement ne considèrent pas la perception humaine visuelle. Ce travail souligne donc la nécessité du développement des mesures de qualité de reconstruction basées sur le système visuel humain. Ceci exige une collaboration avec des psychologues, dont la contribution est de valeur inestimable pour la conception d'une série d'essais pour mesurer la perception visuelle humaine. Ceci pourrait par la suite aider à la définition d'une description mathématique de l'estimation humaine de la qualité d'image.

Heureusement, même avec les inconvénients de l'utilisation des mesures de qualité d'images conventionnelles comme le *Peak Signal-to-Noise Ratio (PSNR)*, CBO a de bonnes performances comparé à MPEG en termes de qualité de reconstruction et de taux de compression. Si les préalables de CBO tels qu'un suivi précis sont réalisés, CBO surpasse facilement la compression MPEG[5] en termes de qualité et de compression. Malheureusement, CBO surpasse aussi MPEG en terme de lourdeur de calcul. C'est dû la plupart du temps aux algorithmes complexes et surtout à des codes non-optimisé. CBO, comme il est présenté dans cette dissertation, est en principe un

[5]Codec de référence est le logiciel `mpeg_encode`de Université de Californie à Berkeley dans diverses configurations [Ber97]

processus hors ligne, quoique les considérations pour une version incrémentale sont documentées et les expérimentations – non incluses – montrent des résultats prometteurs.

1.2 Organisation du manuscrit

Le **chapitre 3** trace les grandes lignes du contexte technique et scientifique de ce travail, analysant la contribution des méthodes de la vision par ordinateur basées sur l'apparence à la compression des signaux vidéo. Les méthodes basées sur l'apparence demandent en général un suivi d'objet, et c'est pour cela que nous définissons trois catégories pour trier le grand nombre d'algorithmes de suivi de la vision par ordinateur. Une sélection des plus importants est utilisée pour illustrer l'utilité d'une telle catégorisation. Après avoir prouvé qu'un suivi de couleur, basé sur des propriétés *(feature-based)*, est suffisant pour l'application destinée, nous tournons notre attention vers la compression vidéo. Un rapide historique des algorithmes de compression vidéo les plus importants d'aujourd'hui est donnée suivie des perspectives sur les développements actuels et à venir.

La base théorique de ce travail est passé en revue dans le **chapitre 4**. Point par point des des critères de classification pour les ingrédients d'un codec[6] vidéo sont dérivés de la théorie d'information. Ces critères de classification permettent une analyse des codecs vidéo actuellement existants et soutiennent la synthèse de nouvelles techniques pour la compression vidéo. Les algorithmes de compression relevant pour ce travail sont développés systématiquement à partir de ces critères de classification, illustrant les relation entre eux ainsi que leurs points forts et leurs faiblesses.

En utilisant les résultats de chapitre 4, nous discutons en bref les techniques de compression les plus importantes telles que MPEG, H.263, JPEG et ZIP dans le **chapitre 5**. Ces techniques fournissent une référence pour, ou sont mis en application dans CBO. On voit comment les techniques des différents critères de classification sont combinées créant des codecs efficaces pour des applications particulières. Des perspectives données sur les développements actuels illustrent les limites des codecs actuels et motivent à nouveau ce travail.

Nous élaborons les détails des techniques développées pour ce travail dans le **chapitre 6**. Un feature-based suivi de couleur est motivé et conçu. Nous démontrons comment des histogrammes de couleur peuvent être employés pour localiser et suivre des objets. Dans une première étape, les histogrammes sont employés pour créer un plan de probabilité pour les pixel de la couleur de peau utilisant la règle de Bayes. Un algorithme efficace de suivi basé sur les premiers et deuxièmes moments de ce plan de probabilité est proposé. Cet algorithme, appelé l'algorithme CENTEROFGRAVITY, est inspiré par la statistique robuste. Il élimine des points éloignés en pesant de nouvelles données d'entrée. De plus l'algorithme CENTEROFGRAVITY utilise la compensation de peser et de mouvement inspirée par les concepts du filtrage de Kalman. Le résultat est un suivi de couleur robuste qui surpasse des algorithmes de suivi conventionnel utilisant des seuils, des composants

[6] codec = *coder - décoder*

connectés et des filtres de Kalman en termes de précision tout en ayant une performance de calculs comparable.

Le **chapitre 7** développe l'algorithme du *Codage de Bases Orthonormales (CBO)*. Le concept de CBO, qui associe l'algorithme à ses applications destinées, est décrit. Puisque la façon dont les données sont traitées par un algorithme est étroitement liée à la façon dont les données sont perçues, nous comparons CBO et compression vidéo conventionnelle par le modèle de données utilisé. Les implications de la représentation des images comme vecteurs au lieu d'une décomposition en blocs sont discutées. Une conséquence de cela est que la quantification de vecteur *(vector quantization)* ou les algorithmes groupants *(clustering)* deviennent applicables. Nous discutons quelques algorithmes groupants importants, et leur impact la qualité de reconstruction et la durée de calcul sont illustrés. En conclusion, en raison de leur importance et de leur fréquente utilisation, les mesures de qualité de reconstruction les plus communes telles que MSE, SNR, et PSNR sont discutées en détail. Elles sont interprétées dans le contexte du traitement d'image et de la vidéo et leurs limitations sont montrées.

Le **chapitre 8** présente les résultats qui démontrent que CBO est une alternative valide pour la standard et d'autres algorithmes visuels de compression. Il est particulièrement efficace si les données d'entrée sont bien normalisées à un objet, dans ce cas le visage d'un locuteur dans un scénario *head-and-shoulders*. Des résultats de codage sont évalués pour sept séquences, dont quatre sont des séquences standard employés par la communauté de chercheurs en compression vidéo, les trois autres séquences ont été créées dans notre laboratoire. On montre que dans tous les cas CBO produit des résultats comparables à un encodeur de MPEG représentant les arrangements largement répandus de compression de DCT/DPCM. Dans certains cas, caractérisée par un ordre visuel bien normalisé, CBO surpasse de manière significative le codec de MPEG de référence en termes de qualité et/ou compression. Cependant, CBO est moins performant en termes de puissance de calcul. En effet, le logiciel encodant de CBO n'a pas été optimisé en ce qui concerne la vitesse de calcul. Des résultats comparatifs sont donnés en utilisant la durée de calcul comme mesure.

Le codec CBO présenté dans cette dissertation peut être vu comme première étude de faisabilité d'un codec vidéo basé sur l'apparence des objets dans une séquence vidéo en remplacement de la corrélation statistique entre les valeurs de pixel. Il y a toujours matière à des améliorations conceptuelles et par rapport à l'implémentation ; Le **chapitre 9** discute donc de quelques perspectives.

Deuxième partie

APPEARANCE-BASED VIDEO COMPRESSION (version anglaise complète)

Prolog

Two definitions of *appearance*:

> Main Entry: **ap·pear·ance**
> Pronunciation: `&-'pir-&n(t)s`
> Function: *noun*
> Date: 14th century
> **1 a :** external show : **SEMBLANCE** <although hostile, he preserved an *appearance* of neutrality> **b :** outward aspect : **LOOK** <had a fierce *appearance*> **c** plural : outward indication <trying to keep up *appearances*>
> **2 a** : a sense impression or aspect of a thing <the blue of distant hills is only an *appearance*> **b :** the world of sensible phenomena
> **3 a :** the act, action, or process of appearing **b :** the presentation of oneself in court as a party to an action often through the representation of an attorney
> **4 a :** something that appears : **PHENOMENON** **b** : an instance of appearing : **OCCURRENCE**

from *Merriam-Webster's Collegiate Dictionary* [MW00]

appearance

SYLLABICATION: ap·pear·ance
PRONUNCIATION: `&-pîr'&ns`
NOUN : **1.** The act or an instance of coming into sight. **2.** The act or an instance of coming into public view: "The author made a rare personal appearance." **3.** Outward aspect: "an untidy appearance." **4.** Something that appears; a phenomenon. **5.** A superficial aspect; a semblance: "keeping up an appearance of wealth." **6. appearances** Outward indications; circumstances: "a cheerful person, to all appearances."

from *The American Heritage Dictionary of the English Language: 4th Edition. 2000.* [AH00]

We will use *appearance* in the sense of 2a and b of [MW00] and 4 of [AH00] hereafter, which we consider equivalent.

Chapter 2

Introduction

Since its creation in 1990 with the definition of the *HyperText Transport Protocol (HTTP)* [HTT00], the World Wide Web has revolutionized almost every aspect of personal and professional communication and computing. At the same time, the *Global System for Mobile Communications (GSM)* [GSM01] has become the most important protocol for mobile telephone. The *Universal Mobile Telecommunications System (UMTS)* is a part of the International Telecommunications Union's third generation (3G) mobile communications systems [UMT01]. The year 2000 saw telecommunication companies spend the unprecedented amount of 305 billion € just for UMTS licenses for third generation mobile communication services in Europe [BBF+01]. UMTS will deliver broadband information, commerce and entertainment services to mobile users via fixed, wireless, and satellite networks, this way supporting the integration of telecommunications, IT, media and content services. This corresponds to a permanent mobile internet connection with integrated voice, video, data communication, and other information services. Figure 2.2 illustrates the anticipated change in (mobile) communication habits for such kind of systems.

Figure 2.1: UMTS terminal as the gold mine of global communications (source [BBF+01])

The targeted transmission bandwidth for UMTS is 2 Mbits/sec, but for its start in 2002, this rate has been reduced to 384 kbits/sec due to technical problems. Main technical obstacle is the power supply for end-user devices. Although it fell short of its expectations, the *Wireless Application Protocol (WAP)* [WAP00] for mobile phones was a first step towards making the physical location of a user terminal invisible. Once it works as announced, UMTS will indeed make the actual physical location of a user and the information provided to him unimportant. A UMTS terminal with a connection to the internet is all that is needed to make any information available everywhere on this planet. Figure 2.3 shows some examples from design studies of UMTS terminals.

This technological environment creates a situation where it is necessary to transmit an enormous amount of data over communication lines with very limited bandwidth, be they radio or wires. This leads to the following technical demands:

⇒ Efficient data representation, elimination of redundant information, data compression

⇒ Protection of security and privacy during communication and data access/transmission

⇒ Ubiquitous and instant/permanent access to information

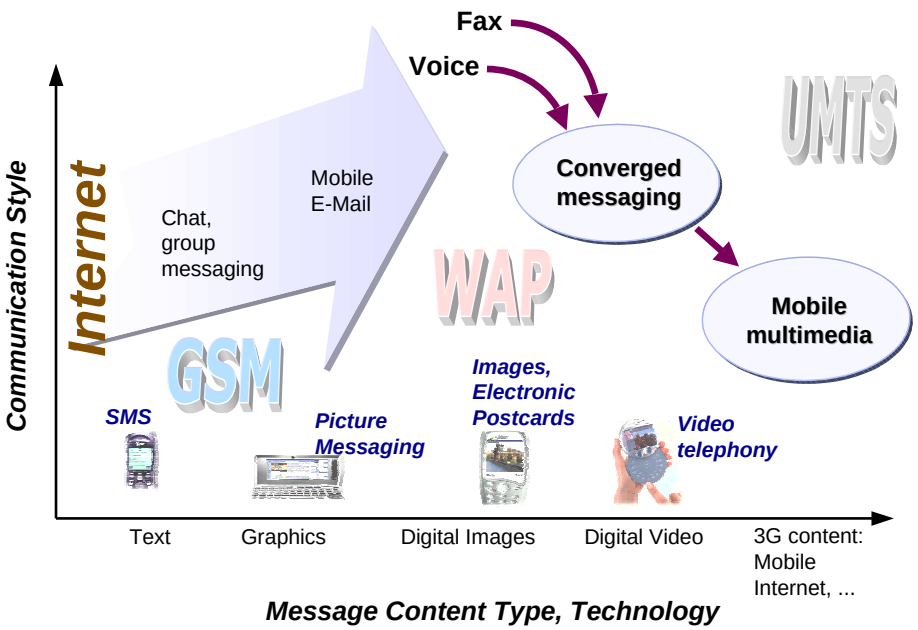


Figure 2.2: Evolution of communication styles and message contents. We note from left to right the development from text-based over graphics, digital images, and digital video to integrated mobile multimedia contents. Vertically we see an integration of separate Fax, Voice, and Internet (WWW, E-Mail, Chat) services into one service.

On the hardware side, exponentially increasing computing power and new peripheral devices and techniques such as steerable cameras with built-in tracker, touch-screens, sensor-gloves, and speech-recognition software enable new ways of human-computer interaction. The consequence is the development of complex multimedia applications for teleconferencing, video telephony, visual data exchange, and other video communications. But the improved capabilities of current (and future) computing environments go beyond the creation of new applications. Recent publications [CCB00, Pen00] show that computer supported systems are now becoming able to *perceive* their environment. Once succeeded, this will change the way people interact with computers. As these new applications are developed, the bottleneck of limited transmission bandwidths and storage space rather than insufficient computing power remains a critical problem.

Ericsson Motorola Siemens SX45 Nokia C3 Nokia F3

Figure 2.3: Design studies of UMTS terminals [GSM01]

Increasing bandwidth and computing power will make video a common feature of communication devices and services. However, even the most modern high-speed communication channels and hardware require video compression. The most common video image sequence formats are those used for television broadcast, i.e., NTSC[1], PAL[2], SECAM[3], and CIF[4]. Table 2.1 contains the corresponding video image sizes [ITU96, ITU93, ITU94].

TAB. *2.1: Important standard image formats*

Standard	pixels / line	lines / image	images / second	pixel encoding
NTSC	858 (720 digital active)	525	30	YC_RC_B (ITU-R BT.601-4)
PAL	864 (720 digital active)	625	25	YC_RC_B (ITU-R BT.601-4)
CIF	352	288	29.97	YC_RC_B (ITU-R BT.601-4)

In order to demonstrate how much data video information can be, consider two examples.

Example 1 *One second of an* uncompressed *PAL video stream corresponds to*

$$864\,\frac{\text{pixels}}{\text{line}} \cdot 625\,\frac{\text{lines}}{\text{image}} \cdot 25\,\frac{\text{images}}{\text{sec}} \cdot 16\,\frac{\text{bits}}{\text{pixel}} \cdot 1\,\text{sec} \;=\; 216\,\text{Mbits} \;=\; 27\,\text{MBytes},$$

[1]NTSC = *National Television Standards Committee*, approved by the Federal Communications Commission (FCC) of the U.S.A.; official analog video standard in the U.S.A., Canada, Mexico, Japan, Taiwan, Korea, some parts of central and south America. Another definition of NTSC is "Never Twice the Same Color".

[2]PAL = *Phase shift on Alternate Lines*; most widespread analog video standard, used in the U.K., Germany, Spain, Portugal, Italy, China, India, most of Africa and the Middle East.

[3]SECAM = *(in Engl.) Sequential Color system with Memory*; used in France, Russia, Eastern Europe, and some parts of the Middle East. It has also been dubbed "SomEthing Contrary to American Methods".

[4]CIF = *Common Intermediate Format*; designed by the ITU-T as an intermediate format between PAL and NTSC especially for video compression.

assuming 8-bit encoding for luminance and chrominance and a 1 :4 sub-sampling of both chrominance values. That is, a 90 minute PAL movie has a raw size of

$$27\,\frac{\text{MBytes}}{\text{sec}}\cdot 60\,\frac{\text{sec}}{\text{min}}\cdot 90\,\text{min} \;=\; 145.8\,\text{GBytes},$$

This corresponds to 225 (!) CDROMs with a capacity of 650 MBytes.

Even a 100 Mbits/sec Ethernet would be largely insufficient to handle this amount of data in a real-time application. It is thus unconceivable to transmit the raw data over the Internet. But even a heavily reduced image size, such as the QCIF format, does quickly produce a considerable amount of data.

Example 2 *One second of an* uncompressed *video stream with images in QCIF format corresponds to*

$$176\,\frac{\text{pixels}}{\text{line}}\cdot 144\,\frac{\text{lines}}{\text{img.}}\cdot 29.97\,\frac{\text{img.}}{\text{sec}}\cdot 16\,\frac{\text{bits}}{\text{pixel}}\cdot 1\,\text{sec} \;=\; 12.2\,\text{Mbits} \;=\; 1.5\,\text{MBytes},$$

assuming 8-bit encoding for luminance and chrominance and a 1 :4 sub-sampling of both chrominance values. A 5 minute video (e-)mail in QCIF format and uncompressed would have

$$1.52\,\frac{\text{MBytes}}{\text{sec}}\cdot 60\,\frac{\text{sec}}{\text{min}}\cdot 5\,\text{min} \;=\; 455.7\,\text{MBytes},$$

This is quite a lot of data for any (video-)mailbox, and would quickly congest even broadband internet connections.

As we will see in chapter 4, there is no universal compression algorithm which efficiently compresses all kinds of data. The performance of compression algorithms always depends on the application and on the data model used. We therefore choose our application area first, then select an appropriate model for our data, and finally make our measurements and decide how well our algorithm performs. As indicated by the beginning of this chapter, the application area chosen for this work is personal point-to-point video communication.

For the application area of video compression, there already is a variety of techniques. We will however justify in the next section our own approach, which is fundamentally different from existing approaches to the problem. Other than proprietary, commercial video compression algorithms such as Cinepak [CTI01], RealVideo [Rea01] streaming, and DVD [DVD01], there are a number of international technical standards such as H.261 [ITU93], and H.263 [ITU96], published by the *Telecommunication Standardization Sector* of the *International Telecommunications Union (ITU-T)*, and MPEG-1 [ISO93] and MPEG-2 [ISO96a] from the *International Organization for Standardization (ISO)*. These standards are employed in most commercial products using video compression technology such as video telephony, video conferencing, movie storage and digital television. While being very efficient, those standards have their usability limits imposing trade-offs between reconstruction quality and compression ratio. This has naturally led to research efforts pushing those limits further.

Moreover, modern application devices such as for UMTS demand an integration of different media and, eventually, possibilities for interaction with their content. Current developments such as ISO standards MPEG-4 and MPEG-7 address these demands. While they use the same three-step encoding scheme as MPEG-1 and MPEG-2 in their video compression layers, these new standards provide a framework to merge video, audio, and other multimedia content, and to interact with that content. The access tools to image or video content come from the area of computer vision. In a sense, computer vision makes computers see by extracting information from an image or video image content. So the interaction of computer vision and video compression is a natural consequence out of the need to satisfy new technological demands. This is the starting point for this thesis.

2.1 Approach

Actual video coding standards from ITU-T and ISO rely mainly on the statistical relation between images and pixels. Intuitively, the use of additional *a priori* information, if wisely applied, should further increase compression ratios. For the case of point-to-point communication of two persons, this basically corresponds to a head and shoulders scenario, which is also used for model based approaches to video compression [HL96]. We agree with this scenario and the paradigm of extracting image information in order to improve video compression over current standards. However, we think that appearance based methods for image content extraction are more flexible to changing image content and easier to use than creating 3D models.

We claim that we do not need all *possible* information about an image content. We rather should focus in what we *see* in the image. This distinction is important, because the way we look at our data determines how we approach them. Since the decisive encoding step in our compression scheme is the computation of an orthonormal basis space our of a set of selected images, we call our technique *Orthonormal Basis Coding*, or short *OBC*. OBC creates an optimal representation of the input data with respect to energy compaction, and provides a representation enabling subsequence content manipulation with other appearance based methods from computer vision. Figure 2.4 localizes OBC at the intersection of video compression, computer vision, and pattern recognition.

Application examples for OBC are, e.g., *Video electronic* mail, where a sequence is recorded to be transmitted like an email ; *Video/Movie compression*, where it has to be considered that an algorithm designed for a head-and-shoulders scenario is not necessarily appropriate for full motion video encoding ; a *talking head on web pages* to guide a user through the product selection process, which is only one example for interactive video, a tool for enhanced user convenience that companies selling over the Web or T.V. could use ; and finally, *video telephony* and *video conference* as examples for an online version of OBC.

Appearance based methods require that the video input stream be normalized to a principal object. We present in this dissertation a new, low-complexity face tracker based on color detection and tracking. This tracker yields a stabilization of the input data sufficiently precise to enable OBC. However, as results suggest, a refinement of the tracker, possibly enhanced by other modules than color detection, could further contribute to the efficiency of OBC.

A particular problem encountered when evaluating OBC is that its compression artifacts are substantially different from fixed-basis compression schemes such as MPEG. Most currently used reconstruction quality measures do not reflect human visual perception. This work therefore underlines the necessity of developing reconstruction quality measures based on the human visual system. This requires collaboration with psychologists, whose input is invaluable for the design of a test series to quantify human visual perception in order to eventually create a mathematical description of human image quality rating.

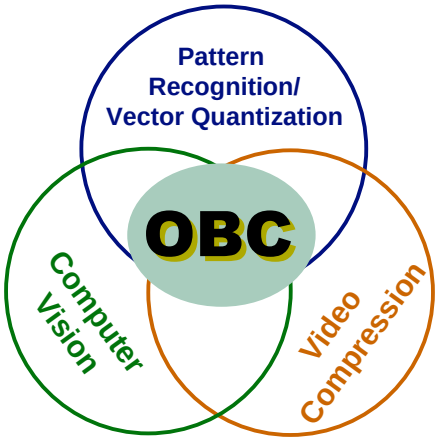


FIG. *2.4: Localization of OBC at the intersection of Video compression, Computer vision, and Pattern recognition.*

Fortunately, even with the drawbacks of using conventional image quality measures such as the *Peak Signal to Noise Ratio (PSNR)*, OBC performs well compared to MPEG in terms of reconstruction quality and compression ratio. If the prerequisites for OBC such as precise tracking are met, then OBC easily outperforms MPEG compression[5] in terms of quality and compression. Solely w.r.t. computing power consumption, OBC is falling short of MPEG. This is mostly due to non-optimized code and complex algorithms. OBC as it is presented in this dissertation is basically an offline process, even though considerations for an incremental version are documented, and a series of – not included – experiments showed promising results.

2.2 Outline of dissertation

Chapter 3 outlines the technical and scientific context of this work, which analyzes the contribution of appearance-based computer vision methods to the compression of video data. Appearance based methods generally require object tracking as a prerequisite, and we define three categories to sort out the plethora of computer vision tracking algorithms. A selection of the currently most important ones is used to illustrate the usefulness of such a categorization. After showing that a feature-based color tracker is sufficient for the intended application, we turn our attention

[5] Reference encoder was the Berkeley `mpeg_encode` program in various configurations [Ber97]

to video compression. A short history of today's most important video compression algorithms is given followed by an outlook on current and future developments.

The theoretical background of this work is reviewed in **chapter 4**. Step by step classification criteria for the ingredients of a video codec[6] are derived from information theory. These classification criteria allow for an analysis of current existing video codecs and support the synthesis of new techniques for video compression. Systematically the compression algorithms relevant for this work are developed out of these classification criteria, illustrating their relation to each other as well as their strengths and weaknesses.

Using the results from chapter 4, we briefly discuss the most important compression techniques such as MPEG, H.263, JPEG, and ZIP compression in **chapter 5**. These techniques provide a benchmark for OBC, or are implemented within OBC. We see how techniques from the different classification criteria are combined to form efficient codecs for special applications. An outlook on current developments highlights their limits and motivates again this work.

We elaborate details of the techniques developed for this work in **chapter 6**. A feature-based skin color tracker is motivated and designed. We show how color histograms can be used to localize and track objects. In a first step the histograms are used to generate a probability map for skin colored pixels using Bayes' Rule. An efficient tracking algorithm based on the first and second moments of that probability map is suggested. This algorithm, called CENTEROFGRAVITY algorithm, is inspired by robust statistics eliminating outliers through weighting of new input data. The CENTEROFGRAVITY algorithm also employs weighting and motion compensation inspired by the concepts of Kalman filtering. The result is a robust color tracker which outperforms conventional Threshold/Connected Components/Kalman filter tracking algorithms in terms of precision while being of comparable computing performance.

Chapter 7 develops the *Orthonormal Basis Coding (OBC)* algorithm. The concept of OBC, which relates the algorithm to its intended applications, is described. Since the way data is processed by an algorithm is closely related to how the data is perceived, we compare OBC and conventional video compression by the data model used. The implications of seeing images as vectors instead of decomposing them into blocks are discussed. One of those consequences are that vector quantization or clustering algorithms become applicable. Some important clustering algorithms are presented and their impact on reconstruction quality and computing time illustrated. Finally, because of their importance and frequent use, the most common reconstruction quality measures such as MSE, SNR, and PSNR are discussed in detail. They are interpreted in the context of image and video processing and their limitations are shown.

Chapter 8 presents results that show that OBC is a valid alternative for standard and other video compression algorithms. It is particularly efficient if the input data is well normalized to an object as in this case a talking head in a head-and-shoulders scenario. Encoding results are evaluated for seven sequences, four of which are standard sequences used by the video compression research community, and three other sequences created in our laboratory. It is shown that in all cases OBC produces results comparable to an MPEG encoder representing the widely used

[6] codec = *coder - decoder*

DCT/DPCM compression schemes. In some cases, characterized by a well-normalized video sequence, OBC significantly outperforms the benchmark MPEG codec in terms of quality and/or compression. However, OBC is less performant in terms of computing power. Since the OBC encoding software has not been optimized with respect to computing speed, comparative results are given using computing time.

The OBC codec presented in this dissertation should be seen as a first feasibility study for a video codec based on appearance of objects in a video stream, instead of using statistical correlation between its pixel values. There still is a lot of room for conceptual as well as implementational improvement, and **chapter 9** gives an outlook on these issues.

Chapitre 3

Technical and Scientific Context

This chapter outlines why it is interesting to reconsider the problem of efficient compression of video stream data. Although computer and communication hardware and software have considerably improved over the last few of years, there still remains (and probably will always remain) a need to compress large amounts of video data. Historically, whenever a communication channel has developed into a powerful tool, another technical possibility emerges to renew the demand for low-bandwidth communication. Digital television or *Integrated Services Digital Network (ISDN)* telephone may serve here as examples [1]. ISDN was designed to enhance telephone communication with different sorts of data transmission services. The increased bandwidth inspired engineers to use ISDN lines for video telephone and internet access. However, such services need even more bandwidth to be truly satisfactory to an end-user. In some places cable television networks are therefore used for internet access, and UMTS targets 2 Mbits/sec transmission bandwidth.

A variety of techniques exist or are being developed in order to meet the demand of low bandwidth communication. Techniques employed in current video compression standards as well as in proprietary video codecs are based on statistical correlation between pixel values, in the image plane as well as on the time axis. In other words, they are based on the fact that the images to be compressed have a *content*, but they make no assumption about what that content is, nor do they extract information about the image content. If an image contained only uncorrelated pixel values, i.e., noise, those techniques lose their effect.

Over the last ten years the video compression problem has not been re-thought *as a whole*. Research in video compression has experienced a kind of stagnation and rather focussed on the refinement of already available techniques than looked for new approaches to the problem. Symptomatic is the fact that the video compression layers of MPEG-4 and MPEG-7 are still the algorithms of MPEG-1 and MPEG-2 [Cha95a, Cha95b], respectively. On the other hand, the field of computer vision has seen remarkable progress, and a large set of reliable techniques have been developed,

[1] ISDN is defined by the I.xxx series of ITU-T recommendations. `http://www.itu.int`

some of which are presented in section 3.2. There is a general agreement that new video compression algorithms should make use of computer vision methods such as segmentation, feature extraction, and 3D modeling, as a preprocessing step for subsequent data compression [HL96].

The idea is that once the video image content has been decomposed, identified and modeled, only its parameters such as size, position, and orientation, need to be transmitted or stored, e.g., in the form of a low-dimensional parameter vector. This approach is called *Second generation video coding*. It assumes, however, that the scene in the image does not change too much, since such an approach requires an appropriate model for each new scene. The subsequent encoding after segmentation is done with conventional data compression techniques such as described in chapter 4. Such approaches are incorporated into new coding and representation standards as ISO MPEG-4 [ISO96b, BCL99, BCL00] or ISO MPEG-7 [ISO00b, NL99a, NL99b].

The success of an approach using segmentation and models is very dependent on the quality of the model. Good models are computationally costly, and the initialization of feature tracking algorithms for good model matching is often done by hand. A promising approach for automated feature tracking using the Hough-transform is currently being developed at the University of Manchester [CWT00]. Yet, efficient model-based video coding, especially in real-time, remains unavailable.

Let us summarize the state of the art about video compression in a few points before we define the goals addressed in this dissertation.

⇒ Current video compression techniques are based on statistical correlation of pixel values, and decomposition in frequency space (DCT, FFT, Wavelets)

⇒ The current paradigm of second generation video coding requires segmentation and content extraction

⇒ No automatic model extraction is yet available.

⇒ Mapping onto models is slow, computationally costly, and unstable.

Our goal is to develop an efficient representation of video data implicitly allowing us to gather information about the video image content. Extraction of image content is the currently accepted paradigm to improve video compression over purely statistical methods. Lacking a more convincing paradigm, we will employ a data representation where we can identify image content. We reject, however, the use of geometric models. Such an approach does not appear to be flexible enough to be used in varying communication situations, it is computationally too costly, plus it requires us to make assumptions about and compute *non-visible* image content, which may result in an unnecessary overhead.

Our technique should therefore :

a. Extract image information,

b. Use only the information supplied by the images,

c. (optionally) makes use of available compression methods and other efficient data representations,

d. Yield a compression ratio higher than that of *"classical"* compression algorithms like those in section 3.3 while maintaining a comparable reconstruction quality.

Section 3.1 gives a description of our approach to video coding as we have realized it. This approach uses tools from computer vision and from data compression techniques designed for video coding. Section 3.2 gives an overview of computer vision techniques with the focus on tracking, since tracking is pre-requisite for OBC. The competition of OBC is then presented in section 3.3, while section 3.4 concludes this chapter.

3.1 What you code is what you see

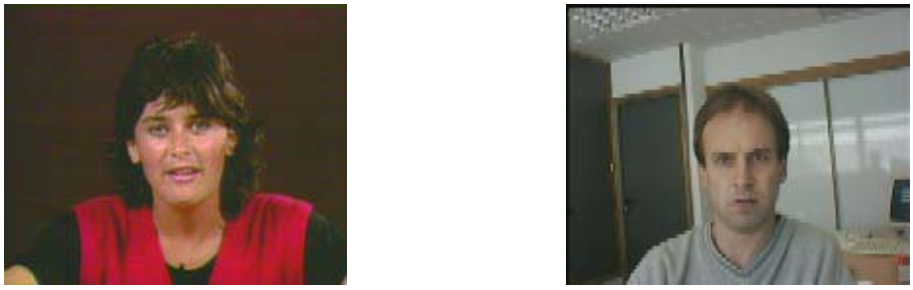
Extraction and identification of image content falls into the domain of computer vision. In order to extract and identify image content, we have to make assumptions about that image content. An automatic creation and update of a knowledge database of image content would imply the use of Artificial Intelligence methods. Such an approach, however, would go beyond the scope of this work. Sole alternative to automatic image content extraction and making assumptions about the image content would be "blind encoding", i.e., fall back on purely statistical methods.

Making assumptions about the content of a video stream is identical to defining a scenario which we design our coding method for. The scenario chosen for this work is the classical one-to-one video communication scenario, that is, a single person talking with only his head and shoulders shown. A major difference between our approach and conventional model-based approaches is that we only use *visible* image information. Hidden information like color, texture, and shape of a talking head is unimportant, as long as the communicating person does not expose it to the camera. Two examples for a head and shoulders scenario are shown in figure 3.1 a) and 3.1 b), which is the same scenario assumed for the development of the ITU-T video compression standards (see subsection 3.3.2). Compression requires finding a more compact representation of data than its original form. Chapter 4 discusses general aspects of data compression with emphasis on techniques which are relevant for this thesis, either as techniques employed or as benchmarks. In many cases, an assumption about the data to be compressed improves the efficiency of the compression algorithm.

Having outlined our application area puts us in the position to look for appropriate techniques. We will do that in the following sections by giving overviews of the fields of computer vision and of video compression, identifying the starting points for this work. The following chapters will then thoroughly describe the realization steps and what results we got and can expect for the future.

3.2 Computer vision techniques for tracking

Computer vision spans a number of areas including 3D-scene (re)construction, indexing of video databases, object tracking and recognition, motion detection and estimation, and image synthesis. A number of robust methods and algorithms have been developed to fulfill those tasks. The



a) b)

FIG. *3.1: Two examples for a typical video communication scenario : a) from the popular* MISSAMERICA *sequence, b) from the* TALKINGHEAD *sequence*

computer vision field that is of most interest for us is object tracking. Object tracking has a number of applications, some of which are surveillance, robot navigation, and scene interpretation.

A tracking algorithm involves a trade-off between precision, speed, robustness, and the extraction of image information such as orientation, size, shape, texture of an object. The reason is that the more detailed the information which is desired from an image, the more complex are the algorithms to obtain that information. For instance, detection and tracking of the approximative position of an object can be done with low complexity algorithms such as correlation or color tracking, followed by a filter stage, e.g., a Kalman or Condensation filter. Information about the object surface, on the other hand, may involve 3D texture mapping or frequency or Eigenspace mapping, which require more computational operations per pixel. Depending on the intended application and its requirements, a selection of desired properties of the tracking algorithm may be determined, and the tracker may then be composed from techniques out of the computer vision tool-box. We use the following classification, inspired by Isard and Blake [IB98], to get an overview of what techniques exists. A very similar categorization of computer vision tracking techniques is used in [Bre99].

Feature-based tracking focuses on object properties or features rather than an object's type. It includes simple, reliable, fast, and robust techniques such as color detection, correlation, histogram matching, blob tracking, corner tracking, region matching, edge and contour tracking, or image differencing. Their design is independent of the object being tracked. This category roughly corresponds to what is otherwise called *Feature tracking* [Bre99] or *Low-level tracking* [IB98]. Combinations of Feature-based tracking techniques are often integrated into a multi-modal tracking system to increase robustness.

Model-based tracking covers tracking algorithms where *a priori* assumptions about the tracked object are made and integrated as a model into the algorithm design. This category is also called *high-level tracking*. Examples are wavelet templates [KHS99], 2D, $2\frac{1}{2}$D, and

3D models [HN98], view-based representations [HHD98], and appearance-based representations [BJ96]. Just as this work, the latter are inspired by the work of Turk and Pentland [TP91].

Multi-modal tracking systems are hybrid tracking systems which require sensor module output fusion, possibly with a supervisor controlling the sensor data fusion. It is sometimes called *Multi-stage tracking* [Bre99] and makes use of several tracking modules, possibly from feature-based and model-based techniques at the same time. They often switch between low-level and high-level modules in order to increase robustness. The fusion of the sensor output is commonly done with estimation filters such as the Kalman or the Condensation filter. Examples of such systems are the SOSICOM[2] multi-modal tracking system reported in [CBC97], ICONDENSATION [IB98], or W^4S, the latter is a real-time system for detecting and tracking people [HHD98].

Table 3.1 contains a checklist for the identification of tacking categories.

TAB. *3.1: Properties of tracking categories*

Category	uses *a priori* knowledge	number of features tracked	tracking module output fusion necessary
Feature-based	no	1	no
Model-based	yes	≥ 1	no
Multi-modal	optionally	> 1	yes

Knowing about the advantages and disadvantages of a technique allows for a good judgment about its possible application areas. This may be important when building or integrating tracking systems. Tracking is, of course, possible with techniques other than computer vision. Tracking by localization of a sound source [GOSC00] or by using laser range data [CWS98, Wal97] can well support or even replace a computer vision algorithm. The following subsections review the most widely used approaches. Details about their realization are either given in chapter 6, or in the references mentioned for tracking based on computer vision methods.

We can anticipate here that the complexity of model-based tracking makes it inappropriate for our purposes. Instead, we use a feature-based tracker using color. As mentioned in the conclusions to our experimental results, it may be desirable to add modules such as background subtraction to our face tracker in order to increase its precision to sub-pixel range. Of particular interest is the exact outline of the tracked head to offset scale changes. Such an extension would make our face tracker a multi-modal.

[2] *Suivi d'Objets par le Son et l'Image pour la COmmunication Médiatisée (SOSICOM)* [KP96]

3.2.1 Feature-based tracking

This category contains simple, but often robust techniques and algorithms, some of which have been used for a long time because of their low computing complexity. Many of the algorithms detecting features are relatively insensitive to appearance variations by view or illumination changes. Moreover, due to their low complexity, feature-based techniques allow for real-time implementations. Their major drawback, however, are their dependency of the actual image content, which makes them fragile w.r.t. even partial occlusions. That is, without a higher-level decision module such as a Kalman or Condensation filter, a tracking system will fail if the tracked feature cannot be detected in the current image. The following overview of feature-based techniques shall give the reader an impression what is available rather than being a comprehensive list.

Color is a commonly used feature used for tracking. For the purpose of face tracking, skin color is a particularly interesting feature to track. Skin color is usually relatively unique within most environment such as offices. We will introduce in chapter 6 a tracking system based on color histograms employing an estimator inspired by robust statistics. We will show that this simple, solely feature-based tracking system [SC00] is sufficiently fast, precise, and robust to enable OBC. A weak point of color tracking is certainly its sensitivity to changing lighting conditions.

Correlation or region matching are basic techniques where two images are compared and areas out of the images compared using some kind of distance metric. Its biggest advantage is its simplicity and thus low complexity. It is, however, sensitive to noise and scale as well as variations of shape.. Correlation or region matching should therefore only be used with caution [CBC97, SCD98].

Optical Flow is a technique to detect motion fields within an image. The idea is to detect entire regions of pixels which move from one image to the next in the sequence. Such regions may be identified as objects moving within an image [TLS93]. Major problem of optical flow algorithms is their sensitivity to changes of illumination. There has been some work on how to address this problem. For instance, Black et al. [BFY98] give a framework to model object appearance changes to offset illumination and iconic changes.

A tracking algorithm based on skin color detection, but using offline sampled pixel value distributions for skin colored pixels is called **blob tracking** [JP99]. Using 3D Gaussian mixture models, 2D blobs (elliptic areas) are generated. The involved *pdf*s are estimated using the *Expectation Maximization (EM)* algorithm [DLR77].

3.2.2 Model-based tracking

For some application it may be desirable to track of object properties which are not accessible without prior knowledge about the object. Contours, shapes, and textures are examples of such properties. Model-based techniques, as the name indicates, try to map image information onto a prior created model, such as a wire-frame model of a head or a car. These are fairly complex operations which may make use of one or more feature-based tracking algorithms. Its complexity

makes model-based tracking relatively robust to noise and occlusions, but compromises seriously its applicability in real-time situations. Moreover, in applications such as face tracking, significant changes of object appearance (wear a scarf or a hat, long beard) can lead to a poor tracking performance.

3D-Models represent an object in 3D geometrical data, for instance as a volumetric or wire-frame model. Based on *a priori* assumptions about the object to be tracked, e.g., a head or a hand, such a 3D geometrical model is created. During the tracking process, the image content is segmented and mapped onto the 3D model. 3D models can handle partial occlusions and are particularly popular in applications tracking cars and walking people. Fundamental problem of tracking using 3D models is the computational complexity with prohibitively high cost even for very simple car models [KDN93].

Eigentracking was introduced by Black and Jepson [BJ96]. It combines eigenspace techniques, parameterized optical flow, and robust estimation techniques. Rather than try to represent every possible view in the eigenspace, or learn surfaces in the eigenspace that interpolate between views, they represent views from only a few orientations. Objects in other orientations are then recognized by recovering a parameterized transformation (or warp) between the image and the eigenspace. The eigenspace itself provides a representation (i.e., an image) of the object that can be used for tracking. Tracking is performed by using a robust parameterized matching scheme, where objects may undergo affine image distortions and changes of view.

Active contours such as *snakes* use *a priori* knowledge to interpret the output of low-level image operations such as convolution [BI98]. It is an approach to track geometric structures such as object outlines. A *snake* is a deformable curve in a feature map generated from, e.g., an image with a edge detection filter. An equilibrium condition for this deformable curve makes it cling to high responses of the feature map. However, more prior knowledge about the object being tracked is needed to make such a system run successfully in real-time, which is equivalent to making this approach model-based.

Wavelet templates can be used to represent arbitrary objects. For instance, they have been used to track faces [KHS99]. A face template is represented by a set of weighted wavelets. Tracking is then performed by detecting a template in subsequent images, which for that are represented by a wavelet network.

3.2.3 Multi-modal tracking

Multi-modal tracking systems are identified by their need to fuse the output of several object detecting sensors out of the tool-box of feature-based and/or model-based tracking algorithms. Depending on the intended application, a selection of individual tracking algorithms are combined in order to increase robustness or to track certain features. Additionally, this allows to trade-off computing speed for precision or other object information such as the posture of a human body.

The Multi-modal face tracker introduced in [CBC97] was an effort to build a multi-modal face tracking system based on several complementary, feature-based modules. A correlation tracker, eye-blink detector, and a color histogram tracker based on a connected components algorithm were alternatively used and their outputs combined using a Kalman filter. While the architecture proved to be adapted for robust tracking, the individual modules were too sensitive to changes in the experimental setup.

Multi-modal tracking is well suited for surveillance tasks. An enhanced, multi-modal people tracker based on background differencing, motion history detection, and color tracking is currently build in our group. This system proves robust to even the most saccadic movements and runs in real-time at video rate. It further allows the seamless integration of an event detector and a recognition module. Figure 3.2 shows its current architecture as a nice example of a multi-modal tracking system.

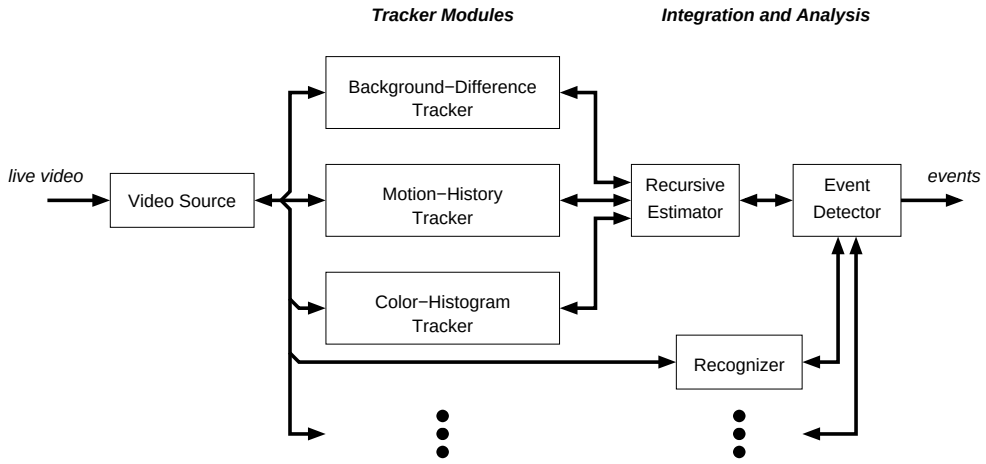


FIG. *3.2: Architecture of the robust multi-modal tracker, and examples of possible extensions (parts in light blue are to be implemented yet). [diagram courtesy J. Piater]*

W^{4}S is a nice example of integrating feature-based and model-based techniques into a multi-modal tracking system [HHD98]. It tracks people and monitors their activities in an outdoor environment. *W^{4}S* uses a shape of stereo and shape analysis, creating models of peoples' bodies in order to be able to track them even in the case of occlusions. The models are simple ellipses for head, hands, feet, and torso.

ICondensation combines high-level and low-level tracking techniques into a probabilistic framework [IB98], using the statistical technique of importance sampling combined with the CONDENSATION-Filter [IB96]. Importance sampling offers a mathematically principled way of

directing search, combing prediction information based on the previous object position and motion with any additional knowledge which may be available from auxiliary sensors. The ICONDENSATION-tracker uses color segmentation to find skin-colored blobs in a sub-sampled image, and feeds this information to a contour tracker specialized for hands (or any other object).

3.3 Overview of Video compression techniques

We can divide video compression methods into three categories : proprietary, standard, and under development. The first hide their actual implementation, so we can give here only an overview over the most important or widespread ones. The second and third are sketched below. This section gives an outline of what is available and used today with references either to relevant literature, or to the chapters and sections in this thesis where the topic is treated further.

3.3.1 Proprietary Video Compression Technology

Two of the more important proprietary video compression techniques are Cinepak from Compression Technologies Inc. (CTI), former Radius Inc.. This technique is used in devices such as Camcorders. Another proprietary video compression technique is RealVideo from Real Networks, Inc. This is currently very widespread for use over the internet. T.V. stations, for instance, use RealVideo to broadcast their emissions over the internet, since it is especially good for low-bandwidth channels. Both of these companies have an interest in keeping their code secret. However, from the artifacts of Cinepak and RealVideo we can conclude that they are based on the same DCT/DPCM encoding scheme as the standards described below.

3.3.2 Video Compression Standards

History

The year 1990 was an important landmark for the video communication research community and industry. It was the year, when the Telecommunications sector of the International Telecommunications Union issued the final draft version of its recommendation *"Video Codec for Audiovisual Services at p x 64 kbits"* [ITU93], creating the first independent *de facto* standard for video compression [Lio91]. Coincidentally, this was the same year as for the invention of the HTTP protocol. Prior to this date, proprietary standards such as the *run-length-encoding* option of the Microsoft AVI file format were used, or the still picture compression standard, defined by the *Joint Photographic Expert Group (JPEG)* of ITU and ISO was widely used for compression of video stream data [ISO94]. We will come back to this standard and its usability for video compression later when we briefly talk about JPEG compression.

The ITU-T recommendation H.261 targets the coding of low-motion video data over p x 64 kbits/s ISDN communication lines, summarizing the research efforts up to that date. The *Moving*

Pictures Expert Group (MPEG) of the ISO quickly adopted the techniques as described in H.261 and expanded them for full-motion video compression for up to 1.5 Mbits/s enabling features such as reverse playback. Also in 1993, they issued the ISO standard 11172 *"Generic coding of moving pictures and associated audio information"* whose second part describes the video compression algorithm [ISO93]. This standard was later adopted by the ITU-T to become recommendation H.262. MPEG-1 was designed for efficient storage and retrieval of full-motion video. MPEG-2 was designed in terms of extensible profiles, each supporting the feature of an important application, i.e., digital video transmission over cable satellite, and other broadcast channels.

With the widespread use of ISDN lines and the arrival of the world wide web plus the developments for digital television and satellite techniques, the shortcomings of those standards became evident. Incorporating some additional techniques, ISO and ITU-T addressed the need for more refined video compression standards in 1996 with the issues of the ITU-T recommendation H.263, *Video coding for low bit rate communication* [ITU96], and the ISO standard 13818, *Generic coding of moving pictures and associated audio information* [ISO96a] for up to 10 Mbits/s.

All of these standards and recommendations, H.261, MPEG-1, H.263, MPEG-2, and JPEG rely on classical statistical methods with the three-step encoding scheme [Gal91] : 1) *energy compaction* by *Discrete Cosine Transform (DCT)* , 2) *entropy reduction* by *Differential Pulse Code Modulation (DPCM)* , and 3) *redundancy reduction* by *Run-length encoding (RLE)* , plus optional pre-processing steps such as motion-detection. There have been numerous papers about how to implement those standards in software and hardware, and how to improve their efficiency with pre-processing [] and post-processing [OBK95]. Chapter 5 will cover those techniques in detail.

More recent work of the standardization organizations lead to the MPEG-4 and MPEG-7 technical standards, which will be introduced in the following subsections. The curious numbering of those standards is of technical origin : A project MPEG-3 was dropped when it became clear that MPEG-4 would fulfill its tasks both project were started about the same time. Looking for a new ordinal to name future projects, 5 and 8 as the logical sequence were rejected and 7 was arbitrarily chosen. Here is an excerpt from the official MPEG-7 FAQ to the MPEG numbering problem [MPE01] : *"So after 1,2 and 4, there was much speculation about the next number. Should it be 5 (the next) or 8 (creating an obvious binary pattern) ? MPEG, however, decided not to follow either logical expansion of the sequence, but chose the number of 7 instead. So MPEG-5 and MPEG-6 are, just like MPEG-3, not defined."*. Remains the question why "1, 2, 4, 6" is less a logical sequence than "1, 2, 4, 7" ? The latest coup of the MPEG committee is called MPEG-21, and it addresses multimedia frameworks.

Technical standards, if widely accepted, ensure the interoperability of software and hardware from different manufacturers. There are numerous examples of wasted effort and money on a big scale in order to push a proprietary technique to a world standard : VHS (Sony) versus Betamax and VCC (Philips) analog video recording, *Digital Compact Cassette (DCC)* (Philips) versus DAT (Sony) and Minidisc (Sony), HDTV versus digital and analog television, various incompatible mobile phone communication networks in the U.S. Technical standards make life easier and cheaper for almost everybody. This is why committees for the DVD storage media has about 200, and for the CORBA software specification about 800 member companies worldwide.

From a research point of view, however, technical standards are – in the best case – a snapshot of the *state-of-the-art* at a given time. As an example, requirements for the performance of the employed algorithms are certainly related to the hardware equipment they were developed on. Their practical application field is future hardware equipment, often – such as for video telephones – with dedicated devices, e.g., DSP instead of FPGA. Motion detection and prediction is a good example for a tentative anticipation of the power of future hardware generations. Early hardware implementations simply ignored that feature because it would have made the encoding performance unpredictable.

Today's video compression standards, even those being currently developed, still use compression techniques from the early nineties. The apparent lack of interest of the research community in fundamentally new approaches to video compression is difficult to explain. Maybe computer vision techniques beyond motion detection are considered too unreliable to be a serious preprocessing step for video compression. Yet there are some attempts to tackle this problem, and below is an overview. The fundamental problem is connected to the fact that the current paradigm is based on (3D-)models for the image content, and indeed in that field there is a lot to be done and powerful computers are needed for a smooth implementation.

Recent work

MPEG-4 The MPEG-4 standard [BCL99, BCL00, ISO96b] of the ISO was raised from committee draft to international standard in April 1999. It was designed to encode (and decode) multimedia content, not just video and audio data. It further permits protection of owner rights, and allows for user interaction during the encoding and the decoding steps. The video codec uses the same 3-step algorithm as MPEG-1, H.263 etc., but it allows for the separate encoding of objects in an image. This is more or less equivalent to the model-based approaches discussed below. However, the extraction of that object information is left to the user.

MPEG-4 draws from VRML, and targets similar (or even the same) application areas. This is not a lack of originality but an intentional feature. Standards do not seek to generate new techniques, but use the supposedly *best* techniques and make them available to everybody.

MPEG-7 [NL99a, NL99b] is not (directly) a further development of MPEG-4, but targets different application areas. While MPEG-4 is designed for multimedia production, distribution, and content access, MPEG-7 [ISO00b] targets the *description* of multimedia materials. It addresses the interoperability and globalization of data resources and management. MPEG-7 standardizes the structure and linking mechanisms for image (video, multimedia) content as well as the representation of that content. It intends to make audio-visual material searchable like text, which will be a great improvement for, e.g., internet search engines which today are still text-based only.

MPEG-7 will be a standardized description of various types of *interfaces* of multimedia information. This description will be associated with the content itself, to allow fast and efficient searching for material that is of interest to the user. MPEG-7 is formally called *"Multimedia Content*

Description Interface." The standard does not comprise the extraction of descriptions/features nor does it specify the search engine or any other program that can make use of the description. MPEG-7 is intended to reach the status of a Draft International Standard by July 2001, and of an International Standard by September 2001.

3.3.3 Second Generation Video Coding

The major difference between first and second generation video coding is that second generation video coding uses information about the image content. This is the paradigm under which research in video coding is done, this thesis included. MPEG-4 offers possibilities to encode a variety of objects However, it says nothing explicit about the extraction of such objects [Sik97]. Creating models or trying to segment image content into objects as a pre-processing step, and doing conventional waveform-based compression afterwards is the common approach. This separates the extraction and the compression part. Since the compression algorithm still consist of DCT, FFT, or Wavelet transform, DPCM or vector quantization, and finally a variation of VLC, this approach has choked research in video compression and shifted the attention to extraction, i.e., computer vision techniques. On of the major contributions of this thesis is that our coding approach implicitly includes the use of image content information into the encoding algorithm. We renounce the standard dual approach and rather promote and integrated coding approach, doing image content representation and compression at the same time.

Model-based approaches

Research in second generation video coding is basically about creating 3D models of image content, notably of heads [Eis00]. This reduces the its applicability to scenes with objects of which there is a model at hand. Of course, we are talking about advanced wire-frame models such as in [Eis00, 3D-01], not simple contour models as in [HHD98]. Anyhow, since model extraction is a complex process, the focus of model-based coding research is focussed on the head-and-shoulders scenario. This is supposed to be the scenario for the most interesting application area, video communication, as has been outlined in the introduction chapter.

It is interesting to note that the initialization process and the transmission of the model is *not* necessarily included in the compression results of model-based approaches [Eis00]. In any case, the encoding procedure – after initialization and transmission of the model – consists of estimating the *Facial Animation Parameters (FAP)* for the model, and optionally eliminate redundancy in the output parameter stream. MPEG-4 defines a set of FAPs [ISO96b].

Another feature of MPEG-4 is that it uses a layered representation of video data. This corresponds to considering a video frame as a superposition of several layers, which comes handy when encoding text, graphics, and scene data at the same time. A particularly interesting technique in that context is the use of *sprites*. A sprite is defined as *[...] a large static image composed from the pixels in an object visible through the entire scene.* [LCL+97]. This may be useful when building up a background layer while a recording camera is moving. The question is if this is still efficient when a changing camera zoom for instance has to be compensated for by scaling.

3.4 Conclusion

We saw in this chapter why it is interesting to reconsider the problem of compressing video data. The idea behind the approach to second generation video coding was discussed, and we laid out why we did not choose an approach based on 3D models. Instead, we consider it more efficient to take an appearance-based approach, concentrating on visible image information and not care about not visible image content. The prerequisite of such an approach is a good normalization of the video data to one principal object, and in this regard appearance-based coding resembles any other model-based approach. Normalization, anyhow, requires the interference of computer vision methods, notably tracking. In order to do this, we divided the huge number of computer vision tracking algorithms into three categories, and selected a feature-based color tracker as our face tracker.

Chapitre 4

Data compression

Video compression is not a one step process. A selection of compression algorithms is used such that their combination exploits the limitations of the human visual system. Although data compression techniques can be very different from each other, we can divide them into groups of fundamental approaches. The classification we use is depicted in figure 4.1, which may also serve as a road map through this chapter. It allows us to compare our own approach to currently existing methods, as well as to identify possibilities for improvement.

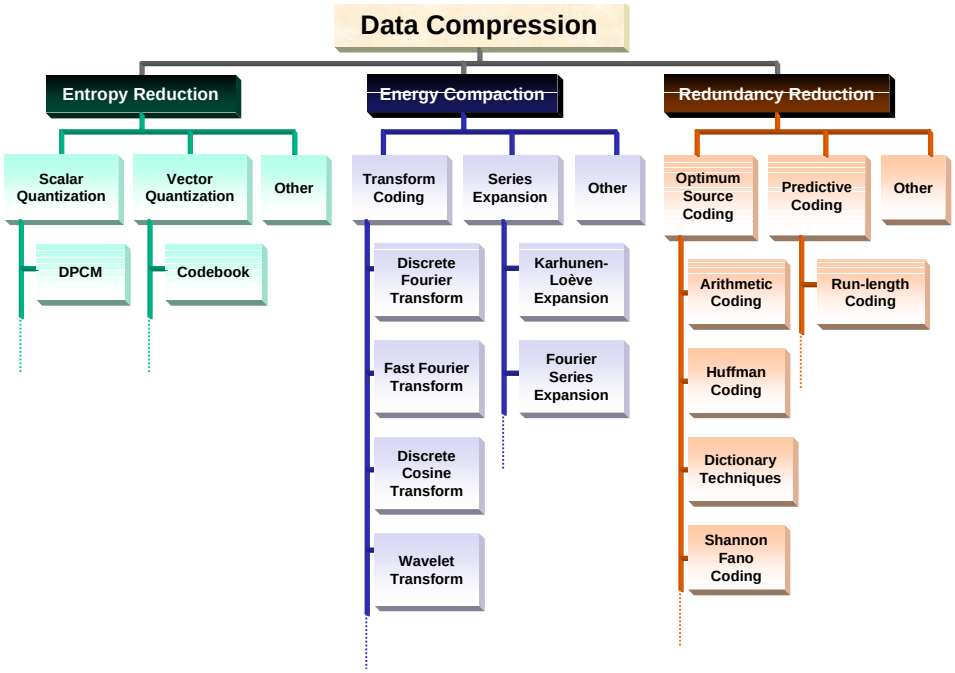


FIG. *4.1: Classification of Data Compression Techniques*

We will see in later chapters that both the choice of the compression algorithms based on

their properties and the order in which they are used are important. This chapter gives a general overview of data representations and coding. Both are discussed in a general manner to show how they are related to this work. Illustrating their limitations leads in turn to the motivation for our own approach.

The classification criteria used in figure 4.1 refer to concepts from information theory. Section 4.1 reviews from information theory fundamental ideas :

Redundancy reduction , which is also called *entropy coding*, is equivalent to searching an optimum data representation without information loss. Some of the most important redundancy reduction algorithms are discussed in section 4.2. They are important for this work for two reasons : we use them to optionally compress the basis vectors of the *Orthonormal Basis Coding (OBC)* algorithm, and they are used in video compression algorithms we use as benchmarks such as MPEG. Redundancy reduction is a *lossless* and therefore *reversible* operation.

Energy compaction is the redistribution of a signal/image energy by changing its representation from one linear space to another. This may be done using some linear transform or series expansion. It can be shown that for most images the major share of energy is concentrated in very few data points of a transformed image (see figure 4.7, and also [Fur94]). This allows for omitting many transform coefficients at coding time. The degradations in the reconstructed image introduced this way are either not perceptible or acceptable by the human visual system.

Section 4.3 gives an introduction to efficient signal representations. The discussion is on a fundamental level, which permits illustrating the link between the OBC approach and other approaches. Furthermore, techniques we use for OBC such as the *Karhunen-Loève Expansion* and the *Gram-Schmidt orthonormalization procedure* are introduced.

If a complete orthonormal basis were used to linearly transform an image, the operation would be *reversible* and no data loss would occur. This would hold even for discrete linear transforms. In reality the number of basis dimensions is limited (usually to 8), and an error is introduced each time an image is transformed. Note also that energy compaction does not necessarily reduce the amount of data. This can be done by subsequent redundancy and/or entropy reduction steps.

Entropy reduction reduces the amount of information of a data source. Compression by entropy reduction takes information loss into account in order to increase compression. This makes the encoding process *irreversible*, and the output of the decoder is just an *approximation* of the original input of the encoder. If it is possible to make sure that the information lost at compression time is not needed or can be well estimated, very high compression rates can be the result.

The most popular entropy reduction technique is quantization. The art of quantization is to identify sampling clusters representing the original data in a way which for the human eye is indistinguishable from the original. Section 4.4 covers some important quantization techniques, which are relevant for this work either being used in benchmarking algorithms such as MPEG, or as an inspiration for OBC coding.

4.1 A Review of Information Theory

While many of his concepts were already "in the air" at that time, it is generally agreed that *Information Theory* was founded by C.E. Shannon in 1948 with the publication of his paper *"A mathematical theory of communication"* [Sha48]. Structuring a general communication system into the components as depicted in figure 4.2, he developed a theory for each of the components shown. This thesis focuses on the first component, the information source, and – more specifically – an information source of images. We briefly sketch in this section why it is possible to compress data coming out of an information source and introduce the three classification criteria as shown figure 4.1.

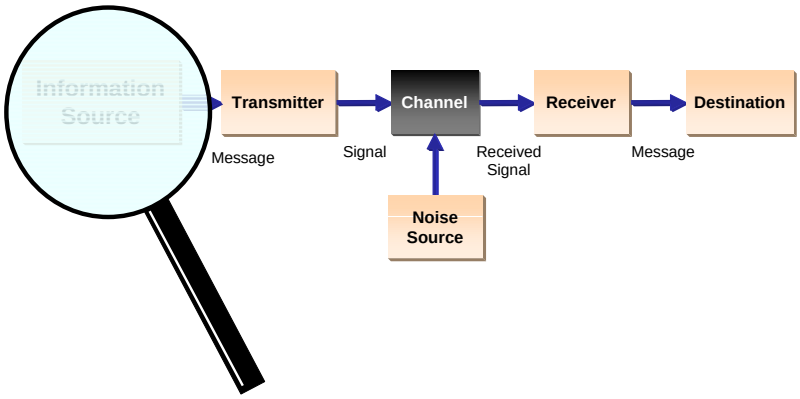


FIG. *4.2: Shannon's schematic diagram of a general communication system*

4.1.1 Entropy, Information

Information theory is primarily interested in channel capacity and efficient, error free transmission over noisy channels. Looking from a channel into an information source, we want to know if and how we can measure then information coming out of the source. For this section, we will use the word *"message"* to describe the data of an information source. Messages can be letters of an alphabet, symbols, numbers, or some other sort of data units. In the general context of this work, the messages in question will be sequences of pixel values. An information source is assumed to have a finite set of such messages, the *alphabet* of the source.

We can actually express the information content of messages coming out of a source independently of the character of the source (sequence of letters, signal (time) functions,etc.). Assuming a source with a set of n different messages and their probabilities, p_1, p_2, ..., p_n, Shannon formulated

in [Sha48] three properties for a measure of average information, which he denoted by H, should have :

a. H should be continuous in the probabilities, p_i, of the single i messages.

b. If all the probabilities p_i are equal, $p_i = \frac{1}{n}$, then H should be a monotonic increasing function of n. In other words, with equally likely events there is more information when there are more possible events.

c. If a message is composed of successive messages, then the H of the composed message is equal to the weighted *sum* of the individual messages.

Theorem 2 in [Sha48] states that the only measure satisfying these three conditions is

$$H = -K \sum_{i=1}^{n} p_i \log p_i, \tag{4.1}$$

where K is a positive constant. H is called the **entropy** of the set of possibilities p_1, p_2, ..., p_n, where p_i is the probability of the ith message from the source.

Note that the base of the logarithm in equation 4.1 can be chosen freely. If the base of the logarithm function is 2, the resulting unit is called *binary digits*, or *bits*, a word which was first suggested by J.W. Tukey[1]. If the base is e, which might be practical if mathematical operations such as integration and differentiation are involved, the resulting units are called *natural units*, or short *nats*. For a base of 10, the name of the units is *hartleys*, after R.V.L. Hartley, who first suggested the logarithm function to be the *"most natural choice"* for measuring information [Har28]. Since $\log_a x = \log_a b \cdot \log_b x$, any of those units can be scaled to another by a constant factor. We will therefore omit the factor K when we talk about entropy.

If the entropy is the average information of all possible messages of a source, then we can interpret

$$I_i = -\log\ p_i. \tag{4.2}$$

as the **information** of the ith message. Thus, the probability of a message determines its information content. The less likely a message, the more *information* it has. Equation 4.2 is also intuitive : It increases as the probability of the message decreases, and it approaches zero as the probability of the message becomes equal to one. Both equations, 4.1 and 4.2, assume a source with no statistical dependence between two messages.

Assuming statistical independence between two messages is in most cases too simple. Consider for example written language. Redundancy in written language require some letters (or words) to appear after another with a probability much higher than the individual, overall probability of that particular letter (or word). The possibilities in English for a letter following the combination `lett` are restricted, even more so in the context of a sentence. However, the point of this section is to only *introduce* the concepts from information theory which allow us to characterize the techniques in the following sections. We need that classification to compare different approaches in a comprehensive way, not the least our own.

[1]This is the same J.W. Tukey who "invented" the Fast Fourier Transform (FFT) together with J.W. Cooley [CT65]. Both were colleagues of Shannon at Bell Labs.

It should be noted that the formulas given here are only valid for a discrete source. Continuous sources are not relevant for this work, because the relevant data are sampled pixel values, i.e. a set of discrete data.

4.1.2 Relative Entropy, Maximum Compression, Redundancy

Shannon defines the **relative entropy**, which we denote as H_{rel}, as the ratio of the entropy of a source to the maximum value it could have. This is the case when all messages have the same probability. With a source alphabet of n messages, the maximum entropy is

$$H_{\max} = -\sum_{i=1}^{n} p_i \log p_i = -\sum_{i=1}^{n} \frac{1}{n} \log \frac{1}{n} = \log n, \tag{4.3}$$

and H_{rel} becomes

$$H_{\text{rel}} = \frac{H}{H_{\max}} = \frac{-\sum_{i=1}^{n} p_i \log p_i}{\log n}, \tag{4.4}$$

where H is the entropy of the actual message or sequence of messages. Equations 4.3 and 4.4 again assume statistical independence between two messages. This assumption, although unrealistic for real applications, simplifies our considerations without losing generality.

The relative entropy, H_{rel}, is the **maximum compression** possible when we encode into the same alphabet. It immediately follows that one minus the relative entropy is the **redundancy** of a sequence of messages :

$$R = 1 - H_{\text{rel}} = 1 + \frac{\sum_{i=1}^{n} p_i \log p_i}{\log n} \tag{4.5}$$

In real life, most data in its raw form are encoded with a code corresponding to its maximum entropy. Some examples :

ASCII character set : The ASCII (= American Standard Code for Information Interchange) is the most common format for text files in computers and on the Internet. The alphabet of the ASCII code has 128 symbols. This gives a maximum entropy of

$$\lceil \log_2 128 \rceil = 7 \text{ [bits]}.$$

Accordingly, in an ASCII file, each alphabetic, numeric, or special character is represented with a 7-bit binary number.

EBCDIC character set : EBCDIC (= Extended Binary-Coded Decimal Interchange Code) is a binary code for alphabetic and numeric characters that IBM developed for its OS/390 operating system of the S/390 servers. The alphabet of the EBCDIC code has 256 symbols. This gives a maximum entropy of

$$\lceil \log_2 256 \rceil = 8 \text{ [bits]}.$$

Therefore, each alphabetic or numeric character is represented in an EBCDIC file with an 8-bit binary number.

Raw image pixel encoding depends on the video acquisition device, the screen color depth, or simply the resolution desired by the user. The most common resolution for grayscale or color components is 256. The encoding is :

$$\lceil \log_2 256 \rceil \;=\; 8 \text{ [bits]}.$$

Each grayscale pixel or color component value is encoded with a 8-bit binary number.

Codes like ASCII and EBCDIC codes are called *fixed length codes*, since they assign every character in a document the same codeword length, even though characters are most likely not to have the same frequency of occurrence within a document. The same is true for pixel encoding.

The absolute optimal average codeword length is equal to the entropy, and the entropy is dependent on the actual occurrence of the possible messages, i.e., the frequency of occurrence of a certain letter in a text or a pixel in an image. On the other hand, it is not possible to encode a source with a smaller average codeword length than the entropy without information loss ! If every message i with probability p_i is assigned some codeword with length l_i, then the average codeword length is

$$l = \sum_{i=1}^{n} p_i \, l_i. \tag{4.6}$$

Assuming that the alphabet was encoded with maximum entropy, $H_{\max}$, and applying equation 4.5, we get the redundancy

$$R \;=\; 1 - \frac{H}{l} \;=\; 1 + \frac{\sum_{i=1}^{n} p_i \log p_i}{\sum_{i=1}^{n} p_i l_i} \tag{4.7}$$

The redundancy is in literature often expressed in the units of the code used, usually bits. This is done by multiplying equation 4.7 with l :

$$\begin{aligned} R_{\text{bits}} \;&=\; R \cdot l \;=\; l - H \\ &=\; \sum_{i=1}^{n} p_i \, l_i - \left(- \sum_{i=1}^{n} p_i \log_2 p_i \right) \\ &=\; \sum_{i=1}^{n} p_i \, (l_i + \log_2 p_i), \end{aligned} \tag{4.8}$$

which is simply the difference between the actual average codeword length and the entropy.

We see that an optimal code with no redundant content therefore has an average codeword length equal to the entropy. Yet another interesting fact can be derived from equation 4.8 : The redundancy is equal to zero, if each single codeword length is

$$l_i = -\log_2 \, p_i = \log_2 \frac{1}{p_i}. \tag{4.9}$$

This is true if the probabilities are powers of the inverse of the base of the codeword. That is for the binary case, all probabilities p_i have to be $(1/2)^n$, where n can be any positive integer.

4.1.3 Source energy

The source energy is the energy of the symbols of the messages coming out of a source. These symbols are represented by a continuous or discrete signal such as pixel intensity (luminance) values. Let $x(t)$ be a stochastic, continuous signal coming out of a source, then

$$E = \int_0^T x^2(t)\, dt \tag{4.10}$$

is the signal energy between time 0 to time T. The integral in equation 4.10 becomes a sum in the case of a discrete source. We come back to signals and signal energy in section 4.3.

Instead of integrating over time, we might be interested of signal's energy (distribution) in space. In that case we integrate along the signal's dimensions in space. If, for instance, we are interested in the energy of a video image, we compute

$$E_{\text{image}} = \sum_{i=1}^{W} \sum_{j=1}^{H} x^2(i, j), \tag{4.11}$$

where W is the image width, H the image height, and $x(i, j)$ the pixel value at position (i, j).

4.1.4 Conclusion

Information theory gives us two measures to decide if data compression is possible and/or efficient : entropy and redundancy. Shannon suggested that the obvious way to do compression is to find a data representation which approaches the entropy of the data to be encoded, eliminating redundancy. The entropy is based on the actual statistical occurrence of the messages from an information source.

Reducing or eliminating **redundancy** is a lossless, completely reversible encoding step. Anything that reduces the data information rate below its entropy introduces loss of information and is therefore irreversible. Such an encoding step, which may be justified by a substantial gain in compression at the cost of reconstruction quality at an acceptable level, is called **entropy reduction**.

In addition, we can change the representation of the source data to change the energy distribution of the source data coefficients, a process which is called **energy compaction**. This does not reduce the actual amount of data, but it can make entropy reduction much more efficient. Note also that energy compaction may involve quantization thus introducing some data loss.

The following sections present algorithms for all three encoding methods.

4.2 Redundancy Reduction

Equation 4.5 defined the redundancy of a source message representation or code. This section discusses some of the most important algorithms of reducing or even eliminating redundancy of an data stream coming out of an information source. Any compression algorithm is closely related to its underlying source model, differing models making for different encoding algorithms. The closer such a model is to the truth, the better the compression algorithm performs.

Four years after Shannon created information theory, D.A. Huffman published his landmark paper about the construction of a code optimized with respect to redundancy [Huf52]. Huffman developed his technique as part of a class assignment at MIT. The class was held by R. Fano and was the first ever held in information theory [Say00]. Because of its importance and since it is still in use, for instance in the ZIP compression algorithm discussed in the next chapter, section 4.2.1 will give an overview over **Huffman's compression** algorithm.

Huffman's code becomes inefficient when probabilities of source messages differ a lot and, in particular, if one of the messages has a probability close to 1. In this case, grouping messages can greatly enhance the efficiency. A Huffman code taking advantage of this is conceivable, but a code for every possible combination of messages would have to be generated (complexity $O(n^2)$ with n being the number of messages.). An algorithm using the grouping of messages without having to generate a code for any possible grouping is called **arithmetic coding**. The roots of arithmetic coding go back to one of Huffman's fellow students in Fano's class, P. Elias. In section 4.2.2 we use the algorithm as it is described in [WNC87].

Both Huffman and arithmetic coding assume statistical independence of the messages to be encoded, an assumption which does not hold in many practical cases. An efficient solution to this problem is to encode messages in words instead of letters, an approach called **dictionary coding**. Two important algorithms encoding messages in words were published by Jacob Ziv and Abraham Lempel in 1977 [ZL77] and 1978 [ZL78].

Last, but not least, section 4.2.4 briefly presents **run length coding**, a widespread one-pass coding technique used for FAX machines [ITU80], as well as in image [ISO94] and video coding [ITU93, ITU96].

4.2.1 Huffman Coding

The Huffman encoding algorithm is based on the assumption that all messages of a source are statistically independent. Two events A and B are statistically independent, if their joint probability — i.e., the probability that they occur at different times — is equal to the product of their individual probabilities.

$$P(A,B) = P(A) \cdot P(B) \tag{4.12}$$

Huffman's algorithm uses two observations about optimum code [Say95] :

a. The higher the probability of a message, the shorter should be its code. This can be directly derived from equation (4.9).

b. The two symbols with the smallest probabilities should have the same code length.

Starting with these properties for an optimal code, Huffman developed the following procedure. (The decoding algorithm is not discussed here.)

Algorithm HUFFMAN(S, K)
Input: A set, S, of N messages
Output: Optimum binary code
1. Sort the probabilities of N messages of the source in descending order : $P(1) \geq P(2) \geq ... \geq P(N)$
2. **repeat**
3. Build a *coding tree* by combining the two lowest probabilities
4. Combine the next two lowest probabilities
5. **until** no probability is left
6. The upper member of each pair is assigned a zero, the lower member a one — or vice versa
7. Trace the path from each probability to the connection point, and keep track of the ones and zeros along each path
8. Write the one-zero sequence for each message from right to left

Figure 4.3 illustrates the algorithm. The minimum fixed length binary code for these seven symbols would be $\lceil \log_2 7 \rceil = \lceil 2.807 \rceil = 3$ bits. In this case Huffman's code is efficient. The code generated this way has some important properties making it interesting for use in codecs.

In fact, messages do not have to be encoded separately. They may be encoded in blocks of equal length. The probabilities of all messages in a block are then multiplied to yield the joint probability of the entire block. It can be shown that the more messages are encoded in one block, the more the average codeword length is approaching the entropy of the source.

Prefix code

Another important property of Huffman code is that, since no shorter codeword can be a prefix of a longer codeword, this sort of code is called *prefix code*. This is assured by two restrictions for the generated code :

a. *No two messages will consist of identical arrangements of coding digits.*

b. *The message codes will be constructed in such a way that no additional indication is necessary to specify where a message code begins and ends [...].*

Prefix codes are thus always uniquely decodable. This is particularly advantageous in the case of serial data transmission like in video conferencing and telephony. Other prefix codes are generated, e.g., by the arithmetic coding algorithm (section 4.2.2).

Huffman coding requires that the probabilities of *all* potential messages of a source have to be known in advance. *If* they are known in advance, then the Huffman Algorithm is likely to generate

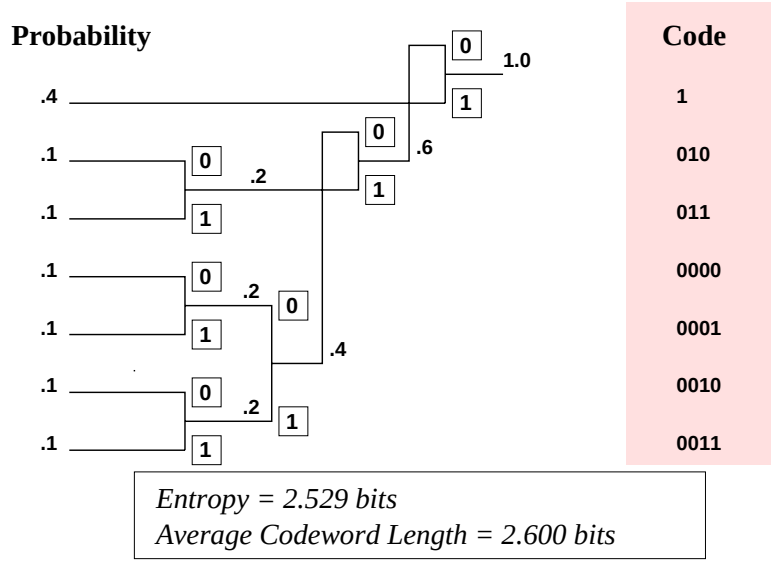


FIG. *4.3: Example to illustrate the Huffman Encoding Algorithm*

the optimal code for a certain base and block length. Huffman emphasizes in his paper that *[...] "optimum code" means "minimum-redundancy code."* [Huf52].

In certain cases, the Huffman algorithm fails to produce efficient code. Section 4.2.2 gives an example and discusses an alternative encoding procedure which is however based on the same source model(s) as Huffman coding.

Adaptive Huffman Coding

If the probabilities of the messages of a source are *not* known *a priori*, they have to be collected first. If, e.g., an image is to be encoded, the pixel data probabilities are computed first, and then in a second pass the actual encoding is done. If, on the other hand, there is no time for a two pass procedure, a certain modified one pass Huffman encoding algorithm is used. This sort of Huffman encoding is called adaptive, because statistics are updated with every newly encoded message. The branches of the coding tree are no longer assigned probabilities but weights, and when a new message arrives that has not been encoded yet, a certain branch of the old tree called NYT (for "Not Yet Transmitted") forks into two new branches : a new NYT, and one for the newly arrived message. Then the whole tree is updated and rearranged, if necessary.

This is a fairly complicated algorithm, whose implementation may require considerable computing power. Its sole advantage is its one pass character, making it attractive for real-time applications provided that enough computational power is available. Simulations show that the two pass Huffman algorithm and the adaptive Huffman algorithm yield about the same results. A more thorough introduction to both of these algorithms can be found in [Say95].

Discussion

Let us consider the Huffman algorithms from the image processing point of view. There are two statistical properties of images to be noted :

a. Any two images are unlikely to have the same entropy unless they have the same content. Therefore, the optimal codes for two images will probably not be identical.

b. Image pixel values are highly correlated within the same image.

These two facts weaken the reasons for a direct employment of the Huffman algorithms for image and video data compression, since the Huffman algorithms are based on the statistical independence of the input data. However, even if a Huffman code is far from being optimal for a particular image, it still provides *some* compression. This might be crucial for a low-bit-rate application like video telephony. A variation of the Huffman coding procedure is used for the JPEG compression standard (section 5.2) and the ZIP encoding algorithm (section 5.1).

4.2.2 Arithmetic Coding

The main drawback of Huffman coding is that it assigns at least one bit of code to a symbol of an alphabet. This can be much more than necessary especially if one symbol has a probability close to one. We can easily illustrate this by calculating the information of a message of probability 0.9 using formula 4.2, which is $\log_2 0.9 = 0.152$ bits. This is almost seven (!) times less than the supposedly optimum, minimum Huffman code. Imagine a fax machine encoding black and white pixels for an almost empty sheet. Huffman code would be very inefficient in this case assigning at least on bit to each pixel, which is why another compression algorithm (section 4.2.4) based on another source model is used for this application [Nel92].

Arithmetic coding is a compression method based on the same source model but avoiding an explicit code for a particular message. It rather generates a unique tag for a sequence of messages from an information source. Only at the final encoding step a binary (prefix) code for that tag will be generated. This is the major advantage of arithmetic coding over Huffman coding, because it automatically takes care of heavily differing probabilities of messages.

It is true that the Huffman algorithm allows grouping of messages (which is not to be confused with assuming statistical dependence between messages !). However, a Huffman encoder would have to generate a code for each possible combination (permutation) of a group of messages. An arithmetic coder is content with the actual occurred message sequence. The source model used determines the entropy of the source and the limits of for compression. As for Huffman coding,

arithmetic coding uses either a fixed model with a given set of probabilities, or an adaptive model. The latter would assign messages of an alphabet only the probability they have had in a sequence at a certain point in the sequence.

In arithmetic coding, a sequence of messages is represented by an interval of real numbers between 0 and 1. As one message after the next is being encoded, the interval becomes smaller and smaller according to the probability of the last message encoded. The probability model is, just as in Huffman encoding, known to both encoder and decoder. A function mapping a source alphabet to the interval $[0,1)$ is the *cdf*, the *cumulative distribution function (cdf)*.

For a random variable, $X(a_i) = i$, representing the occurrence of a message a_i out of a source alphabet $A = a_1, a_2, ..., a_m$, the *probability density function (pdf)* is

$$P(X = i) = P(a_i), \tag{4.13}$$

and the cdf is defined as

$$F_X(i) = \sum_{k=1}^{i} P(X = k). \tag{4.14}$$

The interval assigned to the message a_i is then $[F_X(i-1), F_X(i))$.

A good way to illustrate how arithmetic coding works is by an example. Suppose we want to encode the sequence bookEOF. Assuming statistical independence between the symbols[2] gives us the fixed model for the corresponding alphabet {b, k, o, EOF} shown in table 4.1 (We will see in a second why we need the EOF.), which is known to both encoder and decoder.

TAB. *4.1: Fixed model probabilities for alphabet {b, k, o, EOF}*

Message	Probability	Range
b	0.2	[0.0, 0.2)
k	0.2	[0.2, 0.4)
o	0.4	[0.4, 0.8)
EOF	0.2	[0.8, 1.0)

The column "Range" contains the *cdf* of the alphabet {b, k, o, EOF}. The information (see also equation 4.2) of the sequence bookEOF is

$$-\log(0.2 \cdot 0.4 \cdot 0.4 \cdot 0.2 \cdot 0.2) = -\log 0.00128 \tag{4.15}$$

which using the base 2 corresponds to

$$\lceil -\log_2 0.00128 \rceil \;=\; \lceil 9.6096 \rceil \;=\; 10 \text{ bits}. \tag{4.16}$$

[2] Even in this simple case this is actually not the case as in English not all possible letters can follow an initial b in a word.

Table 4.2 depicts the encoding process. We start with the first letter to be encoded, here b, and keep its range in the cdf of the source alphabet. This gives us the interval $[0.0,\ 0.2)$. We then scale up the range and take the range corresponding to the second symbol to be encoded, the o. This gives us the interval $[0.0+(0.2-0.0)\cdot 0.4,\ 0.0+(0.2-0.0)\cdot 0.8) = [0.08,\ 0.16)$. The third interval for the second o is then $[0.08+(0.16-0.08)\cdot 0.4,\ 0.08+(0.16-0.08)\cdot 0.8) = [0.112,\ 0.144)$. This continues until we reach the EOF symbol terminating the encoding process which is illustrated in figure 4.4.

TAB. *4.2: Arithmetic encoding process for the sequence bookEOF*

Message	Low value	High value
	0.0	1.0
b	0.0	0.2
o	0.08	0.16
o	0.112	0.144
k	0.1184	0.1248
EOF	0.12352	0.1248

Note that the size (upper bound - lower bound) of the last interval is the probability of the entire sequence. From figure 4.4, we get as the size of the last interval

$$0.1248 - 0.12352 = 0.00128,$$

which is identical to the argument of the logarithm in equation 4.16. Thus, *any* real number between 0.12352 and 0.1248 will encode the sequence bookEOF, given the alphabet (and its probabilities) in table 4.1. The "natural" binary code for the sequence bookEOF is the binary floating point number of any real number chosen between the upper and lower bound.

Since *any* real number between 0.12352 and 0.1248 will encode the sequence bookEOF, we need to introduce a symbol indicating that the sequence has an end. Otherwise, the decoder would not be able distinguish between book, bookk, bookkk, and so on. Another practical problem for the implementation of arithmetic coding may be precision, especially for encoding long messages. Special care has to be taken to avoid overflow and underflow.

The example above was very simple to illustrate the arithmetic encoding procedure. The fixed length code for this four letter alphabet would be 2 bits, and the fixed model entropy of the source for the example is 1.92 bits. So there is a priori not much room for compression. Arithmetic coding produced a result of $9.6096/5 = 1.92$ bits / symbol, which is exactly the entropy of the source. We could improve compression by using another source model assuming statistical dependence between the symbols, but that is not the point here. Arithmetic compression works very well with longer sequences of messages where the messages have very different probabilities.

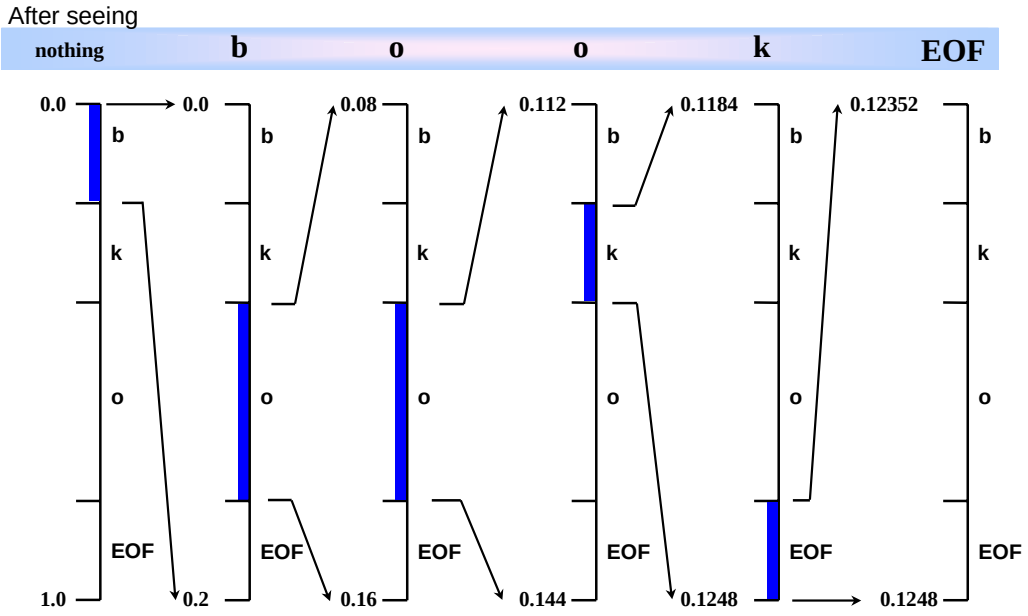


FIG. *4.4: Representation of the arithmetic coding process with the interval scaled up at each stage*

In a sense, arithmetic coding computes recursively the probability of a sequence of messages (assuming statistical independence). The logarithm of the inverse of that probability gives the (self-)information of the message sequence to be encoded.

4.2.3 Dictionary Coding Techniques

Huffman coding and arithmetic coding are both based on the statistical models for an information source. This can be convenient as it allows using simple source models. (*Remember* : the source entropy and therefore the maximum compression possible depend on the statistical model for the source !) Possibilities for improvement for Huffman coding and arithmetic coding include developing better models incorporating dependencies between messages and maybe building multi-dimensional dependency tables for letters and words of a source.

A much simpler approach was taken in 1977 and 1978 by J. Ziv and A. Lempel. With their two papers "A universal algorithm for sequential data compression" [ZL77] and "Compression of individual sequences via variable-rate coding" [ZL78] they started what are called dictionary based techniques. The corresponding algorithms are simply called LZ77 and LZ78 according to their year of publication. (The transposition of the first initials of the authors in the names of the algorithms happened accidentally.)

Dictionary techniques use a table of the most frequently occurring patterns of messages from a source (like *words* or even phrases in a language dictionary). Patterns are then represented by a reference into the dictionary. It is OK to use a **static dictionary**, i.e., to use the same dictionary for any source to be encoded. This may be efficient if a source produces always the same kind of messages. For a general purpose compression scheme, however, it is much more appropriate to use an **adaptive dictionary**, i.e., to *rebuild* the compression dictionary for every new sequence of messages to be encoded. LZ77 and LZ78 use both adaptive dictionaries. Both have been very popular until today and are used or the ancestors of the algorithms used in many general purpose compression applications such as Zip (section 5.1), compress, arc, and their implementations gzip, LHarc, Pkzip, the V.42bis specification for transmission by modem, the image file formats containing compression such as TIFF, PNG, GIF, etc.

Both algorithms, while based on the same basic idea of using dictionaries for compression, use two entirely different approaches.

LZ77 Algorithm

LZ77 uses a sliding window with a search buffer and a look-ahead buffer. The search buffer usually has a length of several kBytes, and the look-ahead buffer has a length of several tens up to some hundreds of Bytes. Data are encoded in triplets of numbers <O,L,C>, indicating the *offset*, the *length*, and the *codeword* for the symbol in the buffer that follows the match.

LZ77 initializes filling up the search buffer with zeros. To illustrate the encoding procedure, which is depicted in figure 4.5, let us assume that the data in the search buffer has already been encoded. The algorithm now searches a symbol or a string of symbols in the look-ahead buffer that has already been encoded. Starting with the first symbol in the look-ahead buffer, there is three possibilities of what can occur :

There is no match for that symbol in the entire search buffer. This corresponds to item c) in figure 4.5. Offset and the length coefficients are 0, and the code coefficient becomes the codeword for that symbol. This could be some variable length code such as a Huffman code (with pre-determined probabilities).

There is a match. This corresponds to items a) and b) in figure 4.5. Here the current symbol is encoded until the matching strings end.

The matched string extends inside the look-ahead buffer. This case is illustrated at item d) in figure 4.5. The string of symbols are simply encoded as if they were entirely in the search buffer. In fact, in [ZL77] there is no distinction between the two buffers. The look-ahead buffer is a part of the search buffer.

It can be shown that the performance of this simple algorithm approaches the efficiency of a compression scheme that has full knowledge of the statistics of the source [ZL77].

A big drawback of the original LZ77 algorithm is that if there was no match or a single match, a single symbol in the look-ahead buffer has to be encoded with a triplet. In real applications, LZ77 is modified to cover that case, for instance by using a flag to add an indication if the code to follow is for a single symbol or a string of symbols.

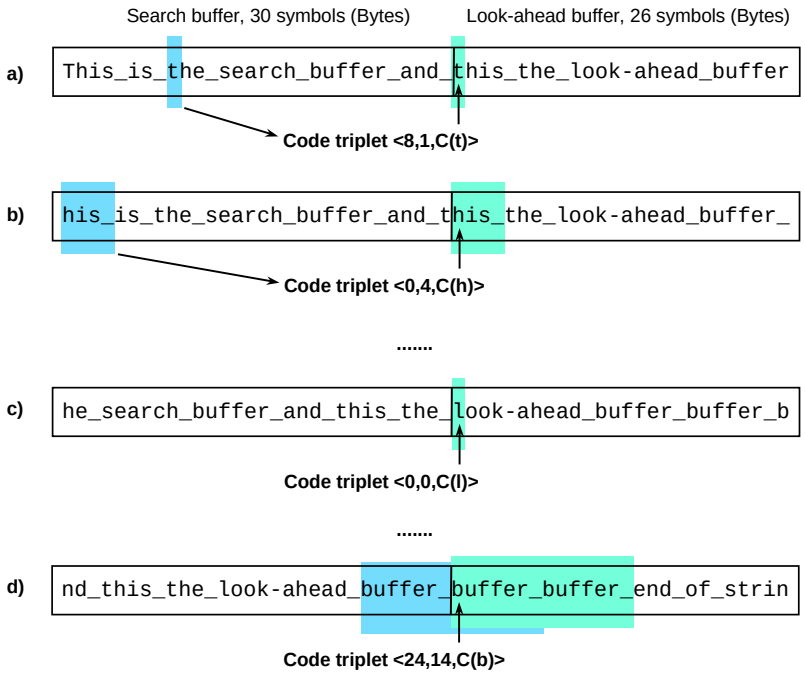


FIG. *4.5: The LZ77 encoding procedure. a) Match of one symbol, b) match of a string of 4 symbols, c) no match, and d) the matched string extents into the look-ahead buffer.*

LZ78 Algorithm

The previously discussed LZ77 algorithm assumes that matching symbol strings occur close together. Anything that lies outside the buffers is not considered to find matches. The worst case is periodic data with a period greater than the two buffers. Increasing the size of both buffers also increases the bits required to encode the code triplets. Plus it seriously affects computing efficiency.

LZ78 overcomes these problems by building an explicit dictionary. The encoder input is encoded in doubles instead of triplets. The doubles have the form $\langle \mathtt{i}, \mathtt{C} \rangle$, were `i` is the index of the current symbol in the dictionary (with a 0 indicating that the current symbol was not found in the dictionary), and `C` the code of the symbol to follow. The dictionary on the encoder side has to be build in a manner, which allows the decoder to build the same dictionary from the code transmitted. The code `C` may be some variable length code such as an adaptive Huffman code.

Let us illustrate the LZ78 algorithm with an example. If we want to encode the following sequence

`This␣is␣a␣test`

we get the initial dictionary in table 4.3 for the first five symbols.

TAB. *4.3: Initial dictionary of a LZ78 encoding procedure*

Index	Entry
1	T
2	h
3	i
4	s
5	␣

After the first five symbols the encoder will encoder will encounter symbols that it already has encoded. The build up of the dictionary during encoding is shown in table 4.4 : The sixth symbol is an "i", a symbol which has already been encoded. The encoder therefore outputs the double ⟨`3,C(s)`⟩, where 3 is the index of the previously encoded "i", and C(s) the code of the symbol to follow the matched symbol. The string "is" is added under index 6 to the dictionary. This continues until the string has entirely been encoded. The final version of the LZ78 dictionary for the string `This␣is␣a␣test` is shown in table 4.4.

TAB. *4.4: Dictionary development of a LZ78 encoding procedure*

	Dictionary	
Encoder Output	Index	Entry
⟨`0,C(T)`⟩	1	T
⟨`0,C(h)`⟩	2	h
⟨`0,C(i)`⟩	3	i
⟨`0,C(s)`⟩	4	s
⟨`0,C(␣)`⟩	5	␣
⟨`3,C(s)`⟩	6	is
⟨`5,C(a)`⟩	7	␣a
⟨`5,C(t)`⟩	8	␣t
⟨`0,C(e)`⟩	9	e
⟨`4,C(t)`⟩	10	st

4.2.4 Run-Length Coding

Run-Length coding encodes *runs* of messages (symbols) and the length of these runs instead of the symbols themselves. In its simple form it deals with a source producing only two different messages, e.g., white pixels and black pixels (as was the case with early fax machines). The underlying source model is a two-state Markov model (see, e.g., [Pap91]) with states S_w and S_b as shown in figure 4.6. S_w represents the case where a white pixel has just been encoded, and S_b the corresponding case for a black pixel.

The Markov model takes care of the fact that the probabilities of a particular color staying the same for two subsequent pixels, $P(w/w)$ and $P(b,b)$, is considerably higher than the probabilities for a color change, $P(w/b)$ and $P(b/w)$. This allows for a much "tighter" code than if it were assumed that the probability of a color change was equally high than the color staying the same for subsequent pixels. The reason is that the entropy in the latter case is much lower.

The actual encoding is a very simple process. Assuming a sequence of 150 white, 40 black, 130 white, 55 black pixels, the code would simply be 150, 40, 130, 55 with an indication for the color of the first string of pixels.

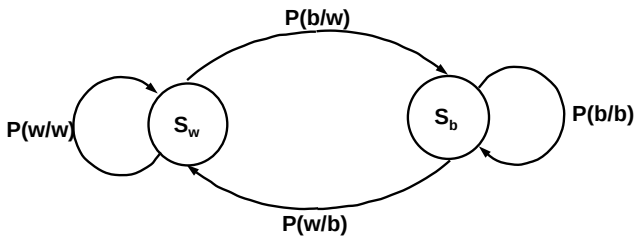


FIG. *4.6: Two-state Markov model for black and white pixel encoding (source : [Say95, Say00])*

A slightly more complex variation of this run length encoding algorithm is used in ITU-T recommendations H.261 and H.263 : Here the zero runs of various lengths are encoded. The coefficients which are non-zero are encoded by a Huffman like variable length code in table lookup manner. The probabilities have been empirically determined beforehand.

4.2.5 Conclusion

We discussed the some of the most important lossless data compression techniques. Two approaches were used : The first is using a model for the information source, where the quality of the model determines the compression performance. Huffman and arithmetic coding are two

examples of how a representation for the source data can be found which approaches the entropy of the source. The second approach is based on repeated patterns in the data coming out of the source. Although the two most important protagonists of this approach seem to be simple *ad hoc* procedures, they have found widespread applications because their performance approaches asymptotically that of compression algorithms based on statistical models for the information source.

4.3 Energy Compaction

Statistical relation of pixel values within an image almost automatically suggests the idea of using decorrelation techniques. Decorrelating pixel values redistributes the video signal energy such that a large amount of energy is concentrated in a small amount of transform coefficients. This eliminates the repeated and therefore redundant encoding of image pixel intensity or energy. The process of concentrating the signal energy of an image into a small number of transform coefficients is called *energy compaction*.

Energy compaction opens up the possibility of selecting salient image information and possibly omit some information. We can do that because of the limitations of human perception, or a tolerance of the human visual system towards certain types of degradation. This section discusses several signal representations commonly used to decorrelate image data. No extraction or segmentation of image content is done. Note that changing the data representation does not necessarily compress the amount of data, but rather rearranges image information for later compression steps such as quantization and other entropy reduction techniques.

Section 4.3.1 briefly reviews some chapters from signal theory in order to show the relation between the techniques relevant for this work such as Gram-Schmidt orthonormalization, Karhunen-Loève expansion, discrete cosine transform, etc. Sections 4.3.2 through 4.3.4 present these techniques in detail.

4.3.1 Discrete Signal Representations

In order to treat signals such as pixel (color) intensities of an image mathematically, we need to develop analytical representations of signals, numerical characterization of significant signal properties, and the characterization of the signal transforming properties of various processing systems. We will sketch here only the most important ideas of signal theory as far as we need them to introduce our own ideas. The reader is referred to appropriate literature such as [Fra81] for an in-depth treatment.

A signal is a quantity which carries information about a physical system. This quantity may be a voltage level, a force, anything that is measurable. Of particular interest here are, of course, intensity values of digitized images, graylevel and, in particular for this work, color intensities. Signals and time functions can be thought of interchangeably. Methods for dealing with time functions are usually also adaptable for use with functions of other variables. An image, for instance, could be described by a function of spatial coordinates or, as does OBC, as multi-dimensional vectors.

Let us review some basics about signals. A signal, x, can be considered as an element of a set, S, whose elements all have a common property, P. We write $S = \{x; P\}$, i.e., the set of all x such that P is true. Two common examples are :

Periodic signals. Let $S_R(T)$ be the set of signals which are periodic, with period equal to T, then

$$S_R(T) = \{x; x(t+T) = x(t), -\infty < t < \infty\} \tag{4.17}$$

Energy-limited signals. The signals in

$$S_E(K) = \left\{ x; \int_{-\infty}^{\infty} x^2(t)dt \ \leq \ K \right\} \tag{4.18}$$

are said to be energy limited to K, where K is a real, positive number. Note that the integral in 4.18 is physically interpreted as the energy content of a signal (compare equation 4.10). In image processing we deal with image pixel values, which are digitized and quantized image intensities and therefore energy limited.

Signals are describe by their properties and how they one signal is different from another with respect to a certain property. The *distance*, which is a positive real number, is used to describe the difference between two signals. A set of signals with a suitably defined distance is called a *signal space*. Distances with certain properties are called *metric*, and a set of signals together with an appropriate metric is called a *Metric space*. The metric that is pre-dominantly used for this work is the Euclidean metric. Adding an algebraic structure to a metric space creates a *linear space*. Elements in a linear space are also called *vectors*. The *norm* of a vector, which is a mapping from its linear space onto the real line, is the distance of a vector from its origin, in other words its *length*. The square of the norm of a signal or vector is its energy. The set of all energy-bounded signals is called L^2 space.

Changing the representation of a vector, e.g., by linear transformation, redistributes its energy. We discuss in the following various representations for signals using geometrical concepts for our signal representations and basis matrices to span the signal space of a certain representation.

Energy compaction may involve omitting some of the basis dimensions available in a signal space. This represents a loss of information. To measure the quality of a linear transform as an energy compaction step, two criteria are used :

a. the *efficiency* of the signal representation, i.e., how the signal energy is distributed over the basis dimensions, and

b. the *upper limit for the error* we make by choosing a certain representation and a maximum number of basis dimensions.

Since this work deals with discrete vector elements, i.e., digitized pixel values, the following sections consider the discrete versions of the concepts discussed. The reader is invited to consult the appropriate literature for the continuous-time versions.

Orthonormal Basis Spaces

It is convenient to use a linear space as signal space. Suppose M is an arbitrary n-dimensional linear space spanned by the basis $\{\vec{u}_i;\ i = 1, 2, ..., n\}$, then any vector $\vec{x} \in M$ can be expressed by

$$\vec{x} \ = \ \sum_{i=1}^{n} \alpha_i \vec{u}_i. \tag{4.19}$$

Taking inner products[3] on both sides yields

$$\langle \vec{x}, \vec{u}_j \rangle \;=\; \sum_{i=1}^{n} \langle \vec{u}_i, \vec{u}_j \rangle \, \alpha_i \qquad j = 1, 2, ..., n. \tag{4.20}$$

This is a set of n simultaneous, linear, scalar equations which can be solved for the n-tuple $\vec{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ providing the representation (in $\mathbb{C}^n$ or $\mathbb{R}^n$) for $\vec{x}$ relative to the basis $\{\vec{u}_i\}$. In most cases, we want the basis of our linear space to be an *orthonormal set*. A set $\{\vec{u};\ i = 1, 2, ..., n\}$ is said to be an orthonormal set if its vectors are *mutually orthogonal* and have *unit norm* :

$$\langle \vec{u}_i, \vec{u}_j \rangle \;=\; \delta_{ij}, \tag{4.21}$$

where δ_{ij} is the *Kronecker delta* defined by $\delta_{ij} = 1$ for $i = j$, and $\delta_{ij} = 0$ for $i \neq j$. With $\{\vec{u}_i;\ i = 1, 2, ..., n\}$ being orthonormal, we can write equation 4.19 :

$$\vec{x} \;=\; \sum_{i=1}^{n} \langle \vec{x}, \vec{u}_i \rangle \, \vec{u}_i \tag{4.22}$$

An orthonormal basis for M gives not only a one-to-one correspondence between vectors in M and their representation in $\mathbb{C}^n$ but an equality of inner products in both spaces. For

$$\vec{x} \;=\; \sum_{i=1}^{n} \alpha_i \vec{u}_i \qquad \text{and} \qquad \vec{y} \;=\; \sum_{i=1}^{n} \beta_i \vec{u}_i \tag{4.23}$$

we have

$$\begin{aligned} \langle \vec{x}, \vec{y} \rangle \;&=\; \langle \sum_{i=1}^{n} \alpha_i \vec{u}_i \,,\, \sum_{j=1}^{n} \beta_j \vec{u}_j \rangle \\ &=\; \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \beta_j^* \langle \vec{u}_i, \vec{u}_j \rangle \;=\; \sum_{i=1}^{n} \alpha_i \beta_i^* \;=\; \langle \vec{\alpha}, \vec{\beta} \rangle \end{aligned} \tag{4.24}$$

In order to compare the quality of various representations, we would like to estimate the error of our chosen representation. The *projection theorem* (see page 73) gives us the means to select an appropriate subspace of a linear space. Further below we present some frequently used orthonormal subspaces important for this work.

Let us now consider the problem of associating a *numerical representation* with an arbitrary signal of finite energy, i.e., a time function $\vec{x} \in L^2(T)$, with $L^2(T)$ denoting the L^2 space on the time interval T. This is equivalent to finding a suitable mapping from $L^2(T)$ into $\mathbb{R}^n$ where n is usually chosen on the basis of a compromise between accuracy and economy of representation. From the relative dimensionality of $L^2(T)$ and $\mathbb{R}^n$, it is clear that the mapping must exhibit a many-to-one nature and some sort of approximation is implied since each signal in $L^2(T)$ cannot have a distinct representation in $\mathbb{R}^n$. It is natural to think of this mapping in terms of an equivalence relation where $L^2(T)$ is partitioned in such a way that that the equivalence sets have a one-to-one correspondence in $\mathbb{R}^n$.

[3] $\langle \cdot, \cdot \rangle$ denoting an inner product.

Subspaces of $L^2(T)$

A common approach to this problem is to select a particular n-dimensional subspace of $L^2(T)$. Suppose $\{\vec{\varphi}_i; i = 1, 2, ..., n\}$ is a linearly independent set of functions in $L^2(T)$; i.e.,

$$\sum_{i=1}^{n} \alpha_i \varphi_i(t) \;=\; 0 \qquad \text{(almost everywhere) for} \quad t \in T \tag{4.25}$$

if, and only if, $\alpha_i = 0$ for each i. Let M_n denote the linear subspace spanned by these functions. If the signal to be represented, $\vec{x}$, happens to lie in M_n, then it has a unique expression as a linear combination of the $\{\vec{\varphi}_i\}$,

$$x(t) = \sum_{i=1}^{n} \alpha_i \varphi_i(t) \qquad \vec{x} \in M_n \qquad t \in T, \tag{4.26}$$

and the n-tuple $\vec{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ provides the desired representation in $\mathbb{C}^n$. Since $L^2(T)$ is an inner product space, the relationship between $\vec{x}$ and $\vec{\alpha}$ can be established according to equation 4.20 :

$$\langle \vec{x}, \vec{\varphi}_j \rangle \;=\; \sum_{i=1}^{n} \alpha_i \langle \vec{\varphi}_i, \vec{\varphi}_j \rangle \qquad j = 1, 2, ..., n \tag{4.27}$$

Introducing the reciprocal basis $\{\vec{\theta}_i; i = 1, 2, ..., n\}$ of $\{\vec{\varphi}_i\}$ in M_n, i.e.,

$$\langle \vec{\theta}_i, \vec{\varphi}_j \rangle \;=\; \delta_{ij} \qquad i, j = 1, 2, ..., n, \tag{4.28}$$

α_i in equation 4.27 can be expressed as

$$\alpha_i \;=\; \langle \vec{x}, \vec{\theta}_i \rangle; \qquad i = 1, 2, ..., n \tag{4.29}$$

Projection Theorem

In order to estimate the error of a certain representation, in how we represent a signal which is not contained in M_n. Since $L_2(T)$ is a metric space, we can associate a vector $\hat{\vec{x}}$, which is the closest vector to $\vec{x}$ in M_n. Each vector in M_n generates this way an equivalence set given by

$$S_{\hat{x}} \;=\; \{\vec{x} \in L^2(T); \quad ||\vec{x} - \hat{\vec{x}}|| \leq ||\vec{x} - \tilde{\vec{x}}|| \quad \text{for any} \quad \tilde{\vec{x}} \in M_n\}, \tag{4.30}$$

and every vector in $S_{\hat{x}}$ is to be represented by the n-tuple $\vec{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ for $\hat{\vec{x}}$. It turns out that this representation is also given by 4.29, even for an arbitrary $\vec{x} \in L^2(T)$. This is a result coming from the projection theorem [Fra81].

Theorem 1 (Projection Theorem) *For any $\vec{x} \in L^2(T)$, there is a unique $\hat{x}$ in M_n, given by*

$$\hat{\vec{x}} \;=\; \sum_{i=1}^{n} \langle \vec{x}, \vec{\theta}_i \rangle \, \varphi_i \tag{4.31}$$

such that $\vec{x}-\hat{\vec{x}}$ *is orthogonal to every vector in* M_n *and furthermore* $||\vec{x}-\hat{\vec{x}}|| < ||\vec{x}-\tilde{\vec{x}}||$, *where* $\tilde{\vec{x}}$ *is any other vector in* M_n.

Complete Orthonormal Sets

The projection theorem helps to find the best representation for an arbitrary signal relative to a given finite-dimensional subspace M_n. We now consider the question of how to select a suitable subspace. For a given subspace, it is clear that there is a large subset of $L^2(T)$ which is not adequately represented, e.g., points in $L^2(t)$ which are orthogonal to M_n. The problem of finding an optimal subspace is meaningful only relative to some (usually compact) suitably restricted subset of $L^2(T)$. It can be shown that the $L^2(T)$ space is *complete* and *separable*. These properties ensure that an approximation by means of orthogonal projection,

$$\vec{x} \sim \vec{x}_n \;=\; \sum_{i=1}^{n} \langle \vec{x}, \vec{\varphi}_i \rangle \, \vec{\varphi}_i \tag{4.32}$$

can be made arbitrarily close by choosing n sufficiently large for any $\vec{x} \in L^2(T)$. If no additional non-zero orthogonal vectors can be added to an orthonormal set, then that orthonormal set is said to be *complete*. A complete orthonormal set for $L^2(T)$ is equivalent to a basis for a finite-dimensional space.

Let us consider an example. The complex functions $\varphi_i(t) = \{e^{j\frac{\pi i}{T}t}; i = 0, \pm 1, \pm 2, ...\}$ are an orthogonal set in $[-T, +T]$. Using the continuous-time versions of equations 4.32 and 4.29 and normalizing we get

$$x(t) \sim x_n(t) \;=\; \frac{1}{\sqrt{2T}} \sum_{i=-n}^{n} \alpha_i e^{j\frac{\pi i t}{T}} \tag{4.33}$$

and

$$\alpha_i \;=\; \frac{1}{\sqrt{2T}} \int_{-T}^{T} x(t) e^{-j\frac{\pi i t}{T}} \tag{4.34}$$

which are the *Fourier series* expansion for duration-limited functions in $[-T, +T]$.

Basis Kernels

Without going into much detail, we can generalize the concept of a basis for a finite-dimensional space. Replacing the index variable i by some variable $s \in S$, where S will be a specified interval on the real line. The basis $\varphi_i(t)$ becomes $\varphi(t,s)$, and equation 4.26

$$x(t) \;=\; \int_S u(s)\varphi(t,s)ds \qquad \text{for } t \in T, \tag{4.35}$$

where $u(s)$ corresponds to α_i. The *basis kernel* $\varphi(t,s)$ becomes analogously to equation 4.29

$$u(s) \;=\; \int_T x(t)\theta(s,t)dt \qquad \text{for } s \in S, \tag{4.36}$$

where $\theta(s,t)$ is the *reciprocal basis kernel*. If a reciprocal basis exists, equations 4.26 and 4.36 are called a *transform pair*, and S is called *transform domain*.

There are several linear transform coding techniques such as the Hadamar, Haar, Fourier, Cosine, or Wavelet transforms, of which the Fourier is probably the best known. They all differ from each other by their bases.

Representation of Random Signal Processes

This section is a little detour to remind us that in physical systems such as digital video cameras, signal sources often produce any one of a large (possibly uncountable) set of time functions. Assigning a probability rule for the occurrence of each element of the set makes possible to analyze the source signals. A signal of such a source, denoted by $\mathbf{x}$, is called a *random process*, or *stochastic process*

Although the description of random processes is different from that of deterministic signals like those considered previously, the signal space concepts such as distance, norms, inner products, and orthogonality can equally be used to characterize random processes. For this work we make abundantly use of this fact when treating image pixel data, face position and size vectors, camera control parameters, etc. For an introduction to random variables, expectations, variances, moments, and filters, the reader is kindly referred to [Pap91] or [CB90].

A random process, $\mathbf{x}$, is defined as a set of *jointly random variables*, each indexed by a (time) parameter t,

$$\mathbf{x} = \{\mathbf{x}(t); \quad t \in T\} \tag{4.37}$$

If T is a countable set, then $\mathbf{x}$ is called a *discrete-time process*. If T is an interval on the real line, then $\mathbf{x}$ is called a *continuous-time process*. Using the expectations of $\mathbf{x}(t)$ as deterministic functions of time allows us to apply the algebraic and geometrical concepts developed above.

4.3.2 Gram-Schmidt Orthonormalization

The advantages of using orthonormal bases raise our interest in procedures for constructing such a basis. The *Gram-Schmidt orthonormalization procedure* is such a procedure, which is computationally advantageous because of its iterative nature. Given a set of n linearly independent vectors in M, $\{\vec{v}_i;\ i = 1,2,...,n\}$, the orthonormal basis set $\{\vec{u}_i\}$ is generated by normalizing the

$\{\vec{w}_i\}$ obtained by

$$\begin{aligned}
\vec{w}_1 &= \vec{v}_1 \\
\vec{w}_2 &= \vec{v}_2 - \langle \vec{v}_2, \vec{u}_1 \rangle \, \vec{u}_1 \\
\vec{w}_3 &= \vec{v}_3 - \langle \vec{v}_3, \vec{u}_2 \rangle \, \vec{u}_2 - \langle \vec{v}_3, \vec{u}_1 \rangle \, \vec{u}_1 \\
&\vdots \\
\vec{w}_i &= \vec{v}_i - \sum_{k=1}^{i-1} \langle \vec{v}_i, \vec{u}_k \rangle \, \vec{u}_k \\
&\vdots
\end{aligned} \tag{4.38}$$

where

$$\vec{u}_i = \frac{\vec{w}_i}{||\vec{w}_i||}; \qquad i = 1, 2, ..., n \tag{4.39}$$

It is important to notice that distinct orthonormal sets will be generated by re-orderings of the set $\{\vec{v}_i\}$ in the Gram-Schmidt procedure.

4.3.3 Karhunen-Loève Expansion

While the Gram-Schmidt procedure generates a complete orthonormal set, it is of course interesting to see if we can find the *optimal* n-dimensional basis in $L^2(T)$ for representing particular realizations of a random process. Optimization criterion is the $L^2(T)$ norm of the error averaged over the ensemble of realizations, denoted by E, which is to be minimized. In chapter 7 we will use the results of this section to design an efficient video encoding algorithm.

It can be shown that E can be minimized using the projection theorem, when just the autocorrelation function of the process to be presented is specified [Fra81]. In fact, equations 4.32 and 4.29 are the optimal representation of $\mathbf{x}(t)$, if $\varphi_i(t)$ are the eigenfunctions of the autocorrelation function $R_x(t)$ of the random process $\mathbf{x}(t)$. That is,

$$\mathbf{x}(t) \cong \sum_{i=1}^{n} \alpha_i \, \psi_i(t); \qquad -T \leq t \leq T \tag{4.40}$$

is the optimal basis w.r.t. E, and $\psi_i(t)$ are the eigenfunctions of the autocorrelation function $R_x(t)$.

Equation 4.40 is called the *Karhunen-Loève expansion*[4] of the random process $\mathbf{x}(t)$. If $\mathbf{x}(t)$ is a zero-mean process, then the coefficients α_i are also zero mean, and they are uncorrelated. If $\mathbf{x}(t)$ is *not* a zero-mean process, then E_φ is minimized by subtracting the mean μ_x from each $\mathbf{x}(t)$. This is equivalent to using the *autocovariance* function $A_x(t)$ instead of the autocorrelation function $R_x(t)$. This result is actually independent from the orthonormal basis $\{\varphi_i\}$ chosen.

[4]The Karhunen-Loève Expansion is sometimes called "Karhunen-Loève transform" [Lim90]. It *transforms* a signal into eigenspace.

Discrete time version

The discrete-time version of the Karhunen-Loève expansion is the expansion of the random vector, $\vec{\mathbf{x}}$, into the eigenvectors of its autocovariance matrix A_{xx}. This is the version we are using when we are processing images of a video stream. For a proof why the discrete time version of the Karhunen-Loève expansion is the optimal representation for a series of random vectors, see [Fuk90].

Analogously to equation 4.40, the N-dimensional random vector $\vec{\mathbf{x}}$ can be represented as

$$\vec{\mathbf{x}} \cong \sum_{i=1}^{M} \alpha_i \vec{\psi}_i \tag{4.41}$$

where $\{\vec{\psi}_i\}$ are the M eigenvectors of the autocorrelation matrix or, if $\vec{x}$ are not zero-mean, autocovariance matrix. If $\mathbf{X} = \{\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, ..., \vec{\mathbf{x}}_M\}$ is the set (or sequence) of all M random vectors $\vec{\mathbf{x}}$, then the first moment, expectation (or mean) vector, $\vec{\mu}_{\mathbf{x}}$, is

$$\vec{\mu}_{\mathbf{x}} \;=\; E\{\vec{\mathbf{x}}_i\} \;=\; \frac{1}{M} \sum_{i=1}^{M} \vec{\mathbf{x}}_i, \tag{4.42}$$

and the autocovariance matrix for a set of M random vectors, $\vec{\mathbf{x}}_i$, is

$$A_{xx} \;=\; \frac{1}{M} \mathbf{X}\, \mathbf{X}^T \;=\; \frac{1}{M} \sum_{i=1}^{M} (\vec{\mathbf{x}}_i - \vec{\mu}_{\mathbf{x}})\, (\vec{\mathbf{x}}_i - \vec{\mu}_{\mathbf{x}})^T \tag{4.43}$$

While A_{xx} is a NxN matrix, with N being the number of dimensions of $\vec{\mathbf{x}}$, it is clear that the order of A_{xx} will not be greater than M. What is more surprising (but nevertheless true) is that the *eigenvalues* of A_{xx} and

$$A'_{xx} \;=\; \mathbf{X}^T\, \mathbf{X} \;=\; \frac{1}{M} \sum_{i=1}^{M} (\vec{\mathbf{x}}_i - \vec{\mu}_{\mathbf{x}})^T\, (\vec{\mathbf{x}}_i - \vec{\mu}_{\mathbf{x}}) \tag{4.44}$$

are the same, which is particularly convenient for $N \gg M$, which in general is the case for images with about 100 kBytes pixel data and sequences of a few hundred video frames. The eigenvalue problem

$$A_{xx}\, \vec{\psi} \;=\; \lambda\, \vec{\psi} \tag{4.45}$$

or

$$\frac{1}{M}\, \mathbf{X}\, \mathbf{X}^T\, \vec{\psi} \;=\; \lambda\, \vec{\psi} \tag{4.46}$$

has the solutions the eigenvalues λ and the eigenvectors $\vec{\psi}$. Multiplying both sides of equation 4.46 from the left with $\mathbf{X}^T$ yields

$$\frac{1}{M}\, \mathbf{X}^T\, \mathbf{X}\, \mathbf{X}^T\, \vec{\psi} \;=\; \mathbf{X}^T\, \lambda\, \vec{\psi} \;=\; \lambda\, \mathbf{X}^T\, \vec{\psi} \tag{4.47}$$

Introducing $\vec{\phi} = \mathbf{X}^T \vec{\psi}$, equation 4.46 becomes

$$\frac{1}{M} \mathbf{X}^T \mathbf{X} \vec{\phi} = \lambda \vec{\phi}, \tag{4.48}$$

which is another eigenvalue problem with eigenvectors $\vec{\phi}$ and eigenvalues λ. Since $A'_{xx} = \frac{1}{M} \mathbf{X}^T \mathbf{X}$, we see that A_{xx} and A'_{xx} have the same eigenvalues. Multiplying both sides of equation 4.47 with $\mathbf{X}$, we get

$$\mathbf{X} A'_{xx} \vec{\phi} = \frac{1}{M} \mathbf{X} \mathbf{X}^T \mathbf{X} \vec{\phi} = \lambda \mathbf{X} \vec{\phi} = \lambda \vec{\psi} = \frac{1}{M} \mathbf{X} \mathbf{X}^T \vec{\phi} = A_{xx} \vec{\phi}, \tag{4.49}$$

which yields

$$\vec{\psi} = \mathbf{X} \vec{\phi}. \tag{4.50}$$

The space spanned by a set of eigenvectors is called *eigenspace*. We compute the eigenspace by solving equation 4.47 and applying equation 4.50. Mapping the vectors $\mathbf{X}$ into the eigenspace is done computing the M-tuple α_i in equation 4.41 applying equation 4.29. Since $\vec{\mathbf{x}}$ is not a zero-mean process, we need to subtract the mean from each $\vec{\mathbf{x}}_i$ before taking the inner product with each of the eigenvectors[5]. The transform coding pair into eigenspace and back using the KLE is given in table 4.5.

TAB. *4.5: KLE transform coding pair*

$$\begin{aligned} \vec{\mathbf{x}} &= \vec{\mu}_{\mathbf{x}} + \sum_{i=1}^{M} \alpha_i \vec{\psi}_i \\ \alpha_i &= \langle \vec{\mathbf{x}} - \vec{\mu}_{\mathbf{x}}, \vec{\psi}_i^* \rangle; \qquad i = 1, 2, ..., M \\ & \text{where } M \text{ are the number of basis dimensions} \end{aligned} \tag{4.51}$$

Principal Components Analysis (PCA) is the process of taking advantage of the optimal representation of data. "Components" means the eigenvalues of the autocovariance matrix of a set of data. Analyzing the principal, i.e., the most important eigenvalues, usually starts with putting them in descending order. The term "PCA" is often used to describe the mapping of a set of (random) vectors into eigenspace. More information on PCA in its original meaning plus an extensive list of literature can be found in [Gif90].

4.3.4 Discrete Cosine Transform

The disadvantage of the KLE is that for each set of vectors, $\mathbf{X}$, the corresponding eigenspace needs to be re-computed. This may even become a very inconvenient task since for 100+ images,

[5] See the comments under equation 4.40.

generation of the autocovariance matrix A'_{xx} needs a non-negligible amount of time. Moreover, for each new set of images, i.e., for each new sequence, the new basis computed from the autocovariance matrix needs to be re-transmitted. In chapter 7 we propose a new method addressing these problems, but for the moment, let us take a look how they have been solved up to now.

If re-computing the transform basis becomes unpractical for complexity, the attention is turned to fixed bases which are computed once and which approximate well the KLE. The *Discrete Cosine Transform (DCT)*, which was introduced by Ahmed and Rao in 1974, has been shown to approximate the KLE better than any other linear transform for a Markov data source [RY90]. Applied to images the Markov model exploits the statistical correlation of neighboring pixel values, which is obviously high in images having a sensible content. The basis of the DCT are cosine functions, and table 4.6 shows the discrete cosine transform pair for one dimensional vectors of dimension N.

TAB. *4.6: 1D Discrete Cosine Transform Pair*

$$X(k) = \begin{cases} \frac{2}{\sqrt{N}} \sum_{n=0}^{N-1} C(k)\, x(n) & \text{for} \quad 0 \le n \le N \\ 0 & \text{otherwise} \end{cases}$$

$$x(n) = \begin{cases} \frac{2}{\sqrt{N}} \sum_{k=0}^{N-1} C(k)\, X(k) & \text{for} \quad 0 \le k \le N \\ 0 & \text{otherwise} \end{cases} \qquad (4.52)$$

where n is the spatial coordinate in the pixel domain, and k is the coordinates in the transform domain, and

$$C(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } k = 0; \\ \cos\left(\frac{\pi(2n+1)k}{2N}\right) & \text{otherwise} \end{cases}$$

In order to further reduce computing requirements, images are sliced up in 8 x 8 pixel blocks before being transformed. The pixel blocks are transformed using a two-dimensional version of the DCT, shown in table 4.7. This results in less than 64 transform coefficients because of the symmetrie of the DCT transform matrix. The indices n_i of the spatial domain correspond to the time variable for the continuous cosine transform in the time domain. This is not to be mixed up, though, with the time dimension at inter-frame coding ! The indices k_i can be considered as frequency variables, since the transform domain of the Cosine Transform is similar to the frequency domain of the Fourier Transform. The normalization coefficients $C(k_1)$ and $C(k_2)$ are chosen to make the transform basis orthonormal.

Since it never changes during encoding, the DCT basis can be hard-coded into a video codec, making the DCT the most widely used transform in the field of image and video compression. Its

TAB. *4.7: 2D Discrete Cosine Transform Pair*

$$
\begin{aligned}
X(k_1,k_2) &= \begin{cases} \dfrac{2}{\sqrt{N_1N_2}} \displaystyle\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} C(k_1)C(k_2)x(n_1,n_2) \\ \qquad \text{for } 0 \le n_1 \le N_1-1, 0 \le n_2 \le N_2-1 \\ 0, \quad \text{otherwise} \end{cases} \quad \text{a)} \\
x(n_1,n_2) &= \begin{cases} \dfrac{2}{\sqrt{N_1N_2}} \displaystyle\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} C(k_1)C(k_2)X(k_1,k_2) \\ \qquad \text{for } 0 \le k_1 \le N_1-1, 0 \le k_2 \le N_2-1 \\ 0, \quad \text{otherwise} \end{cases} \quad \text{b)}
\end{aligned}
$$

where n_1 and n_2 are the spatial coordinates in the pixel domain,
k_1 and k_2 are the coordinates in the transform domain,

$$
C(k_1) = \begin{cases} \dfrac{1}{\sqrt{2}} & \text{for } k_1 = 0; \\ \cos\left(\dfrac{\pi(2n_1+1)k_1}{2N_1}\right) & \text{otherwise} \end{cases}
$$

$$
C(k_2) = \begin{cases} \dfrac{1}{\sqrt{2}} & \text{for } k_2 = 0; \\ \cos\left(\dfrac{\pi(2n_2+1)k_2}{2N_2}\right) & \text{otherwise} \end{cases}
$$

(4.53)

excellent energy compaction properties can be illustrated by transforming random blocks out of some images. Figure 4.7 shows some empirical data of randomly chosen and DCT transformed image blocks [Fur94]. Many spatial frequencies in a DCT transformed block have zero or near-zero values and do not need to be encoded.

4.3.5 Conclusion

We saw in this section that eigenfunctions, or eigenvectors, are the best possible representation for a random process, minimizing the $L^2(T)$ norm of the error averaged over the ensemble of realizations. Moreover, the error introduced by reducing the number of eigenvectors used for reconstruction is always known and therefore controllable. However, calculating the eigenvectors for a signal source is computationally costly, and only possible, if the source signal has zero mean. This is usually not the case ; particularly not for a sequence of images.

This restricts the use of eigenspace coding to cases where the mean of the process can well be estimated, or where the entire set of input vectors is known in advance. The latter is the case when working on pre-recorded sequences, and an eigenspace basis can be constructed using the KLE. Even though the KLE is known to yield the optimal representation, its prohibitive complexity

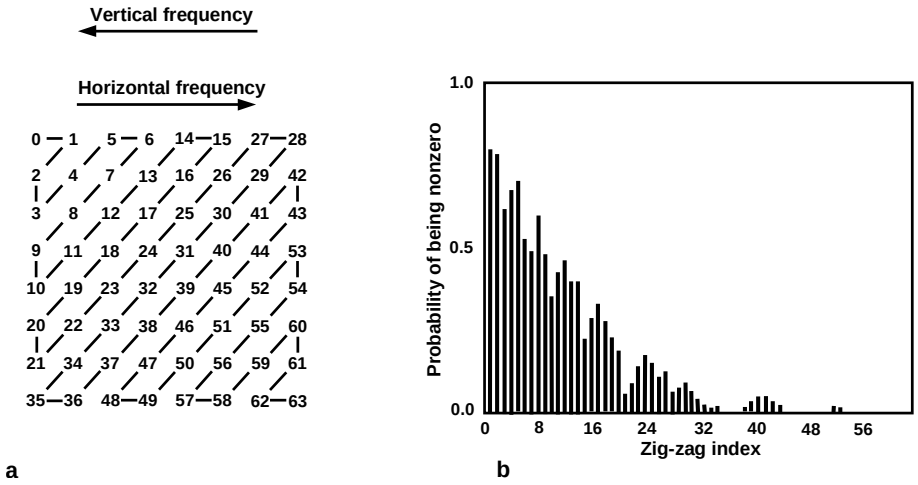


FIG. *4.7: Zig-zag Serialization of an 8 x 8 Pixel Block. [source [Fur94]]*

paved the way for the DCT, which currently is the mostly used energy compaction technique in image and video applications. In chapter 7 we will propose a new approach to handle all those problems.

4.4 Entropy Reduction

This section touches one of the most delicate questions in data compression : How can we efficiently reduce the entropy of our data to be compressed while keeping a maximum of information ? Section 4.2 about redundancy reduction discussed several of the most important techniques of *lossless* compression. Section 4.3 talked about efficient data representations, but changing the data representation does not, or not necessarily, mean a loss of information. In this section, however, we will present some methods of *irreversible* compression techniques effective involving loss of information.

The reason why imperfect reconstruction of data can be acceptable is closely related to the intended application. A certain degree of distortion of an image, for instance, may be acceptable to the human visual system. As an example may serve color sub-sampling of still-images.

The following two sub-sections discuss two different approaches to this problem of reducing the entropy of a data source, both are important for this thesis, and in fact, section 7.3 will consider this problem for the special case of video data. Another important topic when talking about entropy reduction is the choice of distortion metrics. We will touch this in the sections below, but, because of its importance for this work, we have dedicated section 7.4 to a thorough discussion, again with the focus on video data.

The existence of lossy compression techniques stems from the limitations of lossless compression with respect to efficiency. A model of an information source may identify only a limited amount of redundancy, which might not be enough for certain applications (see examples 1 and 2 in chapter 2). It is the application in question which actually determines the choice of the compression algorithm. In particular the application area targeted by this work is most likely not to be satisfied with lossless compression. Bandwidth-limited transmission channels and limited hardware storage space usually require compression rates of 1/100 to 1/1000, maybe higher. At the same time, the limitations of the human visual system and high statistical correlation of video image data allow for the application of quantization techniques as described below.

4.4.1 Scalar Quantization

If we want to represent an image or other data with a finite number of bits, then image intensities (luminance), transform coefficients, or model parameters must be quantized. Quantization involves assignment of the quantization (reconstruction) levels and decision boundaries. If we want to represent a continuous scalar quantity representing a pixel intensity, transform coefficient, or image model parameter, x, with a finite number of bits, only a finite number of reconstruction or quantization levels can be used. Let us assume that a total of L levels are used to represent x. The process of assigning a specific x to one of L levels is called amplitude quantization, or simply quantization. If each scalar is quantized independently, the procedure is called *scalar quantization*. If two or more scalars are quantized jointly, the procedure is called *vector quantization*. Vector quantization is discussed in subsection 4.4.2.

Let $\hat{x}$ denote the quantized value of x. We can express $\hat{x}$ as

$$\hat{x} = Q(x) = r_i, \qquad d_{i-1} < x \leq d_i, \tag{4.54}$$

where Q represents the quantization operation, r_i for $1 \leq i \leq L$ denotes L reconstruction levels, and d_i for $0 \leq i \leq L$ denotes $L+1$ decision boundaries or decision levels. If x falls between d_{-1} and d_i in equation 4.54, it is mapped to the reconstruction level r_i. Writing $\hat{x}$ in equation 4.54 as

$$\hat{x} = Q(x) = x + e_Q, \tag{4.55}$$

we can identify e_q as the quantization error given by

$$e_Q = \hat{x} - x. \tag{4.56}$$

e_Q is also called quantization noise. The quantity e_Q^2 can be regarded as a special case of a distortion measure, or distance metric, $d(x,\hat{x})$, which is a measure of distance or dissimilarity between x and $\hat{x}$. The reconstruction and decision levels are often determined by minimizing some error criterion based on $d(x,\hat{x})$ such as the average distortion, $\overline{d}$, given by

$$\overline{d} = E\{d(x,\hat{x})\} \; = \; \int_{-\infty}^{\infty} d(x,\hat{x}) \; p(x) dx. \tag{4.57}$$

The most straightforward method of quantization is uniform quantization, in which the reconstruction and decision levels are uniformly spaced. Specifically, for a uniform quantizer,

$$d_i - d_{i-1} = \Delta, \quad 1 \leq i \leq L \text{ and} \tag{4.58}$$

$$r_i = \frac{d_i + d_{i-1}}{2}, \qquad 1 \leq i \leq L, \tag{4.59}$$

where Δ is the step size equal to the spacing between two consecutive reconstruction levels or two consecutive decision levels. It can be shown [Say00] that for the simplest case, i.e., a uniform quantizer, a uniformly distributed source, and a fixed-length code of n bits per codeword, the *Signal-to-Noise Ratio (SNR)*, which is the ratio of the signal variance to the distortion variance becomes

$$\text{SNR(dB)} \; = \; 20 \log_{10}(2^n) \; = \; 6.02 \, n \, \text{[dB]}. \tag{4.60}$$

This means that for each additional bit we spend on the quantizer, we gain 6.02 dB SNR. We come back to the SNR in chapter 7 when we talk about reconstruction quality and distortion measures. There we will give the definition of the SNR and discuss its relation to other distortion measures as well as its limitations.

Differential Pulse Code Modulation

Differential Pulse Code Modulation (DPCM) is a quantization technique often used for speech and video compression. Figure 4.8 shows the basic concept. Note that the parts with the colored background, i.e., the predictor, lines are identical. The decoder is therefore part of the encoder,

given that the same predictor is used. This property of a DPCM system enables output control at encoding time, because the final signal output which enters the filter on the receiver side is exactly identical with the input signal of the predictor in the encoder.

A quantizer basically performs a many-to-one mapping (see equation 4.54) of the input data to a limited set of output values. Obviously this means some loss of information. In image coding, especially the low energy (high frequency) coefficients are usually lost causing energy reduction of the whole encoded image.

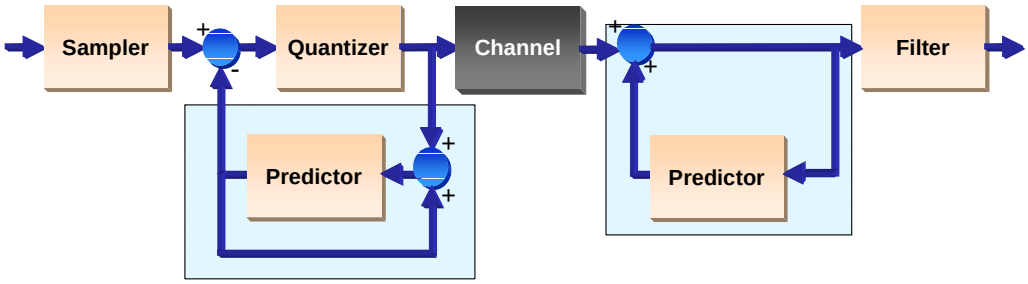


FIG. *4.8: Conceptual DPCM System*

A DPCM system as shown in figure 4.8 works as follows : The difference between the actual sampled signal (For image coding, "signal" can be replaced by "image.") and its predicted value is quantized and transmitted. Taking the difference removes a lot of redundant information in the data. In order to predict the next signal, the just quantized difference is added to the last predicted value. This sum, which is also the output on the receiver side, is the next input for the predictor. The predictor now estimates the next signal, based on its current input value and perhaps some former input values. Usually the predicted value is computed by just weighting the input value(s) with some weighting coefficient(s).

The word "differential" in the name DPCM refers to the fact that not the direct input signal but the difference of the current input signal and its predicted value is encoded. The advantage of this approach is the much smaller variance of this difference compared to the variance of the original input signal. This has two additional benefits :

a. The quantizer resolution can be made much smaller enabling much smaller bit-rates, and/or

b. the quantizer step-size may be reduced, which in turn reduces the signal-to-noise ratio (see section 7.4).

4.4.2 Vector Quantization

An alternate approach to coding source information is to divide scalars into blocks, view each block as a unit, and then jointly quantize the scalars in the unit. This is called *Vector Quantization (VQ)* or *block quantization*, and some of its approaches are equivalent or similar to *clustering* algorithms from the area of pattern recognition [TG74]. Figure 4.9 shows a two-dimensional example to illustrate the idea of vector quantization. Note that in vector quantization we are usually dealing with very high-dimensional data vectors.

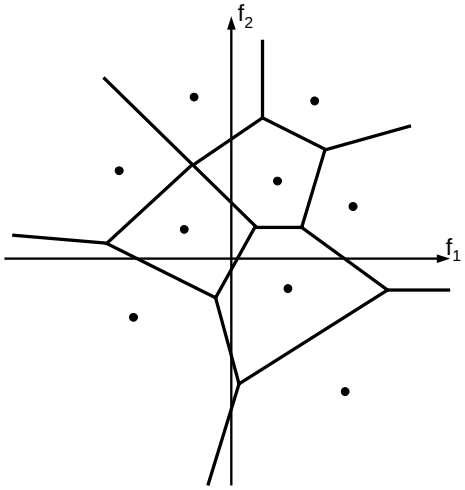


FIG. *4.9: Example of vector quantization. The number of scalars in the vector is 2, and the number of reconstruction levels is 9. (source : [Lim90])*

Let $\vec{x} = [x_1, x_2, ..., x_N]^T$ denote an N-dimensional vector that consists of N real-valued, continuous amplitude scalars x_i. Vector quantization maps $\vec{x}$ to another N-dimensional vector $\vec{r} = [r_1, r_2, ..., r_N]^T$. Unlike $\vec{x}$, whose elements have continuous amplitudes, the vector $\vec{r}$ is chosen from L possible reconstruction or quantization levels. Let $\hat{\vec{x}}$ denote $\vec{x}$ that has been quantized. Analogously to equation 4.54, we can express $\hat{\vec{x}}$ as

$$\hat{\vec{x}} = VQ(\vec{x}) = \vec{r}_i, \qquad \vec{x} \in C_i, \tag{4.61}$$

where VQ represents the vector quantization operation, $\vec{r}_i \, for 1 \leq i \leq L$ denotes the L reconstruction levels, and C_i is called the i^{th} cell.

In order to do vector quantization, the reconstruction levels, $\vec{r}_i$, and their corresponding cells, C_i, need to be determined. A list of reconstruction levels is called *reconstruction codebook* or simply *codebook*. If there are L reconstruction levels in the list, the list is said to be an L-level

codebook. The codebook is needed at the transmitter to quantize a source vector to one of L reconstruction levels and at the receiver side to determine the reconstruction level from the received codeword. The same codebook should, of course, be known to both transmitter and receiver through prior agreement.

Unlike the scalar quantization case, in vector quantization the reconstruction levels are vectors, and the cell boundaries are no longer points. The optimal determination of $\vec{r}_i$ and C_i depends on the error criterion used. One of the most important error criteria is the *Mean Squared Error (MSE)* (see section 7.4.1), which can be expressed as average distortion

$$\overline{d} = E[d(\vec{x},\hat{\vec{x}})], \quad \text{with} \quad d(\vec{x},\hat{\vec{x}}) = \langle \hat{\vec{x}}-\vec{x}, \hat{\vec{x}}-\vec{x} \rangle = (\hat{\vec{x}}-\vec{x})^T(\hat{\vec{x}}-\vec{x}).$$

One of the most important vector quantization algorithms is the so called K-Means algorithm. Because of its importance in clustering and vector quantization, the K-Means algorithm was implemented as a reference. The K-Means algorithm is thoroughly discussed in section 7.3.5.

4.5 Conclusion

This chapter discussed the theoretical basis of the compression techniques for this work. We saw why it is possible to compress data, and introduced classification criteria to divide compression algorithms into three groups : redundancy reduction, energy compaction, and entropy reduction. The details of each criterion were elaborated, and a selection of relevant techniques explained in detail. Particular care was taken for the development of the ideas behind energy compaction, since in this area OBC differs substantially from conventional approaches.

Chapitre 5

Image and Video Coding Techniques

This chapter covers some of the most common compression standards. They employ the techniques developed in the previous chapter. The focus is on three algorithms, a) a lossless Huffman-/LZ77-based algorithm called *ZIP*, b) the *JPEG* algorithm specifically designed for still image coding, and c) the general DCT/DPCM based video compression scheme as used in *ITU-T H.261*, *ITU-T H.263*, *ISO MPEG-1*, and *ISO MPEG-2*. All three are relevant for this thesis either because they are used to improve the results of our own approach, or they serve as benchmarks in terms of reconstruction quality, speed, and efficiency. An image or video codec is usually the first and the last part in a coding chain, serving directly as source coder, as shown in figure 5.1, which is a derivation of Shannon's schematic diagram of a general communication system depicted in figure 4.2.

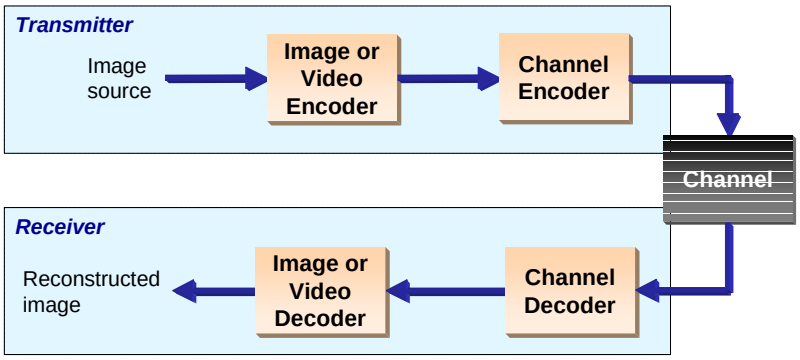


FIG. *5.1: Typical environment for image coding.*

In the context of this work we are less interested in the channel coding, transmission protocols, or file formats. We rather consider the source coding part, i.e., the image or video coder. An image or video codec itself consists of the three basic elements shown in figure 5.2. Even the latest algorithms such as JPEG2000 use these three elements, maybe not in the same order.

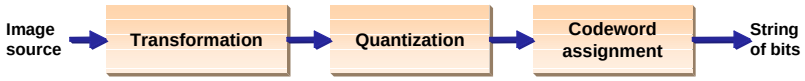


FIG. *5.2: Three major components in image coding.*

The first element of figure 5.2, representing an energy compaction step, is transforming images to the most suitable domain for quantization and codeword assignment. This element determines what specifically is coded. The second element in the image coder is performing entropy reduction by quantization. To represent an image with a finite number of bits, image intensities, transform coefficients, or model parameters must be quantized. The third element in the image coder is the assignment of codewords, the strings of bits that represent the quantization levels, reducing any remaining redundancy if possible. Each of these three elements attempts to exploit the redundancy present in the image source and the limitations of the display and the human visual system. We will find this 3-step scheme throughout this chapter with exception of section 5.1, since lossless compression is restricted to reducing redundancy.

Section 5.1 covers one of the most important lossless compression algorithms, used for a vast variety of applications in archiving, storage, transmission, and not in the least for image and video compression. We use it to compress the basis images or vectors in OBC. For the same purpose, we experimented with the lossy JPEG compression algorithm, which is discussed in section 5.2. Finally, because of its widespread use and its efficiency, the DCT/DPCM based compression scheme, employed in MPEG-1, MPEG-2, H.261, as well as in H.263, is introduced in section 5.3. This will eventually be the benchmark for our own approach.

5.1 ZIP

The ZIP algorithm is a lossless, all-purpose data compression technique based on the DEFLATE compression algorithm by Gailly and Adler [GA96]. DEFLATE uses a combination of Huffman [Huf52] (section 4.2.1) and LZ77 [ZL77] (section 4.2.3) encoding algorithms. GZip[Deu96] is a GNU[1] implementation of the ZIP algorithm. Another implementation of the DEFLATE algorithm is the ZLIB library, a public domain C library[2]. For this work we used both, the GZip

[1]GNU (`http://www.gnu.org`) is an organization creating freeware programs for Unix. Its name is a recursive acronym : *GNU's not Unix*

[2]source : `http://www.info-zip.org/pub/infozip/zlib/`

utility program (as a reference), and the ZLIB library for lossless compression of the basis at orthonormal basis coding. This sub-section sketches the basics of ZIP compression, referencing the techniques introduced in the previous chapter, but referring the reader to the referenced technical documentation, often freely available over the Internet.

5.1.1 Data structure

A file compressed by the GZip utility program consists of a series of so called *members*, which are compressed data sets. A compressed data set in turn consists of a series of blocks, corresponding to successive blocks of input data. The block sizes are arbitrary, except that non-compressible blocks are limited to 65,535 Bytes. Each block is compressed using a combination of the LZ77 algorithm [ZL77] and Huffman coding [Huf52]. The Huffman trees for each block are independent of those for previous or subsequent blocks ; the LZ77 algorithm may use a reference to a duplicated string occurring in a previous block, up to a input of 32 kBytes earlier.

A block consists of two parts : a pair of Huffman code trees that describe the representation of the compressed data part, and the compressed data part itself. The Huffman trees are compressed themselves using Huffman encoding. The compressed data consists of a series of elements of two types :

literal Bytes , which are Bytes of strings that have not been detected as duplicated within the previous 32K input Bytes, and

pointers to duplicated strings , where a pointer is represented as a pair < length, backward distance >.

The data format used for the DEFLATE algorithm limits distances to 32 kBytes and lengths to 258 Bytes, but does not limit the size of a block, except for incompressible blocks, which are limited as noted above. Each type of value in the compressed data , i.e., literals, distances, and lengths, is represented using a Huffman code, using one code tree for literals and lengths and a separate code tree for distances. The code trees for each block appear in a compact form just before the compressed data for that block.

5.1.2 Compression scheme

Literals or match lengths are compressed with one Huffman tree, and match distances are compressed with another tree. The trees are stored in compact form at the beginning of each block. The blocks can have any size, except that the compressed data for one block must fit in the available memory. A block is terminated when the DEFLATE algorithm determines that it would be useful to start another block with new trees. Duplicated strings are found using a hash table. All input strings of length 3 are inserted into the hash table. A hash index is computed for the next 3 Bytes. If the hash chain for an index is not empty, all strings in the chain are compared with the current input string, and the longest match is selected. The hash chains are searched starting with the most recent strings to favor small distances and thus take advantage of the Huffman encoding.

The hash chains are simply linked. There are no deletions from the hash chains, the algorithm simply discards matches that are too old.

To avoid worst-case situations, very long hash chains are arbitrarily truncated at a certain length, determined by a runtime option. The DEFLATE algorithm does not always find the longest possible match but generally finds a match which is *long enough*. Moreover, the DEFLATE algorithm defers the selection of matches with a lazy evaluation mechanism. After a match of a certain length *N* has been found, DEFLATE searches for a longer match at the next input Byte. If a longer match is found, the previous match is truncated to a length of one, thus producing a single literal byte, and the process of lazy evaluation restarts. Otherwise, the original match is kept, and the next match search is attempted *N* steps later.

The lazy match evaluation is subject to a runtime parameter. If the current match is long enough, DEFLATE reduces the search for a longer match, thus speeding up the whole process. If compression ratio is more important than speed, DEFLATE attempts a complete second search even if the first match is already long enough. The lazy match evaluation is not performed for the fastest compression modes. For fast modes, new strings are inserted in the hash table only when no match was found, or when the match is not too long. This degrades the compression ratio but saves time since there are both fewer insertions and fewer searches.

The corresponding decompression algorithm, INFLATE, basically deals with the problem of efficiently constructing hash-tables for table-lookup. Sets of hash-tables are preferred over one single table for memory reasons.

5.2 JPEG

This section gives a brief overview of ISO/IEC standard 10918, "Digital compression and coding of continuous-tone still images." The standard is commonly known as *JPEG* compression standard after its creator, the *Joint Photographic Experts Group*, where "joint" refers to a collaboration between ISO and ITU-T (former CCITT). We will use JPEG compression as an option to enable lossy basis compression for the OBC encoding method. Experiments show that using the ZIP lossless compression algorithm only yields a compression rate between 2/3 to 1/4, depending on the input data. OBC is conceptually designed for sequences of 100+ images. For short sequences, the size basis vectors prevent the algorithm to achieve satisfactory compression results. The ratio of the basis size to the input data size is simply too big. Therefore, JPEG is an interesting alternative to ZIP compression, if we want to increase compression for short sequences.

The JPEG compression standard was officially published in 1990 [ISO94, Wal91], but RFC versions were already used and implemented earlier, displaying the big demand for a still image compression standard. Proprietary still image compression standards were available at that time, but the success of the ITU-T recommendation for FAX encoding [ITU80] encouraged ISO and ITU-T to put together a tool-box of state-of-the-art techniques for lossy still image compression.

Although [Wal91] claims that there is a lossless compression mode, this is effectively not the case. To remedy this, ISO published a lossless version of JPEG, called JPEG-LS [ISO00a]. Other than that, JPEG is inherently *lossy*. It is designed to exploit known limitations of the human eye, notably the fact that small color changes are perceived less accurately than small changes in brightness. The degree of lossiness can be varied by adjusting compression parameters. This means that a user can trade off file size against output image quality. Another important aspect of JPEG is that decoders can trade off decoding speed against image quality, by using fast but inaccurate approximations to the required calculations. Encoders can also trade accuracy for speed, but there's usually less reason to make such a sacrifice when, e.g., writing to a file.

As stated above, JPEG is only for still images. Nonetheless, before the introduction of a generic video coding scheme such as H.261 or MPEG-1, some people used something they called *Motion JPEG* or *M-JPEG* for video compression. *There is no such standard.* All they did was apply JPEG to individual frames of a video sequence, and call the result M-JPEG.

5.2.1 Compression scheme

For grayscale images, an image is compressed in blocks of 8 x 8 pixels. Color image compression can be regarded as compression of multiple grayscale images, which are either compressed entirely one at a time or alternately interleaving 8 x 8 sample blocks from each in turn. Figure 5.3 illustrates this compression scheme on block level. The figure shows the special case of single-component (grayscale) image compression. The underlying compression scheme for JPEG is the following 3-step compression scheme :

Energy Compaction by (Fast)DCT (section 4.3.4). The first compression step is to transform the 8 x 8 pixel blocks using equation 4.53 a), with $N_1 = N_2 = 8$ The inverse transform as used in figure 5.3 b) uses equation 4.53 b). A fast implementation using table look-up may be used in order to avoid extensive multiplication.

Entropy Reduction by scalar quantization (section 4.4.1). After dividing the image into 8 x 8 pixel blocks and transforming them, the 8 x 8 transform coefficients are quantized. This is, of course, the sole source of loss in the JPEG encoding procedure. For efficiency reasons the first transform coefficient (= the average of all pixels in the block) is quantized separately.

Redundancy Reduction by Huffman (section 4.2.1) or Arithmetic coding (section 4.2.2). The final compression step is the reduction of any redundancy left in the output of the quantizer. The 8 x 8 blocks of transform coefficients are scanned in a zig-zag sequence as shown in figure 4.7. The resulting bit-stream is losslessly compressed using either a Huffman encoder (section 4.2.1) or an Arithmetic encoder (section 4.2.2). In either case, no coding tables are given by the standard, and so the data must be scanned twice : first to gather the statistics of the data and build the tables, and second to actually compress the data.

JPEG compression is controlled via a quality parameter, which can take a value between 1 and 100. This quality parameter controls a trade-off between compression and quality. Note that

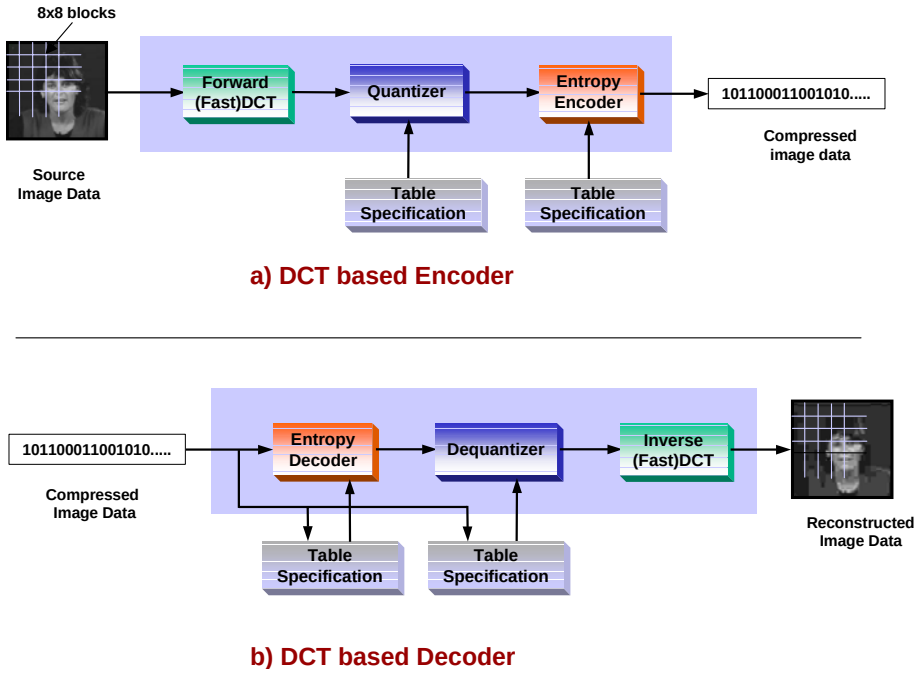


FIG. *5.3: JPEG encoding scheme : a) Encoding, and b) Decoding*

even a quality parameter of 100 implies lossy compression as has been explained in section 5.2. Figure 5.4 illustrates the trade-off between compression and quality for JPEG compression of two example images. ZIP compression for both images gave a compression of 17% for the image from the TALKINGHEAD sequence, and 24% for the image SERAPHINA.

5.2.2 JPEG2000

JPEG2000 is more than an overhaul of JPEG. It will be an entirely new image compression standard, using new algorithms and technologies developed during the last decade. Many of them address the short-comings of the DCT/DPCM scheme, trying to improve compression performance especially for high compression ratios. This is somewhat similar to what this thesis does with respect to conventional video compression techniques. A source of extensive information about the latest developments in JPEG technology can be found at `http://www.jpeg.org/JPEG2000.htm`.

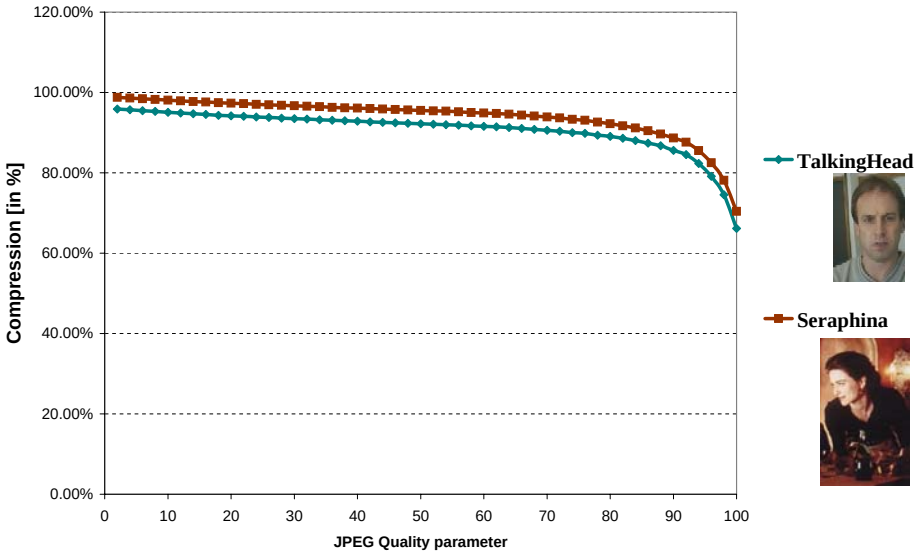


FIG. *5.4: Compression versus quality parameter for JPEG compression*

Compression scheme

The JPEG2000 encoding scheme will still use entropy reduction, energy compaction, and redundancy reduction, but use different algorithms to perform these tasks. Giving more than an overview over the JPEG2000 would go beyond the scope of this thesis, so we refer the reader to `http://www.jpeg.org/JPEG2000.htm` for more detailed information. JPEG2000 uses the following compression scheme :

Energy Compaction (optional) Components of the image are passed through an level shifting operation and a decorrelation transform suitable for RGB images. The latter is done either in a simple reversible version, or in a more accurate irreversible transform for lossy compression.

Energy Compaction by Wavelet transform using Mallat decomposition.

Entropy Reduction For lossy encoding a Daubechies 9/7 wavelet transform is used, or a simple symmetric 5/3 filter for lossless compression.

Redundancy Reduction The output of the transform stage is then organized into blocks using the *EBCOT* techniques, creating a bit-stream that consists of a number of independent layers. These may be given different priorities during transmission over a network to achieve differing image build-up features, or truncated during lossy compression. The output coefficients are quantized, where this process can be controlled to provide a given compression rate.

Redundancy Reduction An arithmetic coder entropy codes the final bit-stream.

Decoding is simply a reversal of these stages. Comparing to JPEG, this introduces an increase in complexity of about an order of magnitude.

Additional Features

JPEG2000 goes beyond employing new techniques for compression to replace the DCT/DPCM scheme. The standard will include a significant number of additional features, such as

- ⇒ imaging transmission from lossy to lossless,
- ⇒ the ability to define Regions of Interest which can be coded at higher resolution, or even losslessly to preserve certain image features (this can be important in medical imaging),
- ⇒ *"resync"* markers to be effective (with a recovery mechanism) on communication channels with high error rates (e.g., in applications such as the new mobile telephony channels),
- ⇒ *"fair"* quality image reproduction at rates down to 0.1 bits / pixel and below,
- ⇒ the ability to customize encoders and decoders to suit low memory or fixed size applications,
- ⇒ random code-stream access and processing,
- ⇒ high quality color image processing, with wider bit depths and larger image sizes,
- ⇒ use of alpha channels and other features to meet graphic arts and internet needs, and
- ⇒ features to provide for image security and content meta-data inclusion.

JPEG2000 vs. JPEG

The advantage of JPEG2000 over JPEG with respect to compression performance is mainly in the additional features and at high compression ratios. The author of [Cla00] claims that for low and medium compression ratios, the performance of JPEG2000 is *not* superior to (the old) JPEG. At least, the performances are not distinguishable for the human eye.

5.3 MPEG-1, MPEG-2, H.261, H.263

MPEG-x [ISO93, ISO96a, Gal91] and H.26x [ITU93, ITU96, Lio91] compression algorithms are application driven. That is, the intended application(s) determine(s) their features and the requirements they have to full fill. Other than that, they are all based on the same 3-step DPCM/DCT/RLE compression scheme

a. Entropy reduction by DPCM (section 4.4.1),
b. Energy compaction by (Fast)DCT (section 4.3.4), and
c. Redundancy reduction by run-length encoding (section 4.2.4).

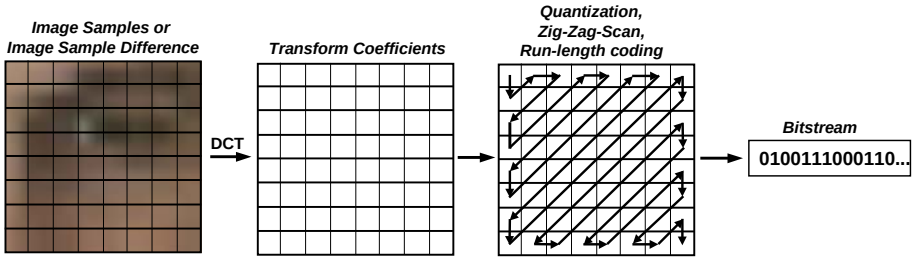


FIG. *5.5: MPEG block encoding scheme*

As with JPEG compression, images are encoded in a hierarchical block structure with blocks of 8 x 8 pixels as basic element. Figure 5.5 illustrates the DPCM/DCT/RLE compression scheme on block level.

This compression scheme is powerful and flexible making high compression rates possible. It resembles the JPEG compression scheme. The difference, however, lies in the exploitation of temporal redundancies. Lacking a generic video compression standard, JPEG was used in the pre-MPEG-x and pre-H.26x era for video compression. The fundamental flaw was just that the JPEG algorithm's incapacity to use prediction to remove temporal redundancies. As a result, video compression using JPEG exhibits a limited compression ratio. The reason for the existence of several similar compression methods such as MPEG-x and H.26x are their targeted application areas. These translate directly into lists of requirements the algorithms must fulfill. A look at the requirements make clearer the differences in the algorithm design.

5.3.1 Requirements for MPEG-x coding

MPEG-x is designed for encoding of full-motion video sequences with a good trade-off between image quality, and only to a lesser degree video communication. Targeted transmission rates are $\leq$ 1.5 Mbits/sec for MPEG-1 and $\leq$ 10 Mbits / s for MPEG-2. An MPEG-x codec can therefore be regarded as a digital VCR. As such it requires random access to single images, fast forward/reverse searches, and reverse playback features. MPEG-x uses bidirectional prediction to meet these requirements.

Robustness to errors in case of data loss or errors on the communication channel and an acceptable coding/decoding delay are also desirable. Additionally, MPEG-x should provide multimedia functionality such as editability for multimedia editing ; in other words, the encoded sequence should provide certain access points. On the economics side, the possibility of cost tradeoffs for hardware should allowing for large scale production.

5.3.2 Requirements for H.26x coding

The ITU-T H.261 recommendation for video compression has the full title "Video codec for Audiovisual services at p x 64 kbits/sec". Targeted application areas are video telephony and video conferencing rather than full-motion video encoding. Low, scalable bit-rates at p x 64 kbits/sec reveal ISDN lines to be the intended communication channel. As a rule of thumb, for $p = 1$ and/or 2, only a face-to-face scenario is to be transmitted with acceptable quality, and only for $p \geq 6$, more complex scenes are supposed to be handled.

Using ISDN lines as communication channel requires an acceptable coding/decoding delay and encoding in real-time. This requirement is softened by the fact that the communication scenario does not change within a communication session. As with MPEG-x, the consideration of a hardware implementation should enable a large scale production for codecs.

H.261 was the first non-proprietary video compression standard available. It can therefore be considered a first sketch of what a generic video compression algorithm can do. H.263 is an adaptation of H.261 to a changing technical environment. Some new features were adopted to exploit further improved hardware capabilities and to improve reconstruction quality at very low bit-rates.

5.3.3 Encoding procedure

In order to exploit temporal redundancy, both MPEG-x and H.26x use predictors. Since it is intended for reverse playback and search, MPEG-x uses bidirectional prediction. Moreover, in order to increase efficiency, it even employs interpolation. Figure 5.6 illustrates how this is done. Three types of frames have been defined : INTRA-frames (called I in figure 5.6), predicted frames (P), and interpolated frames (called B for bidirectional prediction). The number of *P* and *B* frames between two *I* frames can be defined by the user, but it affects directly the compression ratio (higher for more P and B frames) and the reconstruction quality (lower for more P and B frames).

H.26x requires no bidirectional prediction. The encoded video stream is supposed to be watched and therefore decoded in forward playback mode. Prediction for H.261 is therefore always forward prediction. As in any DPCM system, the decoder is part of the encoder in form of a predictor. Figure 5.7 shows the basic codec used by both MPEG-x and H.26x.

DPCM as a quantization method always introduces an error. For MPEG-x and its requirements, this error has to be tightly controlled in order to guarantee a certain quality of service (QOS). H.26x systems are more flexible since QOS requirements are not as restrictive as for MPEG-x. H.26x's prediction mode is called INTER mode, and a simple linear predictor is used. H.26x does not *force* an INTRA every some images as does MPEG-x, but *suggests* an INTRA frame encoding every 132 (!) frames. Of course, this makes for very high compression rates (but low image quality) as is necessary for low-bandwidth applications. Figure 5.7 shows the switches between INTER and INTRA mode.

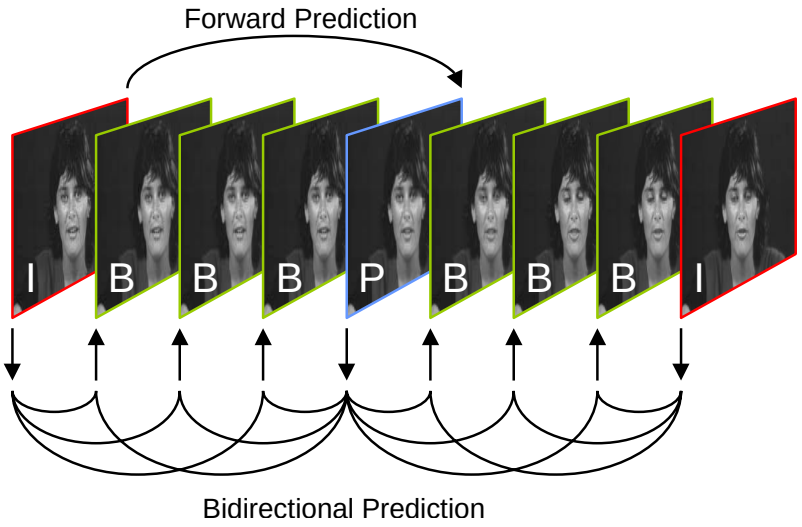


FIG. *5.6: MPEG bidirectional prediction scheme*

In INTRA mode, the entire input image is transformed by a two-dimensional (Fast)DCT in 8 x 8 pixel blocks. According to a user defined reconstruction quality level (this might also be imposed by the transmission channel bandwidth, i.e., the bit-rate at the output of the codec), a quantizer is quantizing the transform coefficients. The quantizer output is then further compressed by a lossless run-length encoding scheme in table look-up manner. The INTRA mode works the same way for H.26x and MPEG-x codecs.

In H.26x's INTER mode, the difference between the last reconstructed image (remember : the decoder is part of the encoder) and the new input image is transformed and quantized. For MPEG-x, the difference between the predicted and/or interpolated image, and the new input image is encoded. Interpolation avoids the reconstruction of every encoded image.

5.4 Conclusion

This chapter discussed three important compression methods, widely used for a variety of applications. The ZIP algorithm, based on a combination of Huffman and LZ77 encoding, is lossless. It preserves all the input data and has therefore the entropy of the source as a limit for its compression efficiency. Depending on how many passed through the data are admitted by the user, ZIP approaches more or less close this limit.

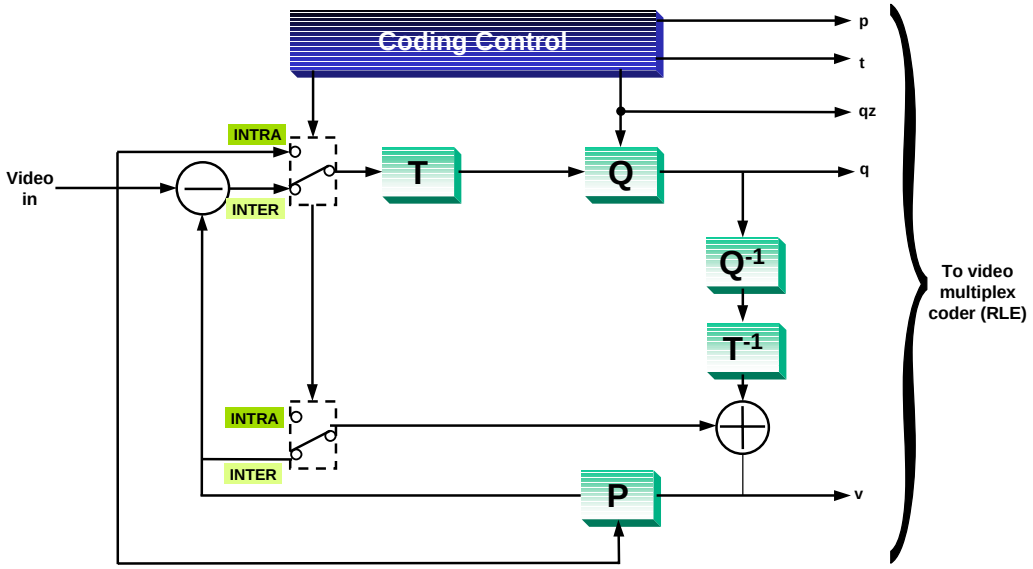


FIG. *5.7: Block diagram of DCT/DPCM based encoding.* ***T*** *: Transform stage,* $\boldsymbol{T}^{-1}$ *: Inverse Transform stage,* ***Q*** *: Quantization stage,* ***P*** *: motion prediction. RLE redundancy reduction stage not depicted.*

The irreversible, lossy compression technique called JPEG has been designed for the compression of visual data, exploiting the limitations of the human visual system, which accepts a coarser resolution of the chrominance (color) more easily than a coarse luminance resolution. It is based on the on the statistical relation between adjacent pixel values, and can yield very good compression introducing image degradations almost imperceptible for the human eye.

We also outlined the features of the new compression standard (to come), JPEG2000, incorporating new technologies and responding to the demands of a changing technological environment. Two things are remarkable with JPEG2000 : a) It is still based on the basic building blocks : energy compaction - entropy reduction - redundancy reduction. b) It outperforms "classical" JPEG only a very high compression rates at the price of a higher complexity on the encoder *and* on the decoder side. That is, "classical" JPEG is still the compression method to beat at still image compression when looking for a trade-off between compression ratio and reconstruction quality at medium and low compression rates. The major gain of JPEG2000 will therefore rather be the added feature listed in section 5.2.2.

Finally, we discussed the baseline compression method of MPEG-1, MPEG-2, H.261, H.263, and supposedly (considering the artifacts) RealVideo and Cinepak. What is true for JPEG and its successor, JPEG2000, is equally true for MPEG : New developments such as MPEG-4 and MPEG-

7 add rather new features around an "old" technique than doing something really new. The effect will be the same as for JPEG : Added functionality at the price of higher complexity. Remains the bottom line that DPCM/DCT based compression is still the method to beat. This is what we try to show that OBC, the new video compression scheme we propose, is capable in the following chapters.

Chapitre 6

Normalization by Tracking

One of the problems addressed in this dissertation is the contribution that stabilizing the position and orientation of a face can make to video compression for communication. Tracking reduces the required bandwidth while providing the speaker with the freedom to move about while communicating. To be useful for video compression, a tracker must be precise, robust, and fast so that orthonormal basis coding can be efficient. These requirements are met by feature-based tracking techniques, since no detailed object information such as shape, texture, or 3D form is needed.

In communication by video telephone or video electronic mail, the desired images are generally restricted to a view of the head and shoulders of a speaker. Relevant variations are movements of the mouth, eyes, and head. Precise reconstruction of the background is unimportant or may even be undesirable. Such image sequences have properties which make possible high compression ratios. Movements of the face and eyes tend to be repetitive making it possible for a compression algorithm to exploit the limited range of movements and their repetitive nature.

Tracking by color matching is a widely applied technique today, and it is implemented in software and hardware. For instance, the EV-ID 30 and EV-ID 31 cameras from Sony Co. use such an algorithm, hard-coded into a DSP, for their built-in automatic tracking system. This tracker can be controlled by a remote control or a serial line (RS 232) connection using a set of binary commands.

The techniques described in this chapter are based on earlier work in which a histogram of normalized skin color was initialized by blink detection and then used to determine the possibility that a pixel represents skin [CBC97]. While this system provided robust tracking of a moving face under changing illumination, the color skin detection technique relied on detecting connected components of thresholded color regions. Grouping thresholded pixels led to an unacceptable amount of jitter in the tracked images. In this chapter we describe a technique which replaces thresholding and connected components with the first and second moments of the 2D probability distribution of skin colored pixels. This approach rejects outlying pixels using a technique commonly used for robust statistics. In section 6.3.7 we compare the performance of both algorithms.

Section 6.1 introduces briefly the concepts of skin color detection by application of color histograms. Using that technique, we apply in section 6.2 a robust tracking algorithm based on

Bayes' rule to compute the probability that a pixel has skin color. In the subsequent steps, position and size of the tracked skin colored object are determined by the first and second moments of the skin colored region using a robust algorithm. This algorithm is a general technique which can also be applied to other tracking algorithms.

6.1 Color detection

Color is a perceptual phenomenon related to human response to the visible spectrum of electromagnetic waves between 400 (blue) and 700 (red) nanometers. The sensation of a color arises from the sensitivities of three types of neurochemical sensors in the retina to the visible spectrum. Each sensor responds to a range of wavelengths. A digital camera tries to model those sensors of the human visual system. The wavelengths form a natural basis or coordinate system from which the color measurement process can be described. Weighted combinations of stimuli at three principal wavelengths are sufficient to define almost all the colors we perceive.

Color, as we perceive it, is a function of several parameters which depend on a surface. Let $S(\lambda)$ be the reflectance function modeling the reflectance on the surface of an object. Let $E(\lambda)$ be the incident light *spectrum*, and $h_R(\lambda)$, $h_G(\lambda)$, and $h_B(\lambda)$ be the sensitivities of the sensors for red, green, and blue light. The responses of an image capture system to a light signal $E(\lambda)$ reflected at the surface $S(\lambda)$ become

$$R = \int S(\lambda)E(\lambda)h_R(\lambda)d\lambda, \tag{6.1}$$

$$G = \int S(\lambda)E(\lambda)h_G(\lambda)d\lambda, \text{ and} \tag{6.2}$$

$$B = \int S(\lambda)E(\lambda)h_B(\lambda)d\lambda. \tag{6.3}$$

The sensor output is the integral of three different wavelength-dependent components : the source E, the surface reflectance S, and the sensor sensitivities h_R, h_G, and h_B. R, G, and B become after sampling (quantizing) red, green, and blue pixel values. Color spaces are a way of organizing the colors perceived by human beings, and R, G, and B form such a color space.

Luminance is the luminous intensity of a surface in a given direction per unit of projected area [MW00]. The luminance of a pixel value is often defined as the sum of its three color components, $R+G+B$. Since we want to track an object under differing lighting conditions, we want to detect color independently of its luminance. For this we map the three sensor responses to a chromaticity space containing only the color information of the signal. *Chromaticity is the quality of color characterized by its dominant or complementary wavelength and purity taken together* [MW00]. Chromaticity explicitly ignores intensity or brightness ; it is a plane in the three-dimensional color space. Figure 6.1 illustrates the transition from the three dimensional RGB color space to the rg chromaticity diagram.

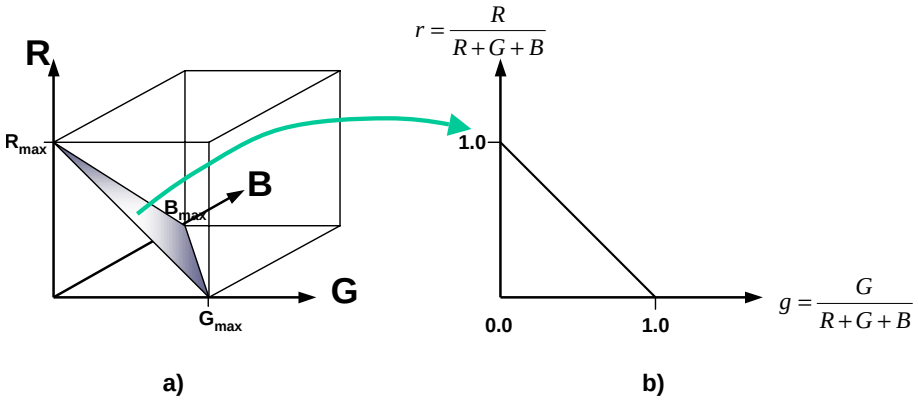


FIG. *6.1: RGB Color cube and its mapping to the rg chromaticity space.* R_{max}, G_{max}, *and* B_{max} *are* $2^8 - 1 = 255$ *each in a 24 bit pixel resolution.*

A common chromaticity space is called *rg chromaticity space* and is defined by

$$r = \frac{R}{R+G+B} \tag{6.4}$$

$$g = \frac{G}{R+G+B} \tag{6.5}$$

Equations 6.4 and 6.5 are sufficient to express any color since the normed blue component

$$b = \frac{B}{R+G+B} = 1-r-g$$

Any perceived color from the three-dimensional color space spanned by R, G, and B can thus be displayed in a two-dimensional diagram. The most common chromaticity diagram is that of the *C.I.E., the International Commission on Illumination (Commission International d'Eclairage)*, shown in figure 6.2. The choice of $(\lambda_R, \lambda_G, \lambda_B) = (410, 530, 650)nm$ maximizes the realizable colors, but some color still cannot be realized since they would require negative values for some r, g, and b.

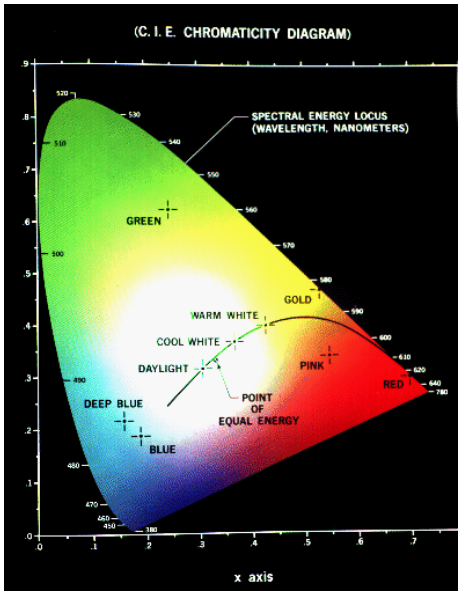


FIG. *6.2: C.I.E. Chromaticity diagram (source : CIE) The outline of the colored area represents the hue, while the distance of a point to the white point represents the saturation of that point.*

6.1.1 Color Histograms

In an image sampled from a digital camera, all
values R, G, and B and therefore r and g are discrete values. This means that r and g can be used as color axes to create 2D color histograms from images. Given a discrete color space defined by some color axes, the color histogram is obtained by counting the number of times each color occurs in an image array. Histograms are invariant to translation and rotation about an axis perpendicular to the image plane, and change only slowly under change of angle of view, change in scale and occlusion. Because histograms change slowly with view, a three-dimensional object can be adequately represented by a small number of histograms, corresponding to a set of canonical views. Swain and Ballard have shown how a histogram of color vectors can be back-projected to detect the pixels which belong to an object [SB91]. Schiele and Waibel showed that for face detection, color RGB triples can be divided by the luminance to remove the effects of relative illumination direction [SW95]. Other than in tracking, there are other techniques in computer vision using histograms, e.g., for video or image indexing. A description of these goes beyond the scope of this thesis.

6.1.2 Skin color

Detecting pixels with the color of skin provides a reliable method for detecting and tracking faces. The statistics of the color of skin can be recovered from a sample of a known face region and then used in successive images to detect skin colored regions. A good model for reflection on a skin surface is the *dichromatic reflection model* [SAG00, KSK90]. It models the surface reflectance, $S(\lambda)$, from the equations 6.1, 6.2, and 6.3, as the weighted sum of the interface reflectance $S_i(\lambda)$ and the body reflectance $S_b(\lambda)$.

$$S(\lambda) = m_i(\Theta)S_i(\lambda) + m_b(\Theta)S_b(\lambda) \tag{6.6}$$

The spectral power *distribution* of the reflected light is the *same* as that of the incident light.

The interface reflection of human skin takes place at the *epidermis*, the thin surface layer of skin. The epidermis is *the outer nonsensitive and nonvascular layer of the skin of a vertebrate that overlies the dermis* [MW00]. The epidermis reflects 5% of the incident light independently of its wavelength and the melanin content of the epidermis, which is different for humans with different genetic background. The rest of the incident light is absorbed in the epidermis and dermis layers of the skin. The dermis is *the sensitive vascular inner mesodermic layer of the skin* [MW00]. Figure 6.3 illustrates the light reflectance/absorption process of human skin. Unfortunately, skin color can not be described by one single reflectance curve, since skin color is composed of many colors around a mean, depending on the blood content in the dermis or the melanin concentration (i.e., the number of dark pigments) in the epidermis [SAG00]. We can, however, create a histogram of a representative color sample and take it as a feature we search in an entire image using the histogram of the entire image.

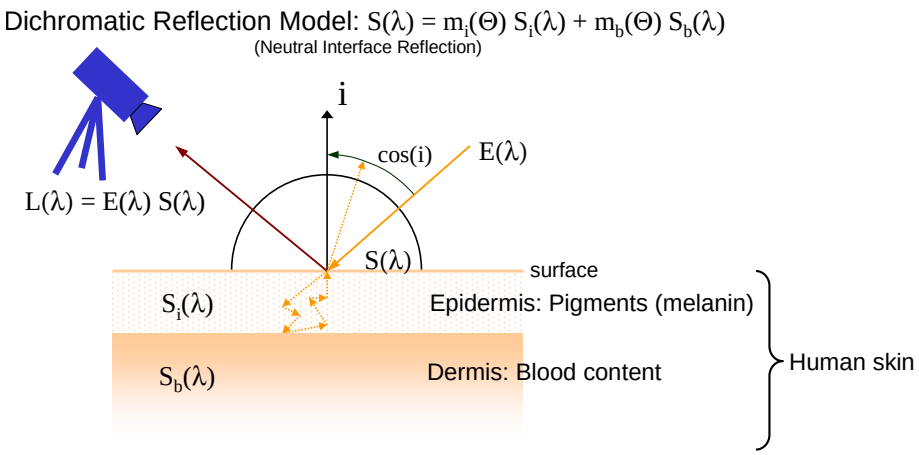


FIG. *6.3: Skin surface reflectance.*

6.2 Probability of Skin

The intensity-normalized pixels from a region of an image known to contain skin can be used to define a two dimensional histogram, $h_{\text{skin}}(r,g)$, of skin color. The effects of digitizing noise can be minimized by smoothing this histogram with a small Gaussian filter. A second histogram, $h_{\text{total}}(r,g)$, can be made from all of the pixels of the same image. This second histogram should also be smoothed by the same filter. These two histograms make it possible to apply Bayes' rule to each pixel of an image to obtain the probability that a given pixel is skin.

6.2.1 Review Bayes' Rule

Given a sample space S, and two events A and B in that sample space, then the conditional probability of A given B is

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{6.7}$$

Re-expressing 6.7 gives a useful form for calculating intersection probabilities,

$$P(A \cap B) = P(A|B)P(B), \tag{6.8}$$

and, exploiting the symmetry of 6.7,

$$P(A \cap B) = P(B \cap A) = P(B|A)P(A), \tag{6.9}$$

We now can equate the right-hand sides of 6.8 and 6.9 to obtain after rearrangement

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{6.10}$$

Here is the definition of Bayes' Rule, which has a more general form than 6.10 [CB90].

Definition 1 (Bayes' Rule) *Let $A_1, A_2, \ldots$ be a partition of the sample space, and let B be any set. Then, for each $i = 1, 2, \ldots$*

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{\infty} P(B|A_j)P(A_j)} \tag{6.11}$$

6.2.2 Conditional Probability of Skin

We now go about finding a probability function to create a probability map for skin colored pixels in an image. We start by hand-selecting an area of skin colored pixels. Let $h_{\text{skin}}(r,g)$ be the histogram of intensity normalized colors from a region of an image known to represent skin, and

$$N_{\text{skin}} = \sum_{r,g} h_{\text{skin}}(r,g) \tag{6.12}$$

the number of pixels in that image having skin color. We can then approximate the probability of a color vector, (r,g), given skin by

$$p(r,g|\text{skin}) \approx \frac{1}{N_{\text{skin}}} h_{\text{skin}}(r,g) \tag{6.13}$$

Let further $h_{\text{total}}(r,g)$ be the histogram of intensity normalized colors from the entire image, and

$$N_{\text{total}} = \sum_{r,g} h_{\text{total}}(r,g) \tag{6.14}$$

the total number of pixels in that image. The probability of observing a certain color vector is given by :

$$p(r,g) \approx \frac{1}{N_{\text{total}}} h_{\text{total}}(r,g) \tag{6.15}$$

The overall probability of obtaining a skin pixel in an image is approximated by the fraction of observed pixels known to be skin, provided all skin pixels are labeled.

$$p(\text{skin}) \approx \frac{N_{\text{skin}}}{N_{\text{total}}} \tag{6.16}$$

Equation 6.16 is valid only if N_{skin} is the real number of all skin colored pixels in the image. We make sure that this is the case by bootstrap re-sampling of all skin colored pixels in the image and recompute h_{skin} at initialization time of the algorithm. The application of 6.10 using 6.13 through 6.15 in order to calculate the conditional probability $p(\text{skin}|r,g)$ is straightforward. The

probability of skin given a color vector (r,g) is

$$\begin{aligned} p(\text{skin}|r,g) &= \frac{p(r,g|\text{skin}) \cdot p(\text{skin})}{p(r,g)} \\ &= \frac{\frac{1}{N_{\text{skin}}} h_{\text{skin}}(r,g) \cdot \frac{N_{\text{skin}}}{N_{\text{total}}}}{\frac{1}{N_{\text{total}}} h_{\text{total}}(r,g)} \\ &= \frac{h_{\text{skin}}(r,g)}{h_{\text{total}}(r,g)} \end{aligned} \tag{6.17}$$

The ratio of these two histograms gives a table which directly converts an intensity-normalized pixel (r,g) into the probability that the pixel is skin, $p(\text{skin}|r,g)$, by table lookup. A default value of 0 is placed in this table for all pixels (r,g) for which $h_{\text{total}}(r,g)$ is zero. Strictly speaking, equation 6.17 is only valid for the image from which the skin sample was obtained. In practice, we have found that the technique will work well for subsequent images provided that the color of the scene illumination does not change. This table is trivial to build and may be renewed whenever an independent source has detected the face in the image.

A number of authors such as [JP99] have indicated a preference for using Gaussian mixture models in place of the two histograms. Our experience is that such a model provides a very slight improvement in the probability image, at a very great cost in computation whenever the histogram must be renewed, making frequent update of the histogram ratio impractical. For a real-time system, the robustness obtained by frequently renewing the histogram ratio table greatly exceeds the slight improvement observed with a static mixture of Gaussian models.

6.3 CENTEROFGRAVITY algorithm

Having computed a probability for every pixel to be skin colored, we will now identify the principal object having skin color, i.e., the object – usually the face – we want to track. For the time being, we ignore the possibility of other skin colored objects in the image. We will handle that case in the next section. In order to detect a skin color region we must group skin pixels into a region. This region is described by the center and the (co-)variances of a two-dimensional probability distribution for skin colored pixels. This probability distribution will have an elliptic shape, which adapts nicely to the shape of a face. Let $P_{\text{skin}}(i,j)$ represent the probability map of skin for each color pixel $(r(i,j),g(i,j))$ at position (i,j).

$$P_{\text{skin}}(i,j) = p(\text{skin}|r(i,j),g(i,j)), \tag{6.18}$$

where $p(\text{skin}|r(i,j),g(i,j)) = p(\text{skin}|r,g)$ from equation 6.17. The center of gravity or first moment of the probability map gives the position

$$\vec{\mu} = \begin{bmatrix} \mu_i \\ \mu_j \end{bmatrix}, \tag{6.19}$$

and the covariance matrix, **C**, containing the second moments the spatial extent of the skin colored region.

$$\mathbf{C} = \begin{bmatrix} \sigma_i^2 & \sigma_{ij} \\ \sigma_{ij} & \sigma_j^2 \end{bmatrix} \tag{6.20}$$

The elements of $\vec{\mu}$ and **C** are

$$\begin{aligned} \mu_i &= \frac{1}{S}\sum_{i,j} P_{\text{skin}}(i,j)\cdot i, \\ \mu_j &= \frac{1}{S}\sum_{i,j} P_{\text{skin}}(i,j)\cdot j, \\ \sigma_i^2 &= \frac{1}{S}\sum_{i,j} P_{\text{skin}}(i,j)\cdot (i-\mu_i)^2, \\ \sigma_j^2 &= \frac{1}{S}\sum_{i,j} P_{\text{skin}}(i,j)\cdot (j-\mu_j)^2, \quad \text{and} \\ \sigma_{ij} = \sigma_{ji} &= \frac{1}{S}\sum_{i,j} P_{\text{skin}}(i,j)\cdot (i-\mu_i)(j-\mu_j), \end{aligned}$$

where $S = \sum_{i,j} P_{\text{skin}}(i,j)$. Figure 6.4 illustrates the probability map generated.

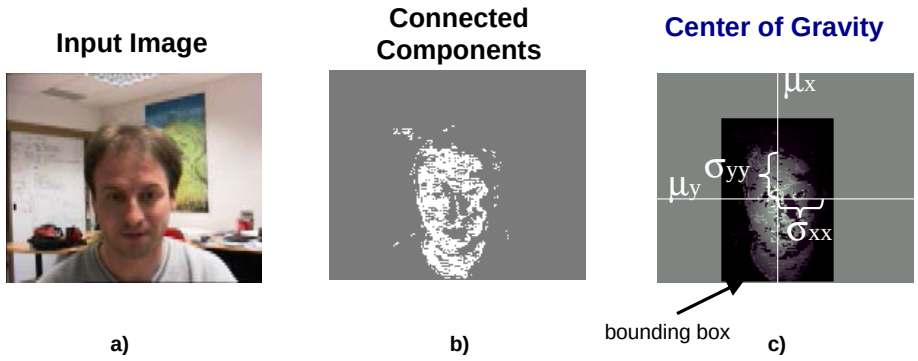


FIG. *6.4: Probability map of c)* CENTEROFGRAVITY *(COG) and b)* CONNECTEDCOMPONENTS *(CCO) algorithm, a) showing the input image. Graylevels indicate probabilities, where white stands for probability = 1, and black for probability = 0. Also shown is the region of interest for the COG algorithm. Using a region of interest significantly speeds up computing.*

6.3.1 Robust estimation and tracking

Unfortunately, skin color pixels in any other part of the image will contribute to these two moments. This effect can be minimized by *weighting* the probability image with a Gaussian function placed at the location where the face is expected. The initial estimate of the covariance of this Gaussian should be the size of the expected face. Once initialized, the covariance is estimated recursively from the previous image.

For each new image at time t, a two dimensional Gaussian estimate of the previous distribution of skin colored pixels is generated using the mean and covariance from the previous image at time $t-\Delta T$,

$$p(i,j;\vec{\mu}_{t-\Delta T},\mathbf{C}_{t-\Delta T}) = g(i,j;\vec{\mu}_{t-\Delta T},\mathbf{C}_{t-\Delta T}) = \frac{1}{\sqrt{2\pi}\det(\mathbf{C}_{t-\Delta T})^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\begin{pmatrix} i \\ j \end{pmatrix}-\begin{pmatrix} \mu_{i,t-\Delta T} \\ \mu_{j,t-\Delta T} \end{pmatrix}\right)^T \mathbf{C}^{-1}\left(\begin{pmatrix} i \\ j \end{pmatrix}-\begin{pmatrix} \mu_{i,t-\Delta T} \\ \mu_{j,t-\Delta T} \end{pmatrix}\right)} \tag{6.21}$$

This two-dimensional Gaussian estimate is multiplied with the new probability distribution as shown in equation 6.22 to give new estimates for the mean and covariance.

$$\begin{aligned}
\mu_{i,t} &= \frac{1}{S}\sum_{i,j} P_{\text{skin}}(i,j)\cdot i \cdot g(i,j,\vec{\mu}_{t-\Delta T},\mathbf{C}_{t-\Delta T}) \\
\mu_{j,t} &= \frac{1}{S}\sum_{i,j} P_{\text{skin}}(i,j)\cdot j \cdot g(i,j,\vec{\mu}_{t-\Delta T},\mathbf{C}_{t-\Delta T}) \\
\sigma^2_{i,t} &= \frac{1}{S}\sum_{i,j} P_{\text{skin}}(i,j)\cdot (i-\mu_{i,t})^2 \cdot g(i,j,\vec{\mu}_{t-\Delta T},\mathbf{C}_{t-\Delta T}) \\
\sigma^2_{j,t} &= \frac{1}{S}\sum_{i,j} P_{\text{skin}}(i,j)\cdot (j-\mu_{j,t})^2 \cdot g(i,j,\vec{\mu}_{t-\Delta T},\mathbf{C}_{t-\Delta T}) \\
\sigma_{ij}(t) = \sigma_{ji}(t) &= \frac{1}{S}\sum_{i,j} P_{\text{skin}}(i,j)\cdot (i-\mu_{i,t})(j-\mu_{j,t}) \cdot g(i,j,\vec{\mu}_{t-\Delta T},\mathbf{C}_{t-\Delta T}),
\end{aligned} \tag{6.22}$$

where $S = \sum_{i,j} P_{\text{skin}}(i,j)\cdot g(i,j,\vec{\mu}_{t-\Delta T},\mathbf{C}_{t-\Delta T})$. The effect of multiplying new images with the Gaussian function is that other objects of the same color in the image (hands, arms, or another face) do not disturb the estimated position of the region being tracked.

6.3.2 Behavior Discussion

The use of a Gaussian weighting function for new input data actually can lead to a problem, if the object being tracked moves above a certain speed. Let us therefore consider the dynamic behavior of the robust color tracker. Since the two-dimensional case is largely more complex, we present here the one-dimensional case which actually suffices to justify our compensation

measures. Let

$$g(i|\mu(t),\sigma(t)) = \frac{1}{\sqrt{2\pi}\sigma(t)} e^{-\frac{1}{2}\left(\frac{i-\mu(t)}{\sigma(t)}\right)^2} \tag{6.23}$$

an approximation of the *probability density function (pdf)* of the new input data (image), and

$$g(i|\mu_{t-\Delta T},\sigma_{t-\Delta T}) = \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{1}{2}\left(\frac{i-\mu_{t-\Delta T}}{\sigma_{t-\Delta T}}\right)^2} \tag{6.24}$$

be the pdf of the weighting function. Both distribution functions are functions of the same variable *i*, but with different mean and variance. The resulting, i.e., effectively detected distribution is then product of both functions :

$$\begin{aligned} &g(i,\mu(t),\sigma(t)) \cdot g(i,\mu_{t-\Delta T},\sigma_{t-\Delta T}) = \\ &\quad \frac{1}{\sqrt{2\pi}\sigma(t)} \exp\left[-\frac{1}{2}\left(\frac{i-\mu(t)}{\sigma(t)}\right)^2\right] \cdot \frac{1}{\sqrt{2\pi}\sigma_{t-\Delta T}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu_{t-\Delta T}}{\sigma_{t-\Delta T}}\right)^2\right] = \\ &\quad\quad \frac{A}{\sqrt{2\pi}\sigma_{new,t}} \exp -\frac{1}{2}\left(\frac{i-\mu_{new,t}}{\sigma_{new,t}}\right)^2, \end{aligned} \tag{6.25}$$

where

$$A = \frac{\sigma(t)\ \sigma_{t-\Delta T}}{\sqrt{2\pi}\ (\sigma^2(t)+\sigma^2_{t-\Delta T})} \exp\left[-\frac{1}{2}\frac{(\mu(t)-\mu_{t-\Delta T})^2}{(\sigma^2(t)+\sigma^2_{t-\Delta T})}\right] \tag{6.26}$$

$$\mu_{new,t} = \frac{\mu(t)\sigma^2_{t-\Delta T} + \mu_{t-\Delta T}\sigma^2(t)}{(\sigma^2(t)+\sigma^2_{t-\Delta T})} \tag{6.27}$$

$$\sigma^2_{new,t} = \frac{\sigma^2_{t-\Delta T}\sigma^2(t)}{\sigma^2(t)+\sigma^2_{t-\Delta T}} \tag{6.28}$$

Equation 6.25 is a function with the *shape* of a Gaussian function which integrates from $-\infty$ to $+\infty$ to *A*. However, it is *not* a *pdf*, since it does not integrate from $-\infty$ to ∞ to 1. The presence of the factor *A* poses some interesting problems onto the behavior of the color tracker, which are being discussed in the following.

6.3.3 Compensation for weighting

Let us first consider the case where the tracked object stays in the same position for subsequent images, i.e., $\mu(t) = \mu_{t-\Delta T}$. Let us further assume that the tracked object does also not change its size, which is equivalent to assuming $\sigma(t) = \sigma_{t-\Delta T} = \sigma_0$.

$$\sigma^2_{new,t} = \frac{\sigma^2_{t-\Delta T}\sigma^2(t)}{\sigma^2(t)+\sigma^2_{t-\Delta T}} = \frac{\sigma_0^2\sigma_0^2}{2\sigma_0^2} = \frac{\sigma_0^2}{2} \tag{6.29}$$

That is, if we assume the distribution function of the tracked object – even if it does not move – to be the same for a couple of subsequent images, the combined pdf of weighting and distribu-

tion function will shrink with each cycle, if no measure of compensation is taken. Equation 6.29 suggests a compensation factor of $2\sigma^2_{new,t} = \sigma^2_0$.

6.3.4 Motion compensation

If the tracked object moves, then the center of the combined pdf will lie between the center of the weighting function and the center of the pdf of the new input data. Equation 6.27 shows that the relation $\sigma(t)/\sigma_{t-\Delta T}$ determines if the new center is closer to the center of the weighting pdf or to the center of the new input pdf. For $\sigma(t) \approx \sigma_{t-\Delta T} = \sigma_0$ (a likely case for a sufficiently high frequency), the combined center will lie in the middle of both distribution functions.

Depending on the speed of the tracked object, equations 6.26 and 6.27 can cause the combined pdf to vanish, thus breaking the tracker. Let

$$v_{i,t} = \frac{\Delta\mu_{i,t}}{\Delta T} = \frac{\mu_{i,t} - \mu_{i,t-\Delta T}}{\Delta T} \tag{6.30}$$

$$v_{j,t} = \frac{\Delta\mu_{j,t}}{\Delta T} = \frac{\mu_{j,t} - \mu_{j,t-\Delta T}}{\Delta T} \tag{6.31}$$

be the detected object's speed at time t in horizontal (i) and vertical (j) direction, where ΔT is the time elapsed since the last frame was processed. Experiments with compensation measures suggest a Kalman filter like update function

$$\mathbf{C}_{new,t} = \mathbf{C}_t + \Delta T^2 \cdot \begin{bmatrix} v^2_{i,t} & 0 \\ 0 & v^2_{j,t} \end{bmatrix} = \mathbf{C}_t + \cdot \begin{bmatrix} \Delta\mu^2_{i,t} & 0 \\ 0 & \Delta\mu^2_{j,t} \end{bmatrix} \tag{6.32}$$

Equation 6.32 compensates uncertainty due to accelerations and object movements. Combining the results of equations 6.29 and 6.32 we get as the covariance matrix of the Gaussian weighting function for new incoming data :

$$\mathbf{C}'_{new,t} = 2 \begin{bmatrix} \sigma^2_{i,t} + \Delta\mu^2_{i,t} & \sigma_{ij,t} \\ \sigma_{ij,t} & \sigma^2_{j,t} + \Delta\mu^2_{j,t} \end{bmatrix} \tag{6.33}$$

6.3.5 Compute orientation by PCA

It may be desirable to know the orientation of the ellipse that the color tracking algorithm has given in the form of the covariance matrix $\mathbf{C}$. This might be helpful in order not only to center the tracked object, but also re-orient it so that it has always the same upright position relative to the image borders. By assuming a 2D Gaussian distribution as the underlying distribution function for the tracked object (or better : its color), we assume the object to have an elliptic shape. The corresponds nicely to the shape of a face. The eigenvectors of the covariance matrix represent the axes of this ellipse.

6.3.6 Noise

Calculating the first and second moments instead of the connected components of a thresholded probability map for skin color of an image levels out random noise in the image. Random noise may be sampling/digitizing noise, or quantizing noise (histogram). Noise is the main reason for jitter.

6.3.7 Performance

In order to do a comprehensive assessment of the performance of the skin color tracker described above, we use the color tracker introduced in [CBC97] as a benchmark. This tracker, called *CCO* for CONNECTEDCOMPONENTS algorithm hereafter, is based on thresholding and a connected components algorithm in order to identify a large skin colored region within an image which is interpreted as a face. A face is represented as an image position, vertical and horizontal extent, and a confidence factor corresponding to the relative distance of the *detected* face and the *expected* face. The difference in execution times are due to a different complexity of the employed algorithms as shown in table 6.1. Although both algorithms have a complexity of $O(n)$, the algorithms used by the CCO algorithm are computationally less costly.

TAB. *6.1: Complexity of the algorithms employed by the robust estimator (COG) and the tracker using the connected components algorithm (CCO). n is the number of images the tracker is applied to. A 1 indicates that this task has only to be performed once, usually at initialization time.*

Tasks	Algorithm	
	Connected Components	Robust Estimator
Generate 2D histogram of skin color	1	1
Generate 2D histogram of entire image	–	1
Generate probability map (includes computing the first and second moments)	–	n
Find connected components	n	–

The robust tracking algorithm, called *COG* hereafter, carries a slightly higher computational cost than a connected components of a thresholded image. This is illustrated with the computing times shown in Figure 6.5. This figure shows the execution time for QCIF [ITU93] sized images on a PC under Linux equipped with a 333 MHz Intel processor. Average execution times are around 13.8 milliseconds per image for the connected components and 17.6 milliseconds for the robust algorithm.

Jitter is the number of pixels that the estimated position moves when the target is stationary. Jitter is the result of interference with illumination, electrical noise, shot noise, and digitizer noise.

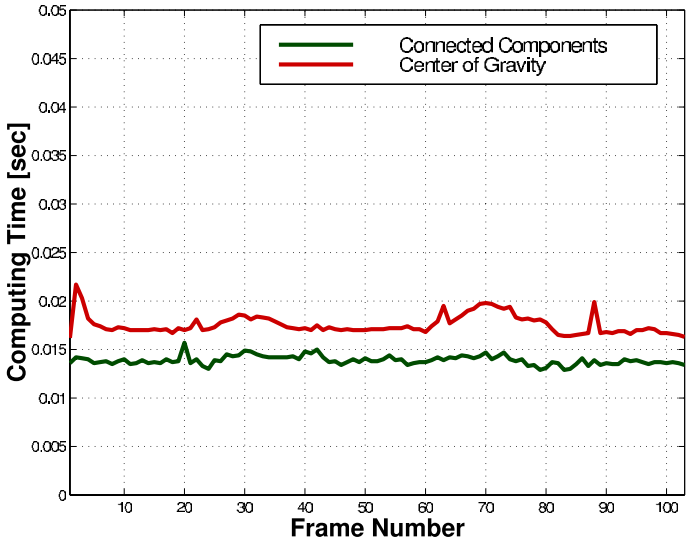


FIG. *6.5: Computing time per image for the robust estimator (COG), and the connected components based algorithm (CCO).*

Algorithms which employ a threshold are especially sensitive to such noise. Table 6.2 illustrates the reduction in jitter for the robust tracker when compared to connected components. The numbers shown are the accumulated error over the entire sequence.

TAB. *6.2: Jitter energy measured for a stationary object recorded in a sequence of 105 images; results are given for the robust estimator (COG), and by connected components with (CCO w/ KF) and without (CCO w/o KF) a Kalman Filter.*

	COG	CCO w/o KF	CCO w/ KF
Jitter Energy	29	308	151

Figure 6.6 compares the precision of tracking an object moving in the horizontal direction. All three trackers were applied to the same image sequence. The output of the color tracker using the connected components algorithm is shown with and without Kalman filter. The Kalman filter eliminates position jitter but reduces precision of global position estimation.

It should be noted that the CENTEROFGRAVITY algorithm loses its advantages over a thresholding algorithm such as the connected components algorithm, if the tracked object's surface

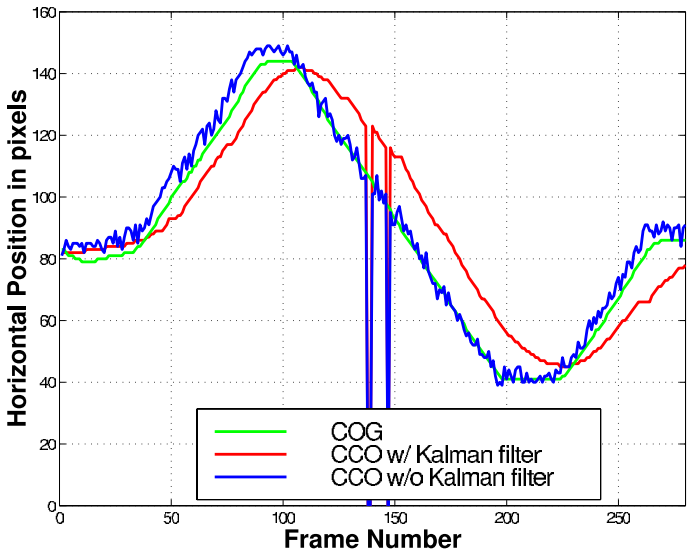


FIG. *6.6: Comparing tracking precision of a moving object. The dips in the blue and red curve are failures of the CCO tracker. The COG tracker is hence more precise and more robust.*

has uniform color. This is the case for matte colored objects. For surfaces with varying color such as skin (see section 6.1.2) or with a varying color reflectance, the COG performs considerably better.

6.4 Conclusion

This chapter showed how skin color can be detected to build an efficient and precise face tracker. Color histograms are used to calculate the probability that a pixel has skin color. Instead of thresholding probabilities, we use the CENTEROFGRAVITY algorithm to compute the first and second moments which give position and size of the tracked object. Weighting new input images with a Gaussian estimate based on the previous image makes the tracker robust. The output of the CENTEROFGRAVITY algorithm is used to steer a camera to keep a tracked object in the center of the image. Cutting out a region of interest further eliminates possible jitter due to a lag in reaction time between computer and camera.

The CENTEROFGRAVITY algorithm is an elegant way to avoid thresholding situations. It may be applied to other techniques such as image differencing and background subtraction. The CENTEROFGRAVITY algorithm for skin color tracking has been successfully re-implemented several times.

We will see in the next chapter how tracking reduces bandwidth by reducing the number of required basis dimensions for eigenspace based coding.

Chapitre 7

Orthonormal Basis Coding

Orthonormal Basis Coding (OBC) has been designed as an alternative approach to creating models of image content and projecting the image content onto the model(s) [HL96]. The idea is to store or transmit only a set of parameters, but matching image content onto a model is very computing power consuming. Moreover, model-based[1] approaches have the disadvantage that encoding (and decoding) becomes dependent on models of image content like heads or hands. Figure 7.1 illustrates the difference between a) conventional video coding methods such as described in chapter 5, and b) the OBC approach.

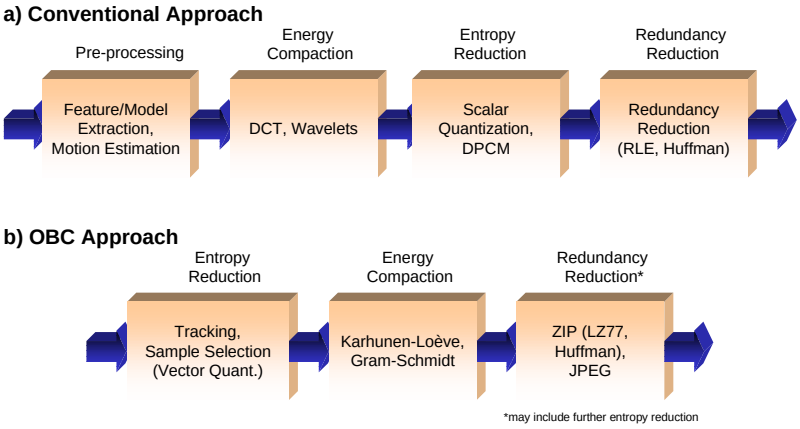


FIG. *7.1: Block diagrams of the a) conventional and b) the OBC approach*

We propose an alternative to a model-based approach to eliminate redundant image information by interpreting images, i.e., samples of appearance, as (high-)dimensional vectors and creating

[1] "Models" in this context are geometrical models, not appearance models.

an orthonormal basis space from a sequence of such vectors. This corresponds to a decomposition of a sequence of vectors into basis vectors which are orthonormal to each other. This is the fundamental idea of *all* transform-based video coding algorithms. Like any other linear transformation, the idea is to decorrelate video (image) data allowing for a controlled entropy reduction and an efficient redundancy removal in order to yield high compression ratios.

What characterizes our approach is that feature extraction is an integral part of the encoding process instead of just being a pre-processing step. We use the term *Orthonormal Basis Coding (OBC)* as a name for our approach to video compression, including not only the transform coding process, but also the entropy reduction and the final redundancy reduction steps.

Section 7.1 gives an overview of the OBC concept as an encoding procedure, using the results of the previous chapters and sections. Section 7.2 elaborates on the difference in the treatment of video data between the conventional approach and the OBC approach, motivating this work. Section 7.3 takes a closer look to the problem of the estimation of the mean of our input data. This is always a critical step when computing an orthonormal basis. Section 7.4 is a contribution to the discussion of evaluation metrics for video compression.

7.1 The Concept

The OBC compression scheme operates as follows :

a. Normalize the input stream to stabilize the main face position in the center of the image, and choose a limited set of images from the sequence to form the basis. For this sample selection process, we use in virtually all cases the simple but efficient EQUIDISTANT selection algorithm. Note that we could improve our compression performance using K-MEANS or MAXMIN-DISTANCE clustering at the cost of increasing the required computing time. All these algorithms and their performance are discussed in section 7.3.

b. Compute orthonormal basis space or sub-space either by Karhunen-Loève expansion (KLE), or by the incremental Gram-Schmidt procedure, and project each image in the sequence into this basis space, resulting in a small set of coefficients,

c. ZIP loss-less compression or JPEG lossy compression of the basis and, if sensible, the parameter vectors. The basis vectors are converted from a 4 Byte float representation into 1 Byte per value, and the resulting image compressed. The question if it makes sense to compress the parameter vectors depends on the quantity of data they represent. In general there is no substantial gain and therefore no reason to do so.

The projection of an image into the basis space will produce a number of coefficients equal to the number of images used to create the basis space.

A stabilized video sequence is cropped in order to provide a sequence of images with the face normalized and centered in each image. Selected frames from the sequence are used to create a basis space into which new images can be mapped. Each mapped image is represented as a vector of coefficients. The number of coefficients is equal to the number of images in the original

basis space. By only storing and transmitting the vectors, extremely high compression rates can be achieved, especially for long sequences.

The principal encoding parameters for OBC are the choice of the selection algorithm, the basis size, the choice of the basis compression algorithm, and – in case of JPEG basis compression – the JPEG quality parameters. Once the basis images are selected as cluster centers, all images from the sequence are assigned to a cluster. For all selection algorithms the cluster center is recomputed as the mean of all samples in the cluster. In case of the K-MEANS algorithm, this might happen several times. The average image in each cluster is then used to compute the basis by Karhunen-Loève expansion. Images close to the cluster centers have therefore a low reconstruction error.

We identify three distinct cases for video compression. Two criteria are used to describe them.

Universe is the set of possible data to be represented by the code. An *open universe* means that at the time the encoding starts, we do not know all the data that has to be encoded. That is, the data is either still being generated, or the data access is restricted for one or the other reason. *Closed universe* describes the complementary case where all the (video) data, i.e., the entire sequence that is to be encoded, is known and accessible at encoding time.

Online/Offline : This criterion describes the manner in which the data is to be encoded. *Online* coding is characterized by a sequential data feed-in into the codec. The data is touched only once by the codec and immediately encoded. The codec has a FIFO character, and any algorithms employed must be one-pass. Moreover, the entire data processing should possibly be computed in real-time. *Offline coding* an evaluation of the input data in several steps, e.g., we *can* use iterative sample selection algorithms to identify the input vectors to create the orthonormal basis space. Additionally, computing time restrictions are more relaxed (within a certain limit), and compression ratio or reconstruction quality becomes more important than computing speed.

7.1.1 Closed universe, offline coding

The case of closed universe, offline coding is perhaps the "simplest" of all possible cases : Given a recorded sequence of images, the codec has unlimited access to the data and can take its time to assemble an efficient code. True eigenspace coding of the entire sequence is (still) not recommendable for computing time reasons. However, the number of basis dimensions can be chosen comfortably large in order to guarantee a good reconstruction quality. The offline character allows to use any of the selection algorithms presented in section 7.3, whichever gives the best results.

Application examples for a closed universe, offline codec are

Video (e-)mail : A sequence is recorded to be transmitted like an email. Such a system is anticipated in Barry Levinson's interpretation of Michael Crichton's bestseller *Disclosure* (1994).

Video/Movie compression : Video file formats such as QuickTime from Apple Co. offer the possibility to include different compression modules such as JPEG or MPEG. An OBC module for QuickTime is conceivable and immediately implementable using the provided *user defined* chunks.

Web head : Interactive video might be a tool for enhanced user convenience which companies selling over the Web or T.V. could use. An example is a talking head on web pages to guide a user through the product selection process.

7.1.2 Open universe, offline coding

Encoding an unlimited set of images with a constraint basis makes *a priori* sense only if either the variance of the input data is limited, or if the basis size is sufficiently large to ensure decent reconstruction quality. Restraining the variance of the encoded data, however, may become a feature in the context of *privacy protection* [CCBS97]. Note that *offline* refers to the basis building rather than the encoding/decoding process.

As an example, imagine a basis obtained from a talking head sequence of a person. If that person wants to communicate, he opens a communication session, uses the OBC codec to encode his session, but transmitting the pre-recorded sequence of him as a basis. Given that the content of the sequences are close enough, the transmitted encoding parameters can reconstruct a communication scene on the other end using the pre-recorded basis. This allows the communicating person to hide things like background, other people present in the room, personal features such as haircut, beard, maybe facial expressions. A reasonable complete basis of anybody could even be used. Fun examples could be using a certain actor or pop star as basis. Assuring reconstruction quality is a particularly difficult problem here.

7.1.3 Open universe, online coding

This is finally the communication scenario live and online. It requires incremental, one-pass algorithms to build the basis. Computing speed is important to ensure acceptable delays, compression ratio and reconstruction quality are a parameter determined by the specs of the communication channel. The number of basis dimensions have to be chosen as a function of those parameters. The online character requires one-pass selection algorithms. Video telephone and video conference are common examples for this scenario.

7.2 The Video Compression Problem revisited

Before we try to find an answer to "How to compress ?" we have to take a look at "What to compress ?". The current section gives the reader a description of the data we are working with. We will show that the way we look at our data has an impact on how we tackle the compression problem.

Digital images are commonly considered two-dimensional signals, usually quantized to 255 levels of red, green, and blue. The situations that are of interest for us are that the images have a certain content. If that content is a mapping of a 3D-scene to a two-dimensional canvas, then the

picture elements (short : pixels) usually have a very high statistical correlation. For instance, if a pixel represents a part of a sky in the image and has a bluish color, then the probability for the neighboring pixels to have the same or a very similar color is very high (see, e.g., figure 7.2). Current video encoding algorithms based on linear transforms, as well as feature extraction modules like edge detection filters exploit this statistical correlation.

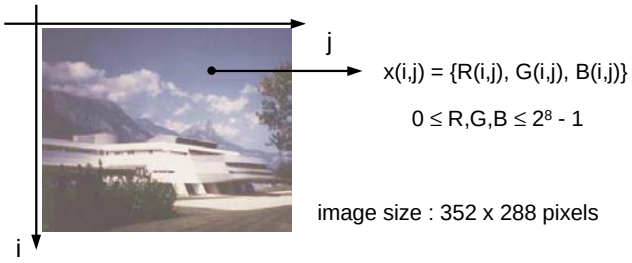


FIG. *7.2: Example of an 24 bits/pixel color image with 3 x 8 bits / pixel color encoding.*

Considering images not singularly, but as a part of a sequence of images, adds a third dimension to the space from which we extract our information (horizontal, vertical, temporal). But adding a time axis also adds more statistical correlation, this time between consecutive images. Figure 7.3 shows an example of a video sequence. We note that pixels representing the background do not change from one image to the next, unless the camera zoom or pan/tilt position changes. This is the conventional way to look at image sequences, which is closely related to how the processing for compression is supposed to be done. A second possibility to look at images in an image sequence is to consider them as vectors in a high-dimensional linear space. The actual number of dimensions is equivalent to the number of pixels in the image. A certain image with a certain content is then simply a *point* in that space

We saw in section 4.3 that the optimal representation for random vectors is the eigenspace spanned by the eigenvectors of their covariance matrix. A certain image is now simply a point in eigenspace.

Figures 7.4 and 7.5 show that images with related content have a dense representation in eigenspace, and even more so if the input sequence has been normalized. This is the starting point for our approach. *Instead of exploiting statistical correlation between image pixels, we exploit the closeness of images in eigenspace to create an efficient representation and to compress the image data.*

FIG. *7.3: Video sequence of a talking head communication scenario*

7.2.1 Appearance Space Analysis of the Video Compression Problem

Considering images as high-dimensional vectors instead of a three-dimensional data space, we reduce the dimensions of our data space to two. At first glance this has no particular advantage,

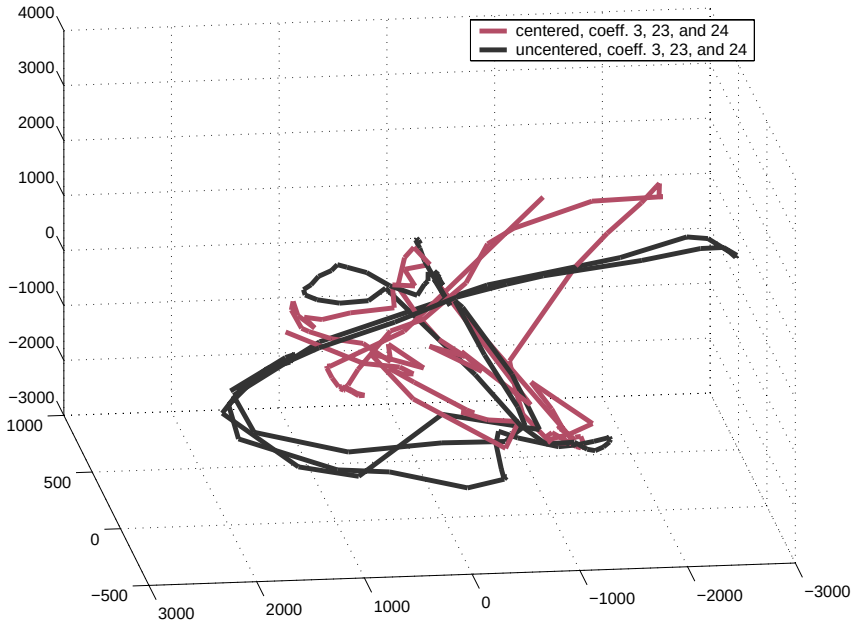


FIG. *7.4: Eigenspace representation of the* TALKINGHEAD *sequence, encoded with 25 basis dimensions; parameters 3 ($\sigma^2_{\text{centered}} = 14,241$, $\sigma^2_{\text{uncentered}} = 22,164$), 23 ($\sigma^2_{\text{centered}} = 1,486,308$, $\sigma^2_{\text{uncentered}} = 2,049,223$), and 24 ($\sigma^2_{\text{centered}} = 2,630,822$, $\sigma^2_{\text{uncentered}} = 7,170,681$) are sketched.*

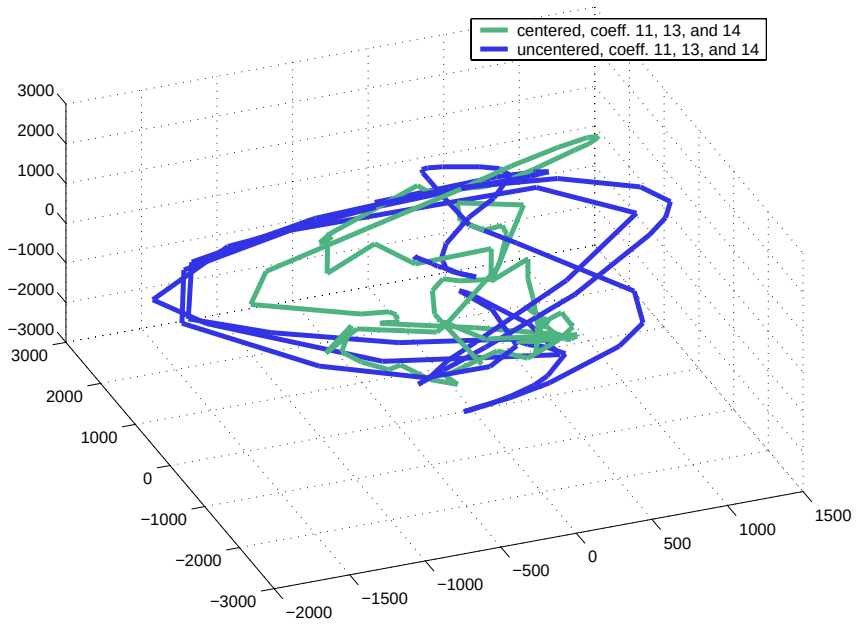


FIG. *7.5: Eigenspace representation of the* TALKINGHEAD *sequence, encoded with 15 basis dimensions ; parameters 11 (*$\sigma^2_{\text{centered}} = 635,332$*,* $\sigma^2_{\text{uncentered}} = 736,302$*), 13 (*$\sigma^2_{\text{centered}} = 1,498,268$*,* $\sigma^2_{\text{uncentered}} = 1,963,827$*), and 14 (*$\sigma^2_{\text{centered}} = 2,583,328$*,* $\sigma^2_{\text{uncentered}} = 7,121,455$*) are sketched.*

but it allows us to consider the encoding problem as a vector quantization or clustering problem. Another possibility is to consider it as a data analysis problem that we can tackle with, for instance, principal components analysis. Both approaches have advantages and disadvantages, but we will show in the following that, by combining the two, we can avoid some of those disadvantages and develop efficient video encoding algorithms.

Before discussing vector quantization and principal components analysis, let us develop the idea and consequences of regarding images as vectors. Much has been said about exploiting statistical correlation of image data for video compression. The consensus is that for even higher compression ratios, we need to gain information about the image content, extract that information, and encode how it is being changed rather than repeatedly encoding the information and its changes.

A common example is the simple video communication scenario. We see a talking head with a uniform or at least unimportant background, performing limited movements like talking, head shaking, nodding, etc. (see figure 7.3). The beginning of the 90's saw a great amount of research effort go into building models of heads, extracting information about a talking head out of an images stream, mapping this information onto the model. All that with the idea in mind that even-

tually all that had to be transmitted was a string of parameters about position, size and shape of the talking head. A summary of this approach is given in [HL96].

At the same time, parts of the computer vision community investigated on using computer vision methods for face and gesture recognition. Since 1997, the IEEE International Conference on Automatic Face and Gesture Recognition is being held annually, clearly displaying the importance (and the progress) in this research areas. Turk proposed eigenfaces [TP91] (i.e., eigenspace representations of images (=vectors) of faces) for face and gesture recognition in 1991. What is important from this work for this thesis is that the eigenspace can be used to represent or encode the essential features of an object like a face in an image.

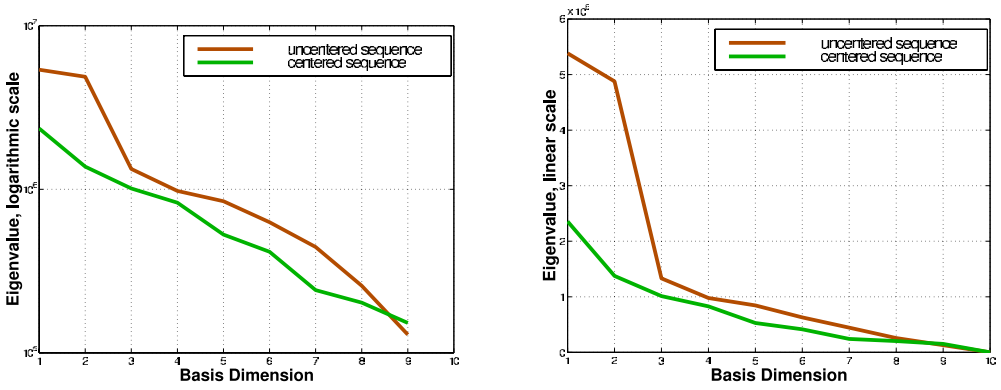
Since it only can access information about the surface of objects or *the* object shown in an image, i.e. as those objects *appear*, we refer to an eigenspace description of an object as its *appearance space*. See page II for definitions of appearance. This idea, however, is only applicable if the entire sequence of images to be encoded in advance. Possible applications may be efficient encoding of movies or recorded video messages to be transmitted. We call the latter video electronic mail or, short, V-Mail. Principal components analysis can then yield an efficient representation of an image sequence in a fraction of its basis dimensions.

7.2.2 Reducing the Dimensionality of the Appearance Manifold

Consider a head and shoulders sequence like the one in figure 7.3. In order to demonstrate the impact of normalizing of the sequence to the talking head, let us first cut out a region of interest in the center of the image. Taking ten out of the 103 images of the sequence and computing the eigenvalues of those ten images and ordering them according to their size, we get the eigenvalue distribution in figure 7.6 for the uncentered sequence (upper curve). Graph 7.6 b) uses a linear scale instead of a logarithmic in 7.6 a). Repeating the same exercise for fifteen instead of ten images yields the curves for the uncentered sequence in figure 7.7 a) and b). The input images used to compute the eigenvalues are shown in figures 7.9. The corresponding eigenvectors as well as the average image are depicted in figure 7.9.

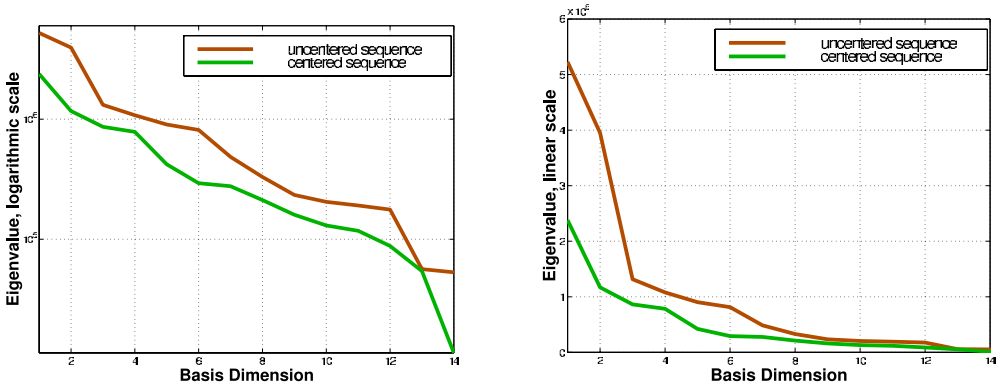
However, if we use the tracker described in the previous chapter to cut out a region of interest around the face, we get the sequence in figure 7.10. We then compute the eigenvalues, shown in figure 7.7 a) and b), centered sequence, and the eigenvectors, shown in figure 7.11.

Comparing the curves for the centered and uncentered sequences in figures 7.6 a) and b), and in figures 7.7 a) and b) respectively, we see that the centered sequence generates smaller eigenvalues. The error made by choosing a representation with less than the total number of eigenvalues equals the integral (sum) over the remaining eigenvalues. One of the convenient side-effects of using an eigenspace representation instead of a fixed basis such as a series of cosine functions is that we know exactly the error we make by omitting some basis dimensions in our representation. This result is also valid if we build an orthonormal basis representation of the input data other than the eigenspace, for instance, by using the Gram-Schmidt procedure.



a) b)

FIG. *7.6: Eigenvalues for a 10 image basis in descending order for an uncentered and a centered sequence, a) logarithmic scale, b) linear scale.*



a) b)

FIG. *7.7: Eigenvalues for a 15 image basis in descending order for an uncentered and a centered sequence, a) logarithmic scale, b) linear scale.*

The denser representation of our input sequence in eigenspace, illustrated already in figures 7.4 and 7.5 directly translates into two optional benefits : Either the possibility of choosing a more compact representation by omitting some of the basis dimensions while making the same error than for an uncentered image. Or an improved reconstruction quality, i.e., a reduced error with the

FIG. *7.8: 15 images cut out of a sequence of a typical one-to-one video communication scene of 103 images, image format : 54 x 84 pixels ; sequence not centered by the face*

FIG. *7.9: Basis images created out of the images in figure 7.8.*

FIG. *7.10: 15 images out of a sequence of a typical one-to-one video communication scene of 103 images, image format : 54 x 84 pixels . Sequence centered with the tracking algorithm from the last chapter.*

FIG. *7.11: Basis images created out of the images in figure 7.10.*

same number of basis dimensions.

7.3 The sample selection problem

Using eigenspace techniques such as in orthonormal basis coding, we are in front of a dilemma, which can be formulated in the following two problems :

a. calculating the eigenspace representation of an entire sequence of, say, 100 or more images becomes computationally extremely costly. The most computing power consuming part is the generation of the (auto-)covariance matrix. Even for an offline process such as the compression of a movie video or a pre-recorded video (e-)mail, computing time would quickly become unacceptable for users.

b. In order to compute a real eigenspace representation, all the data has to be known in advance ; i.e., sequential online processing (streaming) would be implicitely impossible.

Although this makes a true eigenspace representation unpractical, we propose a compromise, which is based on the observation that a principal components analysis (PCA) is looking for the most *typical* data content. This data content is the eigenvector of the largest eigenvalue. Just as with a common Fourier transform, eigenvectors are of varying importance for reconstruction, comparable to lower and higher frequencies of the FT. Taking advantage of this we can approximate a data analysis such as PCA in a computationally much less costly way. The idea is to divide the set of input data (video stream) into clusters of similar samples (images). This is equivalent to a vector quantization. Of course, it is not by chance that the same algorithms (e.g., K-Means) are used in clustering for pattern recognition as well as in vector quantization for, e.g., sound compression.

We claim that with this combined approach, clustering and eigenspace coding, we can do better for a normalized video stream than with pre-selected bases such as a series of cosine functions (DCT), even though we might fall short of the performance of a true eigenspace representation. This section presents and analyzes several selection algorithms and study their properties and performance for our purposes.

The sample selection problem is thus basically a quantization or clustering problem such as discussed in section 4.4. When we talk about *samples*, we are thinking in the context of this work of images understood as vectors (see 7.2). The abbreviation *OP* indicates the one-pass version of the considered algorithm. One-pass algorithms are potentially appropriate for online coding (streaming).

7.3.1 EquiDistant Algorithm

This is a straightforward selection algorithm. All it does is calculate the temporal distance between two images on the time axis for a certain number, i.e., the desired number of basis dimensions, of images. There is no selection criteria other than the desired maximum number of images and its equal distribution over the time axis. As simple as it may seem, this algorithm is surprisingly efficient. In this simple form it is fast and has only one pass.

Algorithm EQUIDISTANTOP(S, K)
Input: A set, S, of N samples x_n ; a number, K, indicating the amount of desired clusters.
Output: A set, S_K, of K samples with cluster centers z_k.
1. Compute the (temporal) distance $\Delta = \left[\frac{n}{K}\right]$
2. **for** $k \leftarrow 1$ **to** K
3. **do** $z_k = x_{\Delta \cdot k}$
4. **return** S_K

The performance of the EQUIDISTANT Algorithm w.r.t. reconstruction quality (not computing time !) can significantly be improved if we take the selected samples as cluster centers, assign all samples to a cluster, and calculate new cluster centers using the Euclidean distance metric. The complexity of the one-pass EQUIDISTANT algorithm (EQUIDISTANTOP) is $O(K)$, and of the enhanced EQUIDISTANT algorithm (EQUIDISTANT) is $O(n)$.

Algorithm EQUIDISTANT(S, K)
Input: A set, S, of N samples x_n ; a number, K, indicating the amount of desired clusters.
Output: A set, S_K, of K samples with cluster centers z_k.
1. $S_K \leftarrow$ EQUIDISTANTOP(S, K)
2. Assign each sample to a cluster, such that the Euclidean distance $d_E = ||x_n - z_k||^2$ is minimized for all $1 \leq n \leq N$ and $1 \leq k \leq K$. ($*$ It is clear that $< x_n, z_k >= \min$ for $n = k$, so we can actually skip that multiplication. $*$)
3. **for** $k \leftarrow 1$ **to** K
4. Compute new cluster centers :
$$z'_k = \frac{1}{N_k} \sum_{x \in S_k} x$$
5. ($*$ where N_k is the number of samples, x, in the kth cluster $*$)

7.3.2 Threshold Algorithm

The threshold method assumes that similar frames are likely to be located sequentially in the video sequence. This is not necessarily the case when each image contains only a face talking. The threshold method has a complexity of $O(n)$ and works as follows.

The Euclidean distance is computed between image one and subsequent images until it drops below a certain threshold. At that point in the sequence, the current image becomes a new cluster center. For subsequent images only the distance to the new cluster center is computed until the threshold is crossed again.

Algorithm THRESHOLDOP(S, T)
Input: A set, S, of N samples x_n ; a threshold, T , indicating the maximum allowed (Euclidean) distance of samples in a cluster.
Output: A set, S_K, of K clusters of samples with cluster centers z_k.
($*$ K is not known in advance. $*$)

1. Set $k = 1$, $z_1 = x_1$
2. **for** $i \leftarrow 2$ **to** N
3. **do if** $T < ||x_n - z_k||^2$
4. **then** $S_k \leftarrow x_n$
5. **else** $k+ = 1$
6. $z_k \leftarrow x_n$
7. **return** S_K

Usually, we want to fix the desired basis size, B, in advance. That is, it is possible that our chosen threshold, T, is so low that THRESHOLDOP(x_n, T) produces more clusters than the desired number of basis images, B. This is where the sorting step at the end of THRESHOLD(x_n, T) becomes important. Averaging the samples in a cluster to calculate the new cluster center, z_k, and choosing the first B out of K clusters makes sure that a maximum number of samples contribute to the basis.

Algorithm THRESHOLD(S, T)
Input: A set, S, of N samples x_n; a threshold, T, indicating the maximum allowed (Euclidean) distance of samples in a cluster.
Output: A set, S_K, of K clusters of samples with cluster centers z_k.
($*$ K is not known in advance. $*$)
1. $S_K \leftarrow$ THRESHOLDOP(S, T)
2. **for** $k \leftarrow 1$ **to** K
3. Compute new cluster centers :

$$z'_k = \frac{1}{N_k} \sum_{x \in S_k} x$$

4. ($*$ where N_k is the number of samples, x, in the kth cluster $*$)
5. Sort clusters, S_K by the number of samples they contain
6. **return** S_K

In a real application, it is further possible that $B > K$, i.e., T was chosen too high. For that case, we can re-run THRESHOLD(x_n, T), THRESHOLDOP(x_n, T), or with a lower T; in an offline application this is not too inconvenient. In an online application this could compromise the reconstruction quality since it reduces the order of the encoding basis.

7.3.3 MostRepresentative Algorithm

An algorithm, called MOSTREPRESENTATIVE ALGORITHM, first reported in [VSC99, CS99], attempts to find similar images anywhere in the sequence to be encoded.

In order to do this, we take the first image of the sequence as the first cluster center. Then compare it to all the other images in the sequence, and assign any image to that cluster of which the (Euclidean) distance to the cluster center falls below a chosen threshold, T. Any other images are put into a to do set of unassigned images, S_U.

We then take the next image out of the to do set, S_U, and save it as our next cluster center. We then assign all images close enough to that cluster center into that new cluster.

This procedure continues until there are no images left in S_U, and all similar images are grouped in sets.

We then have the choice of either take an arbitrary image out of each cluster to form the basis, or compute the real new cluster center, i.e., the mean of all samples in the cluster. According to our experience the latter gives the better results in image quality, taking only a negligible extra computing time into account.

Algorithm MOSTREPRESENTATIVE(S, T)
Input: A set, S_N, of N samples, x_n; a threshold, T, indicating the maximum allowed (Euclidean) distance of samples in a cluster.
Output: A set, S_K, of K clusters of samples with cluster centers z_k.
($*$ K is not known in advance. $*$)
1. Put all samples x_n of S_N into a set S_U of un-assigned samples
2. Set $k = 0$
3. **repeat**
4. Take a sample, x_u, out of S_U
5. Increase k by 1, and
6. Set $z_k = x_u$
7. **repeat**
8. Take another sample, x_u, out of S_U
9. **if** $||x_u - z_k||^2 < T$
10. **then** $S_k \leftarrow x_i$
11. **else** Put x_u back into S_U
12. **until** z_k has been compared to all remaining samples in S_U
13. **until** there are no images left in S_U
14. **for** $k \leftarrow 1$ **to** K
15. Compute new cluster centers :

$$z'_k = \frac{1}{N_k} \sum_{x \in S_k} x$$

16. ($*$ where N_k is the number of samples, x, in the kth cluster $*$)
17. Sort clusters, S_K by the number of samples they contain
18. **return** S_K

The most-representative sample selection method has a best case complexity of $O(n)$ and a worst case of $O(n^2)$ although neither are very likely.

In real situations, the resulting amount of clusters is rarely equal to the number of desired basis dimensions. If we got more clusters than desired basis dimensions, B, we just take the B largest sets of clusters. This is where the sorting step in line 17 comes in. If $B > K$, it makes sense to re-run MOSTREPRESENTATIVE(S, T), with a lower threshold, e.g., MOSTREPRESENTATIVE($S, 0.9T$), until $B \leq K$.

7.3.4 MaxminDistance Algorithm

The MAXMINDISTANCE Algorithm [TG74] is a simple heuristic procedure, based on the Euclidean distance concept, to divide a set of N samples into reasonable clusters. The only input parameter is a relative threshold, T_{rel},

The algorithm selects some of the input samples as cluster centers. As an option, we can calculate the mean of all samples within a cluster and take this as the cluster center.

Algorithm MAXMINDISTANCE(S, T_{rel})
Input: A set, S, of N samples x_n ; a (relative) threshold, T_{rel}, indicating the minimum allowed (Euclidean) distance of two cluster centers, relative to the largest distance between two clusters
Output: A set, S_K, of K clusters with cluster centers z_k.
($*$ K is not known *a priori*. $*$)
1. Determine a first (arbitrary) cluster center z_1 ; e.g., $z_1 = x_1$
2. Set K to 1
3. **repeat**
4. **for** $n \leftarrow 2$ **to** N
5. **for** $k \leftarrow 1$ **to** K
6. Compute (Euclidean) distances : $d_(n,k) = d(x_n, z_k)$
7. Determine $d_{n,k}^{min} = \min_k(d_(n,k))$
8. Determine $d_{n,k}^{maxmin} = \max_n(d_{n,k}^{min})$
9. **if** $d_{n,k}^{maxmin} \geq T_{rel} \cdot d_{n,k,old}^{maxmin}$
10. **then** Increase K by 1
11. $z_K = x_{n,max}$
12. **until** $d_{n,k}^{maxmin} < T_{rel} \cdot d_{n,k,old}^{maxmin}$
13. **for** $k \leftarrow 1$ **to** K
14. Compute new cluster centers :

$$z'_k = \frac{1}{N_k} \sum_{x \in S_k} x$$

15. ($*$ where N_k is the number of samples, x, in the kth cluster $*$)
16. Sort clusters, S_K by the number of samples they contain
17. **return** S_K

The MAXMINDISTANCE(S, T_{rel}) proved to be efficient especially for a large number of cluster centers (see figure 7.14).

7.3.5 K-Means Algorithm

The K-MEANS algorithm [TG74, Lim90] is one of the classical clustering algorithms. It is still of interest today, not the least because of its application in vector quantization (for sound compression, for instance [Say00]). The K-means algorithm was described by Forgy [For82]. The algorithm is also called LBG-algorithm, since Linde, Buzo, and Gray [LBG80] have shown that

the algorithm can be used with many different distance measures and have popularized its use to applications of vector quantization to speech and image coding.

Although the choice of the initial K cluster centers is free, it is a good idea to use the EQUIDISTANTOP algorithm. This ensures that the (initial) cluster centers are well distributed over the time axis. This may be important to catch some movements otherwise not covered by using the first K samples as initial cluster centers.

K-MEANS is an iterative algorithm, i.e., in its classical form it is not appropriate for online coding. Since the number of samples is known in advance and thus limited, K-MEANS is guaranteed to converge within a few iterating steps. In our experiments, about four or five iterating steps were necessary for about 100 samples, independently of the desired number of basis images ($B = K$).

Algorithm K-MEANS(x_n, K)
Input: A set of n samples x_n ; a number, K, indicating the amount of desired clusters.
Output: A set, S_K, of K clusters with cluster centers z_k.
1. $i = 1$, and $S_K(i) \leftarrow$ EQUIDISTANTOP(x_n, K)
($*$ This is the only step differing from "classical" K-Means $*$)
2. **while** $z_k(i+1) \neq z_k(i)\ \forall\, k$
($*$ where i indicates the ith iteration step $*$)
3. Distribute the samples among the K cluster domains, using $x \in S_K$ if $||x - z_k|| \leq ||x - z_j||\ \forall\, k, j = 1, 2, ..., K, j \neq k$
4. **for** $k \leftarrow 1$ **to** K
5. Compute new cluster centers :

$$z_k(i+1) = \frac{1}{N_{S_K}} \sum_{x \in S_k} x$$

K-MEANS yields a set S_K of K clusters, where the performance index

$$J_k = \sum_{x \in S_K} ||x - z_k(i)||^2,\ k = 1, 2, ..., K \tag{7.1}$$

is minimized.

7.3.6 Comparison of algorithms

All algorithms could be made sensitive to movements of the eyes and mouth by multiplying these regions by an extra weighting factor during the comparison of images with the to-do set. Thus variations in eye and mouth configurations receive a better representation in the selected sample set. If we consider the results of table 7.1 and figures 7.12, 7.13, and 7.14, we see that the EQUIDISTANT algorithm should in most cases do.

TAB. *7.1: Properties of comparative clustering algorithms; for the algorithms with complexity* $O(n) \leq C \leq O(n^2)$*, the complexity is in most cases closer to* $O(n)$ *than to* $O(n^2)$*.*

Algorithm	One-pass	Complexity	Iterative
EQUIDISTANTOP	$\surd$	$O(n)$	$\times$
EQUIDISTANT	$\surd$	$O(n)$	$\times$
THRESHOLDOP	$\surd$	$O(n)$	$\times$
THRESHOLD	$\surd$	$O(n)$	$\times$
MOSTREPRESENTATIVE	$\times$	$O(n) \leq C \leq O(n^2)$	$\times$
MAXMIN-DISTANCE	$\times$	$O(n) \leq C \leq O(n^2)$	$\times$
K-MEANS	$\times$	$O(n) \leq C \leq O(n^2)$	$\surd$

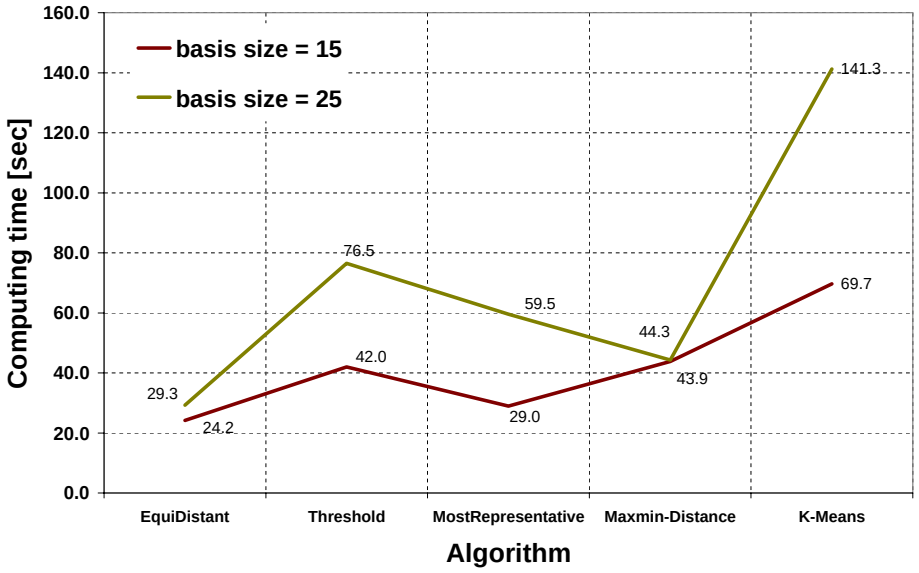


FIG. *7.12:* TALKINGHEAD *sequence : Computing time in sec of the different clustering algorithms for various basis sizes.*

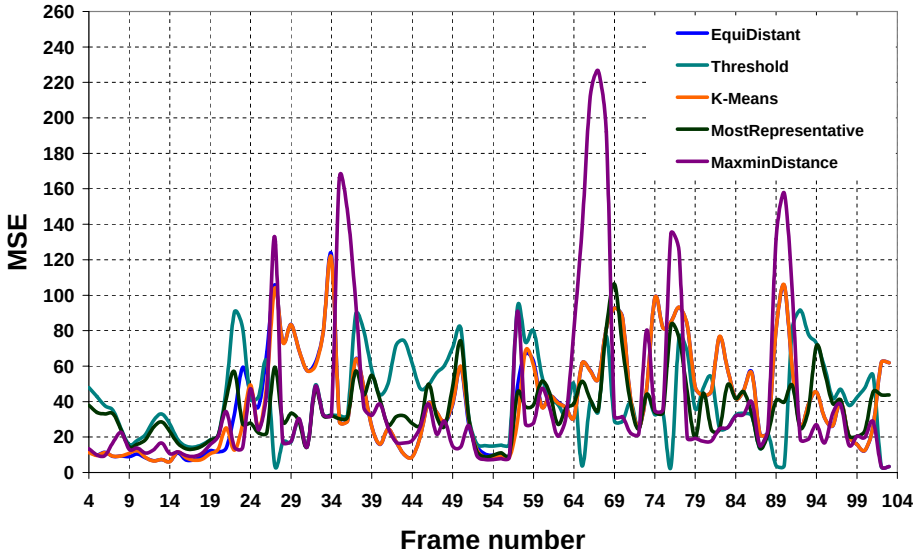


FIG. *7.13:* TALKINGHEAD *sequence : MSE for 25 basis images, quality parameter 100, and different clustering algorithms. The* K-MEANS *(average MSE = 43.74) algorithm starts out with the same frames (equally distributed over the time axis) than the* EQUIDISTANT *(average MSE = 44.73) algorithm, iterating to a stable distribution of frames to the clusters. We see that the resulting MSE for the reconstructed frames are very close. See table 7.2 for the average MSE and PSNR of all five algorithms.*

TAB. *7.2: Average MSE and PSNR [in dB] of the curves in figure 7.13.*

Algorithm	MSE	PSNR
EQUIDISTANT	44.73	31.27
THRESHOLD	41.62	31.23
MOSTREPRESENTATIVE	41.83	31.24
MAXMIN-DISTANCE	48.06	31.87
K-MEANS	43.74	31.37

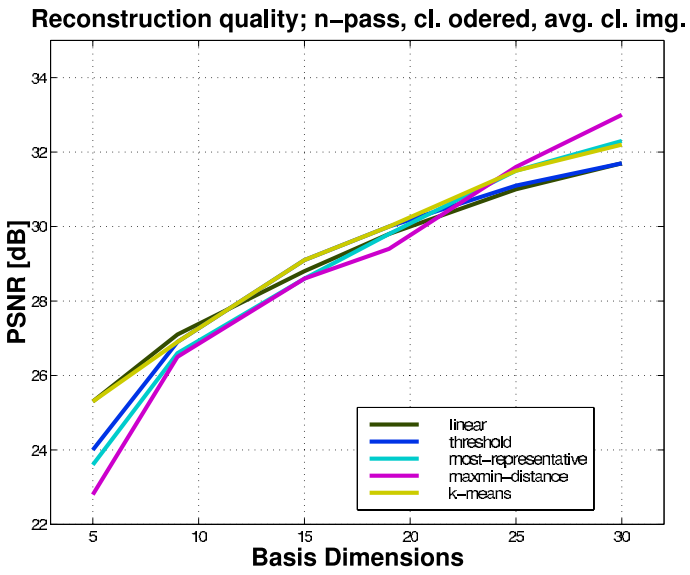


FIG. *7.14: Average PSNR for the* TALKINGHEAD *sequence*

7.4 Performance Evaluation

After having determined the ingredients of OBC, we want to know how good its works. This section is about evaluation criteria for the encoding process which can directly be applied as design criteria for the video codec ; they also are the criteria used in the following chapter to compare OBC to conventional video compression methods. We identify the following evaluation metrics :

Compression ratio : The compression ratio as an evaluation criterion is closely related to the reconstruction quality. A trade-off between the two determines if an algorithm is appropriate for an application or not.

Reconstruction quality : While the two criteria above produce objective numbers easy to verify, measuring reconstruction quality is a more difficult task. In the following section we introduce commonly used reconstruction quality measures. Note that the most important criterion for measuring reconstruction quality is human perception. Finding error metrics modeling human visual perception would be a thesis topic by itself. We can only point out here the importance of such a work and explain why the error metrics introduced hereafter fail to fulfill this task in a satisfactory way.

Computing speed : The algorithms discussed above are of varying complexity. Eigenspace techniques have been increasingly used the last couple of years, particularly in computer vision for localization, recognition, indexing, etc. This is made possible by increasingly powerful computers. Computing speed is also an important criterion whether an algorithm is appropriate for online, i.e., real-time coding.

Reconstruction quality w.r.t. compression ratio is *the* crucial performance metric for every image and video codec. The problem we are concerned with here is that coding artifacts of OBC are substantially different from those of conventional, block based coding schemes. Using another orthonormal basis is thereby less important than encoding images as a whole instead of in 8 x 8 pixel blocks. The problem is such that, as we shall see below, commonly used quality metrics fail.

7.4.1 Reconstruction quality measures

We call the error measures for reconstruction discussed in this section *objective*, because they measure the error of a reconstructed (decoded) signal, i.e., the signal estimate, to the original input signal, regardless of perception issues. We will define here four quality measures being used for measuring image reconstruction quality.

Definition 2 (Mean Squared Error) *The* Mean Squared Error (MSE) *of an estimate $\hat{x}$ of a signal x is the function of x defined by*

$$MSE(x) \;=\; E(\hat{x}-x)^2 \;=\; \sigma_{\hat{x}}^2 + (E\hat{x}-x)^2, \tag{7.2}$$

where $(E\hat{x}-x)^2$ is the bias *of the estimator $\hat{x}$, and*

$$\sigma_{\hat{x}}^2 \;=\; E(\hat{x}-E\hat{x})^2 \;=\; E\hat{x}^2 - (E\hat{x})^2, \tag{7.3}$$

its variance. If $E\hat{x} = x$, that is, if the expected value (mean) of the signal is equal to the signal itself, then the MSE(x) is unbiased and equal to the variance of $\hat{x}$. This is a reasonable assumption which should be met in most cases that concern us. Otherwise we would have to review the estimation process.

For the vector case, i.e., an estimate $\hat{\vec{x}}$ of a signal $\vec{x}$, both with N dimensions, and using inner product notation,

$$MSE(\vec{x}) \;=\; \frac{1}{N} < \hat{\vec{x}} - \vec{x}\,,\, \hat{\vec{x}} - \vec{x} > \;=\; \frac{1}{N}\,||\hat{\vec{x}} - \vec{x}||^2 \tag{7.4}$$

for and unbiased $\hat{\vec{x}}$, which is generally the case in our context. The MSE is also called the average distortion. In the context of coding and reconstruction of images, $\vec{x}$ is the original input image, and $\hat{\vec{x}}$ the reconstructed image.

Definition 3 (Root Mean Squared Error) *The* Root Mean Squared Error (RMS) *is the positive square root of the Mean Squared Error (MSE). It is thus*

$$RMS(x) \;=\; \sqrt{MSE(x)} \tag{7.5}$$

and accordingly

$$RMS(\vec{x}) \;=\; \sqrt{MSE(\vec{x})} \;=\; \sqrt{< \hat{\vec{x}} - \vec{x}, \hat{\vec{x}} - \vec{x} >} \;=\; ||\hat{\vec{x}} - \vec{x}|| \tag{7.6}$$

for vectors. For the case of images, the RMS can be interpreted as the average error per pixel.

Mean Squared Error and Root Mean Squared Error give the error as an absolute value, i.e., the squared difference (or its root) of a signal and its estimation. This is a *quantitative* measure saying nothing (or not much, unless the signal's unit range is known) about the *quality* of a reconstruction. Therefore, two error measures have been defined relating the error to the *properties* of the original signal. It is convenient to use the logarithm of the resulting error value.

Definition 4 (Signal-to-Noise Ratio) *The* Signal-to-Noise Ratio (SNR) *of an estimate $\hat{\vec{x}}$ for a signal $\vec{x}$ is defined as*

$$SNR(\vec{x}) \;=\; 10 \cdot \log_{10}\left[\frac{\sigma_x^2}{MSE(\vec{x})}\right] \;=\; 20 \cdot \log_{10}\left[\frac{\sigma_x}{RMS(\vec{x})}\right] \tag{7.7}$$

where σ_x^2 is the variance of the signal $\vec{x}$:

$$\sigma_{\vec{x}}^2 \;=\; E(\vec{x} - E\vec{x})^2 \;=\; E\vec{x}^2 - (E\vec{x})^2. \tag{7.8}$$

The unit of the SNR is decibels [dB].

The SNR relates the reconstruction error to variance of the (local) signal amplitude, which is particularly important in sound coding. In image coding, using the variance of the signal amplitude

does not say much. The signal amplitude corresponds here to a gray-scale or color, and all possible colors are generally equally desirable. A common quantity to relate the reconstruction error to is the *amplitude range* of the signal, which for image processing translates to the maximum (peak) signal value. The corresponding metric for measuring the image reconstruction quality is called the

Definition 5 (Peak Signal-to-Noise Ratio) *The* Peak Signal-to-Noise Ratio (PSNR) *of an estimate $\hat{\vec{x}}$ for a signal $\vec{x}$ is defined as*

$$PSNR(\vec{x}) \;=\; 10\cdot\log_{10}\left[\frac{(\max_N(\vec{x}))^2}{MSE(\vec{x})}\right]\;[dB] \quad = 20\cdot\log_{10}\left[\frac{\max_N(\vec{x})}{RMS(\vec{x})}\right]\;[dB], \tag{7.9}$$

where $\max_N(\vec{x})$ *is the maximum (peak) over all N values of $\vec{x}$. Again, the unit is decibels [dB]. A bigger PSNR represents a better reconstruction.*

All those error measures are defined mathematically as the deviation of an estimate to the original signal. *There is no physical interpretation to them.* This becomes particularly problematic when we talk about human perception. A human supposed to judge the quality of an image will look for information, i.e., objects. Random noise has no information as we saw in section 4.1. Image content means information, means coherently colored areas with more or less sharp edges. A human will then try to identify the objects he sees. He will do this trying to map the edges and areas he sees to concepts of objects he has in his memory. Moreover, the human supposed to judge image quality does generally not know the original image. He is thus unlikely to use it as a reference.

This is only vague discussion about perception and image quality. What becomes immediately clear, though, is that none of the error measures above reflects in the least human perception. The only image degradation *really* measured by those measures is random noise. Let us consider a simple example to show how deceptive a PSNR can be.

Example 3 *Given a QCIF gray-scale image with 176 x 144 pixels and a maximum pixel value of 255. It is easy to show that a) 2601 pixels with a deviation of 5 produce the same MSE (= 2.57) and thus PSNR (= 44.04 dB) than b) 1 (!) single pixel with a deviation of 255 (for instance by an overflow).*

Case b) would mean a black instead of a white pixel or vice versa. This can be considered as a minor artifact, which may barely be perceptible (depending on its position), and which can be smoothed out by filtering. In Case a), however, 10.3% of the image are degraded, which can considerably compromise image quality and can certainly not as easily been smoothed out by filtering as case b).

What *should* be the starting point for developing image quality measures, though, is human perception. This is a vast research area and differences from one person to the next may be considerable, depending on cultural background, age, personal experience, and so on. As has been said above, this goes beyond the scope of this work, which nevertheless hopes to stimulate efforts in that direction.

7.5 Conclusion

This chapter finally introduced the prerequisites and techniques employed for OBC. The three-step encoding scheme is based on the classification introduced in chapter 4. OBC is applicable in any head-and-shoulders scenario. With some further research, it may even be interesting to be used beyond that. OBC is, as model-based content extraction methods, based on the paradigm that second generation video coding involves image content extraction. MPEG-4 and MPEG-7 point in that direction, however they offer the means of *how to compress* extracted information rather than *how to extract* that information. OBC goes a different way by exploiting only the *appearance* of an object in a scene to be encoded.

OBC is designed as an approximation of a true PCA, using vector quantization or clustering as a means to increase coding efficiency. The major difference to classical DCT/DPCM or wavelet based approaches and OBC, however, is the way the data, i.e., the video stream, is being looked at. Instead of considering an image as a matrix, OBC treats images as vectors. That opens up new possibilities for energy compaction and entropy reduction, notably the use of vector quantization or clustering techniques for sample selection. We saw that a relatively simple sample selection (clustering) algorithm is sufficient for a pre-selection of the basis input images.

Since data is being treated fundamentally differently in OBC than in conventional approaches, the resulting artifacts are of a different nature. This puts the focus on reconstruction quality measures. We discussed today's most important ones and noted that they have the fundamental drawback that they are *not* derived from models of human perception. This is not a new insight. In fact, it is a well known problem. It becomes particularly inconvenient, though, when we try to compare the results of fundamentally different compression algorithms such as OBC and MPEG, as we do in the following chapter.

Chapitre 8

Experiments and Results

This chapter presents the performance of the OBC codec in various configurations. The results are compared with those of a DCT/DPCM type MPEG codec. We concentrate on compression performance and reconstruction quality, although computing speed is considered, too. The following video sequences are used for evaluation :

TAB. *8.1: Video sequences used for performance evaluation*

Sequence	Frequency	# of Frames	Width x Height	Duration	1st Image
BILLS2	6 Hz	348	80 x 96	58 sec.	
CARPHONE	30 Hz	382	176 x 144	12.73 sec.	
CLAIRE	30 Hz	494	176 x 144	16.47 sec.	
GRANDMA	30 Hz	870	176 x 144	29 sec.	
JEANBAPTISTE	10 Hz	159	130 x 96	15.9 sec.	
MOM	30 Hz	150	176 x 144	5 sec.	
TALKINGHEAD	8.5 Hz	103	176 x 144	12.12 sec.	

All of these sequences show a head-and-shoulders scene, which is the scenario OBC was designed for. BILLS2 was one of the first sequences for which results of OBC coding were reported. The face tracker use to record this sequence used connected components and Kalman filtering and exhibits here still some instability. JEANBAPTISTE and TALKINGHEAD are two more examples of the head-and-shoulders scenario and were originally recorded as test sequences for the face tracker. Sequences CARPHONE, CLAIRE, GRANDMA, and MOM are frequently used by the video processing

community. They can be found at [Vid01b].

Section 8.1 presents `mpeg_encode`, the program we use as a benchmark, and the parameters it takes. The actual encoding procedure and the resulting performance are discussed in section 8.2. Finally, section 8.3 gives a critical evaluation of the results obtained, discussing shortcomings of the OBC codec as it works today, and giving pointers on further research to overcome these shortcomings. Throughout this chapter, curves in different shades of red indicate results of MPEG encoding, and OBC results are in different shades of blue.

8.1 The Benchmark : `mpeg_encode`

There are some freeware programs doing DCT/DPCM based coding. A popular one which has been around for a while is `mpeg_encode`, a MPEG-1 encoding program from the *Berkeley Multimedia Research Center* [Ber97], which is approved by the MPEG group. It is freeware, and comes along with some other programs to replay or analyze a MPEG-encoded stream. All those tools have no audio facility, but that was no problem since we focus on the video encoding performance. Employing all of its features including motion detection and compensation, `mpeg_encode`is a fully functional MPEG-encoding software. It is therefore a valid reference for any video codec.

There may be the question of why not use another software for benchmarking OBC. Any encoding program based on DCT/DPCM, for H.26x as well as MPEG-x encoding, uses basically the same encoding scheme. That is, they are trading off quality for compression in the same way, even if they differ in detail. MPEG uses forward and backward prediction and imposes the frequent use of INTRA frame encoding, whereas H.26x has no such constraints, but both allow the use of motion compensation algorithms. The result is that H.26x can yield higher compression rates... at the cost of reconstruction quality. An MPEG encoder is configurable to output compression rates comparable to H.26x, but only to lower the quality to the same level. Thus, it does not matter which DCT/DPCM encoding software we use. Finally, MPEG is more popular than H.26x, there is more software handling MPEG streams.

Encoding with `mpeg_encode`is implicitely offline, since backward prediction is being used to enable reverse playback. The program `mpeg_encode`uses the DCT/DPCM based encoding scheme discussed in chapter 5. Frames are treated as *INTRA frames*, hereafter short *I-frames*, predicted frames, *P-frames*, and interpolated frames, *B-frames*. Frames are encoded not exactly sequentially, but I-frames are encoded first, then P-frames are predicted, and finally B-frames interpolated. This has no impact on the compression performance.

The configuration of `mpeg_encode`is done with command line options and a parameter file containing the input parameters for the encoding process. The following is a list of input parameters for the `mpeg_encode`program :

Q-Scale : There are three different quantizer scales, `IQ-SCALE`, `PQ-SCALE`, and `BQ-SCALE`, for INTRA frames, predicted frames and interpolated frames. These have a direct impact on the compression ratio and the reconstruction quality. The coarser the quantizer scale, the higher the compression ratio, but the worse the reconstruction quality.

GOP : The Group of Pictures parameter, indicated by `GOP_SIZE`, indicates an independently decodable sequence of frames. This is important for devices or programs using forward and reverse playback and search. `GOP_SIZE` actually denotes the *minimum* number of frames in a Group of Pictures. This parameter has no impact neither on quality nor on compression.

IPB Pattern : indicates the desired sequence of I-, P-, and B-frames. One possible example for the parameter `PATTERN` is `IBBBPBBBI`, although this sequence can be arbitrarily chosen. Is has to start with an I-frame. This parameter has a direct impact on quality and compression (and computing speed), since B-frames compress best, and compression of P-frames is between I-frames and P-frames. Higher compression, of course, means lower quality.

Slice : `SLICES_PER_FRAME` divides a frame in independently decodable units. A slice represents a transmission unit, i.e., more slices are recommended for noisier channels. For our purposes where the channel is a file on harddisk, `SLICES_PER_FRAME` is set to one.

Motion vector search : P- and B-frames are searched for motion vectors. For P-frames, there are four algorithms which can be selected, and the parameter `PSEARCH_ALG` can the corresponding values `LOGARITHMIC`, `SUBSAMPLE`, `TWOLEVEL`, or `EXHAUSTIVE`. The parameter `LOGARITHMIC` for B-frames can be `EXHAUSTIVE`, `CROSS2`, `SIMPLE`. Please see [Ber97] and [GR94] for more details. All these parameters determining the desired motion vector search algorithm have little impact on the reconstruction quality, but can improve compression at the cost of computing time.

Search Window : This is a square area where motion vectors are searched for. The parameter `PIXEL` can have the values `FULL` or `HALF`, depending on if half-pixel vectors are supposed to be allowed or not. The second parameter `RANGE` indicates simply the size of the search window in pixels. The type and size of the search window influence quality and compression.

There are some other options, but they have no impact on the compression results. They include things like computing the reconstruction error an saving it into a statistics file. Table 8.2 contains typical values used for encoding with `mpeg_encode`. In order to vary the output compression and image quality, only the quantization parameters were modified in general, if not otherwise indicated.

8.2 Encoding procedure and performance

The sequences from table 8.1 were compressed and reconstructed with varying parameters. The encoding scheme follows the three step process from page 118. For the first encoding step, the input sequences from table 8.1 were fed into the face tracker, and an area of about 80 x 80 pixels was cut out around the detected face. This corresponds roughly to the resolution of a LCD color mobile phone display.

For the results below, the input images for OBC and MPEG are the same, i.e., MPEG is using the same normalized video sequences than OBC. This may cause the question if normalizing, which is prerequisite for efficient OBC coding, affects also the performance of an MPEG codec.

TAB. *8.2: Typical encoding parameters for the* `mpeg_encode` *program*

Encoding parameter	Typical values
IQ-SCALE	3
PQ-SCALE	6
BQ-SCALE	8
GOP_SIZE	10
PSEARCH_ALG	TWOLEVEL
BSEARCH_ALG	CROSS2
PATTERN	IBBBPBBBI
SLICES_PER_FRAME	1
PIXEL	HALF
RANGE	12

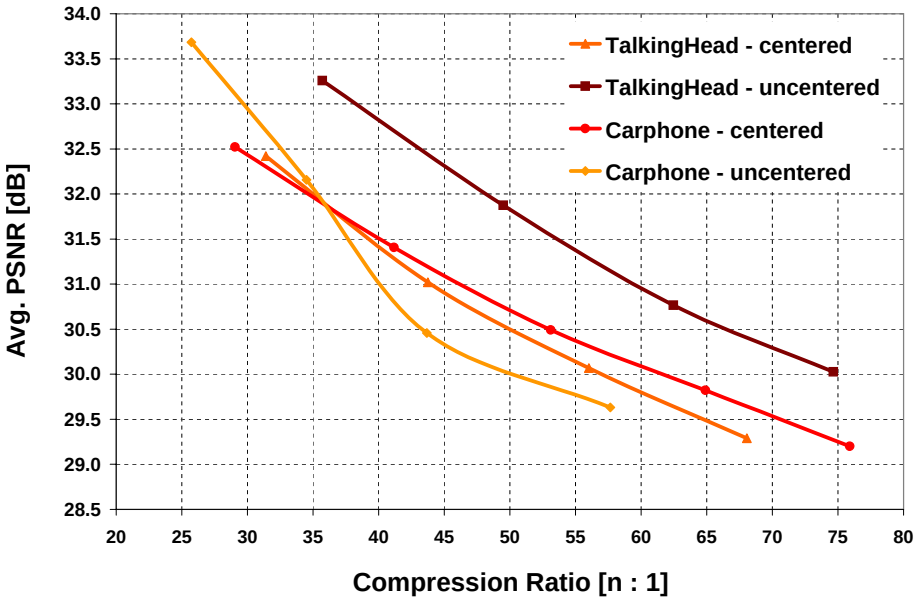


FIG. *8.1: MPEG encoding, sequence centered versus uncentered ;* TALKINGHEAD *sequence : differences due to more static background in the uncentered case, allowing the motion detection algorithm to be effective ;* CARPHONE *sequence : curves are arbitrary, since in both cases the background changes*

However, figure 8.1 shows that, as we would expect, normalizing has no impact on compression performance.

Unless otherwise stated, `mpeg_encode`uses the coding parameters from section 8.1. These parameters are sometimes varied as stated. The MPEG reconstruction errors were due to the typical blocking artifacts of DCT/DPCM codecs. These artifacts are caused by quantizing and consist usually in blurring within encoding blocks (8 x 8 pixels), and abrupt transition of pixel values (colors).

The OBC codec output consists basically of two parts : the basis and the encoding parameters for each image. The basis can be further compressed using ZIP or JPEG ; the encoding parameters of a single frame are one floating point number for each basis dimension. For example, each frame for a 15-basis-frame reconstruction adds 60 Bytes to the OBC codec output. For open universe, offline coding, this means that each additional frame needs only 60 Bytes to be reconstructed, *regardless of its size*. A serialization of the codec output using multiple resolutions of the basis is thinkable. Lack of time prevented experiments in that direction.

The typical reconstruction error of OBC is blurred reconstructed images. A reconstructed sequence with high error rate appears as if someone played with the focus of a camera lens. Depending on the denseness of the clusters, a gesture such as turning the head may appear like an animated transition between two images as is done in images synthesis. Also, some image features such as mouth opening or an eye blink may disappear if they are levelled out by averaging the samples in a cluster. All of these errors can be significantly reduced by precisely normalizing the input video stream and by choosing the basis size big enough to capture all essential motion in the image.

Let us point out that our results are *not* restricted to the face area, and that *everything* needed to decode a sequence – including the basis [!] – is included in the output stream.

8.2.1 Sequence BILLS2

The BILLS2 video clip is a sequence which was recorded online. It shows a typical head and shoulders scene with 80 x 96 pixels containing 348 frames and lasting 58 seconds (6 fps). We reported previously in a first publication [VSC99] that very high compression rates are possible at a reconstruction quality comparable to MPEG (at comparable compression). In fact, the BILLS2 sequence is not the best example, even if OBC comes close to MPEG performance. The reason is that the face is poorly centered compared to the other sequences used. The tracking was performed then by the "old" CCO/Kalman filter face tracker.

Figure 8.2 shows the average PSNR for the entire sequence versus the overall compression rate. The three OBC curves correspond to different encoding parameters, notably the basis size, i.e., the number of dimensions used for the basis. We see that for low compression rates (40-) OBC stays between 2 and 3 dB under the performance of MPEG. This is a deviation of about 10%. It is only for very high compression rates (100+) that OBC comes within 1 dB PSNR to MPEG. Table 8.3 show some of the measured values of figure 8.2 in numbers.

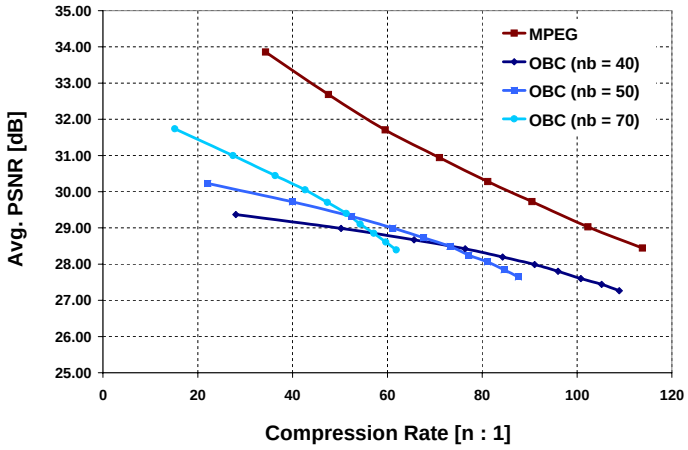


FIG. *8.2:* BILLS2 *sequence : Avg. PSNR in [dB] vs. Compression Rate in n : 1 for MPEG and OBC encoding in various configurations.* nb *indicates the number of basis images used.*

Figure 8.3 finally illustrates the artifacts of MPEG and OBC coding by displaying the frames that have been reconstructed with the highest and lowest PSNR, plus their originals. We can easily identify the blocking artifacts of DCT/DPCM-based MPEG compression. The OBC reconstruction error consists of a more or less pronounced fuzziness of the decoded frame.

Figures 8.4 and 8.5 compare OBC and MPEG compression for an increasing number of frames to be encoded. The basis size is adapted for different numbers of frames. As we would expect, the

TAB. *8.3: Sequence* BILLS2*, face region 80 x 80 pixels : Some encoding parameters and results from figure 8.2 in numbers*

OBC coding			
Basis size	Basis quality	Compression Rate [n : 1]	Avg. PSNR [dB]
40	55	108.90	27.27
40	70	96.00	27.80
40	80	84.32	28.19
40	90	65.61	28.67
40	100	28.02	29.37
MPEG coding			
	P : B : I quantizer scales	Compression Rate [n : 1]	Avg. PSNR [dB]
	24 : 22 : 19	113.81	28.45
	18 : 16 : 13	90.49	29.72
	16 : 14 : 11	81.19	30.28
	12 : 10 : 7	59.54	31.71
	8 : 6 : 3	34.32	33.86

OBC compression rate is increasing with the number of frames encoded. However, the reconstruction quality stays virtually always under that of MPEG compression.

The reason for OBC falling behind MPEG in terms of reconstruction quality for the BILLS2 sequence lies in the poor normalization of the input data. This increases significantly the variance of the input vectors (images) and reduces thus the efficiency of eigenspace-based encoding. However, the performance of OBC compression versus reconstruction quality is non-linear as can be seen in figure 8.2. The trade-off between compression and quality for MPEG is a relatively linear curve, even the motion estimation. With variation of basis size and basis compression, OBC has two powerful tools to further push-up compression if desired. An encouraging fact is that for very high compression rates, OBC reconstruction quality approaches that of MPEG. This is not (necessarily) the case for model-based coding [Eis00].

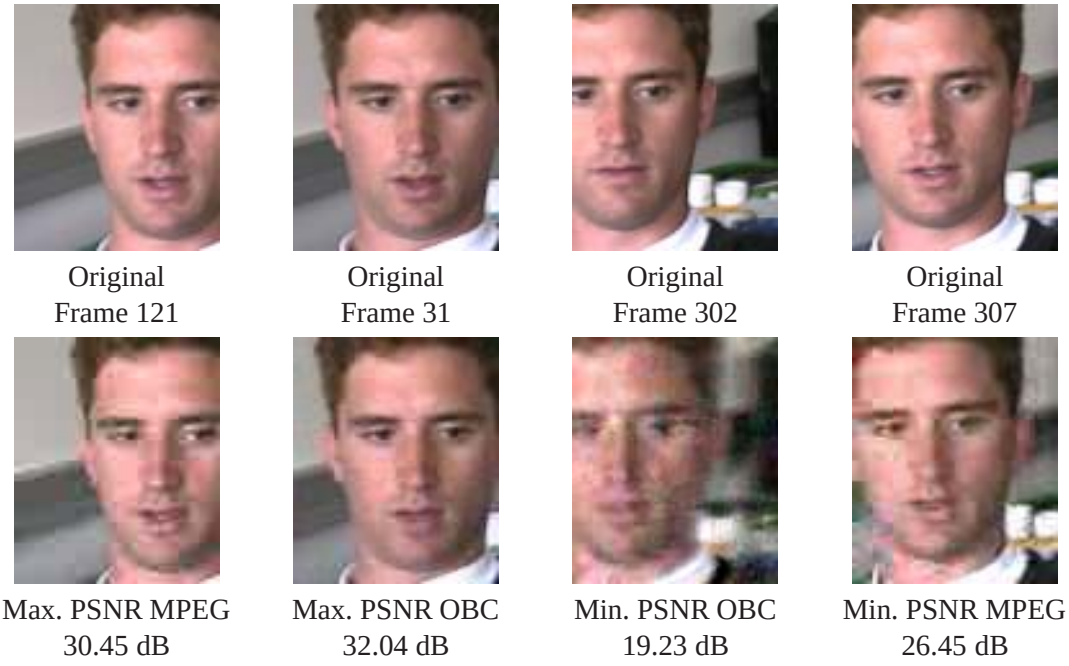


FIG. *8.3: Example frames from sequence* BILLS2 *showing the coding artifacts, illustrated by the best (maximum PSNR) and worst (minimum PSNR) reconstruction : MPEG encoding with P : B : I quantizer scales of 24 : 22 : 19 (avg. PSNR 28.45 ; CR 113.81 : 1), and for OBC with 40 basis images and quality parameter 55 (avg. PSNR 27.27 ; CR 108.90 : 1).*

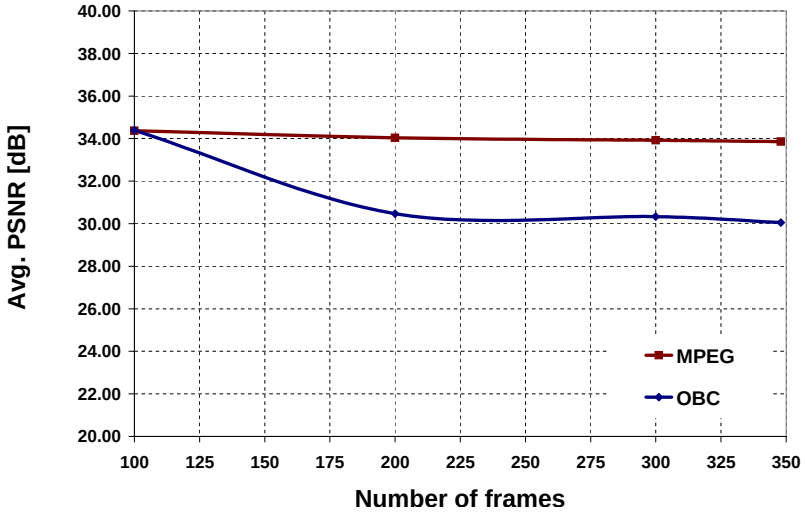


FIG. *8.4:* BILLS2 *sequence : Average PSNR in dB versus number of frames for MPEG and OBC encoding.*

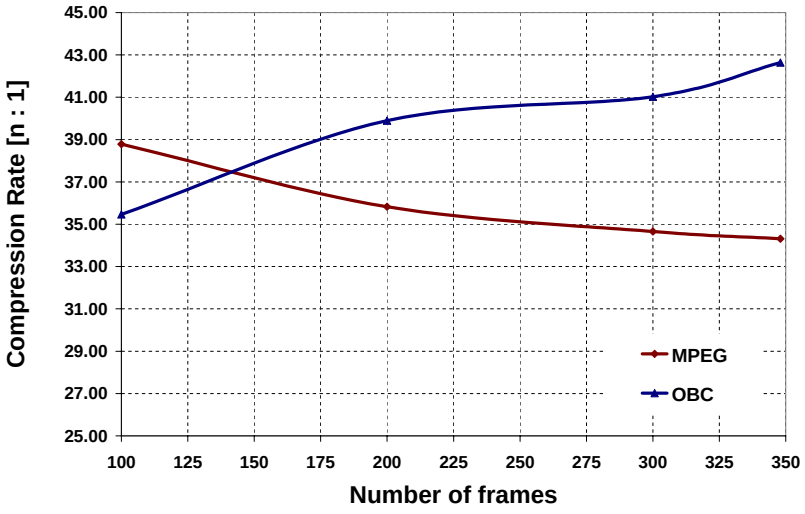


FIG. *8.5:* BILLS2 *sequence : Compression Rate in n : 1 versus number of frames for MPEG and OBC encoding.*

8.2.2 Sequence CARPHONE

OBC shows a similar behavior with the CARPHONE sequence as with the BILLS2. The sequence CARPHONE is also one of the standard sequences used by the video compression community. The video clip in QCIF color format shows a young man in a car talking to a camera. It contains 382 frames and lasting about 12.7 seconds (30 fps). The speaker is significantly changing his facial expression and speaks very expressively as if he wanted to convince the person he is speaking to of something. Efficient tracking is difficult, perhaps impossible, since the speaker is changing the distance of his head to the camera several times. Moreover, the background changes significantly while the car is driving. A pre-recorded sequence such as CARPHONE offers no possibility to make up for such changes by zooming[1]. So, like BILLS2, the 80 x 80 pixels area cut out around the head of the talking person has significant variance, seriously compromising OBC performance.

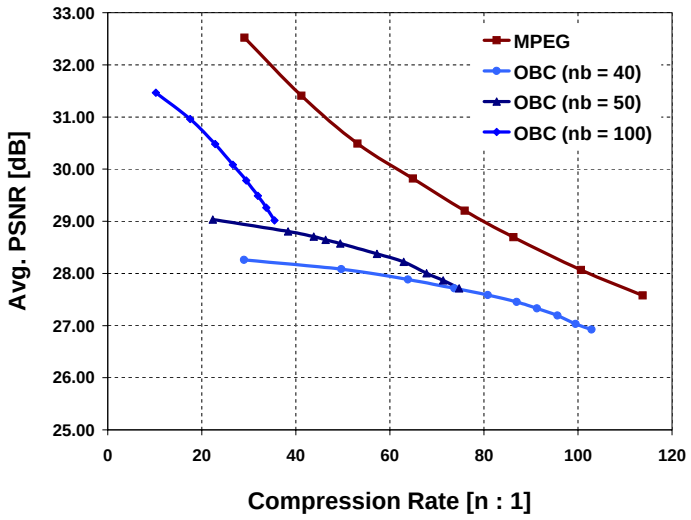


FIG. *8.6:* CARPHONE *sequence : Avg. PSNR in [dB] vs. Compression Rate in n : 1 for MPEG and OBC encoding in various configurations.* nb *indicates the number of basis images used.*

Figure 8.6 shows the average PSNR to compression rate graph for the CARPHONE sequence, whereas figure 8.7 illustrates the artifacts of MPEG and OBC coding by displaying the frames that have been reconstructed with the highest and lowest PSNR, plus their originals. We can easily identify the blocking artifacts of DCT/DPCM-based MPEG compression. Again, the OBC reconstruction error causes some fuzziness in the decoded frame. Some of the encoding parameters varied in figure 8.6 are listed in table 8.4.

[1] Scaling would be a not very attractive alternative

TAB. *8.4: Sequence* CARPHONE*, face region 80 x 80 pixels : Some encoding parameters and results from figure 8.6 in numbers*

OBC coding			
Basis size	Basis quality	Compression Rate [n : 1]	Avg. PSNR [dB]
40	55	102.90	26.92
40	60	99.49	27.03
40	80	80.81	27.59
40	90	63.87	27.88
40	95	49.70	28.08
40	100	28.99	28.26
MPEG coding			
	P : B : I quantizer scales	Compression Rate [n : 1]	Avg. PSNR [dB]
	24 : 22 : 19	113.75	27.58
	21 : 19 : 16	100.63	28.07
	18 : 16 : 13	86.29	28.69
	14 : 12 : 9	64.90	29.89
	12 : 10 : 7	53.12	30.54
	8 : 6 : 3	29.07	32.51

Figures 8.9 and 8.8 show the compression rate and the PSNR reconstruction quality in [dB] versus the number of frames. The underlying experiment is coding the first 100, 200, 300, and all 382 frames. Basis sizes are adapted in order to keep the compression rate at about the same value. We see that if we keep reconstruction quality of OBC at about 3.5 dB under MPEG[2], we can get a 15% higher compression rate. The properties of CARPHONE mentioned in the above, which actually make it a full-motion video, prevent OBC from further approaching MPEG compression. What we should keep from the results with CARPHONE is that a) even under unfavorable conditions, OBC is doing a decent job, and b) OBC *is* dependent on input normalization. Again, we notice that OBC reconstruction quality approaches that of MPEG for very high compression rates.

[2]OBC can in this case not do better for 100 basis images or less.

FIG. *8.7: Example frames from sequence* CARPHONE *showing the coding artifacts, illustrated by the best (maximum PSNR) and worst (minimum PSNR) reconstruction : MPEG encoding with P : B : I quantizer scales of 24 : 22 : 19 (avg. PSNR 27.58 ; CR 113.75 : 1), and for OBC with 40 basis images and quality parameter 55 (avg. PSNR 26.92 ; CR 102.90 : 1).*

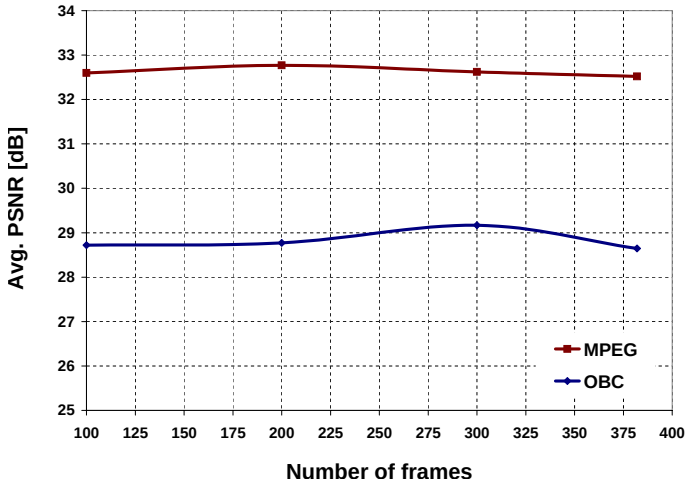


FIG. *8.8:* CARPHONE *sequence : Average PSNR in dB versus number of frames for MPEG and OBC encoding.*

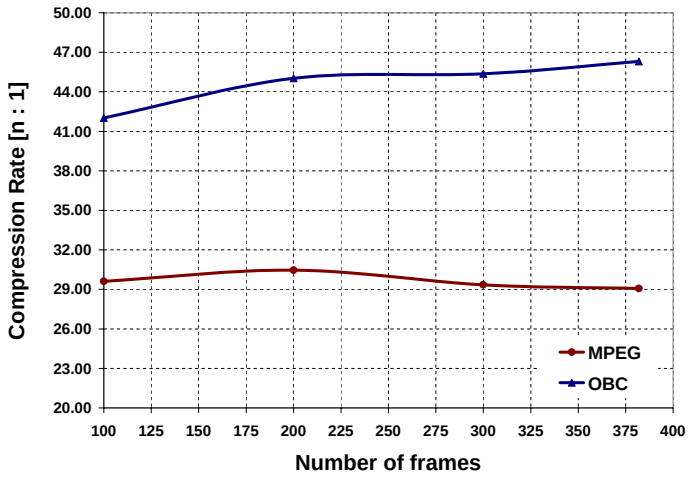


FIG. *8.9:* CARPHONE *sequence : Compression Rate in n : 1 versus number of frames for MPEG and OBC encoding.*

8.2.3 Sequence CLAIRE

The sequence CLAIRE is again one of the standard sequences used by the video compression community. The video clip in QCIF color format shows a lady speaking, apparently a news reader, containing 494 frames and lasting about 16.5 seconds (30 fps). The speaker is making mostly nodding movements with her head, and her facial expression ranges from serious to smiling. Her head stays relatively calm so generating a sequence of 80 x 80 pixels with her tracked face is easy. This sequence is nicely normalized to the head of the talking person, which enables OBC to perform better in terms of compression as well as reconstruction quality.

Figure 8.10 shows the average PSNR to compression rate graph for the CLAIRE sequence. The average PSNR to compression rate graph displays how OBC is performing w.r.t. MPEG. Both the MPEG as well as the OBC encoding parameters were varied in order to show how their performance changes when encoding the entire sequence. For MPEG encoding, the parameters varied were essentially the quantizer scales. We observe that curves are partially crossing. In general, we can almost always get a significantly better performance with OBC than with MPEG, given that we adapt the basis size. Some of the encoding parameters varied in figure 8.10 are listed in table 8.5.

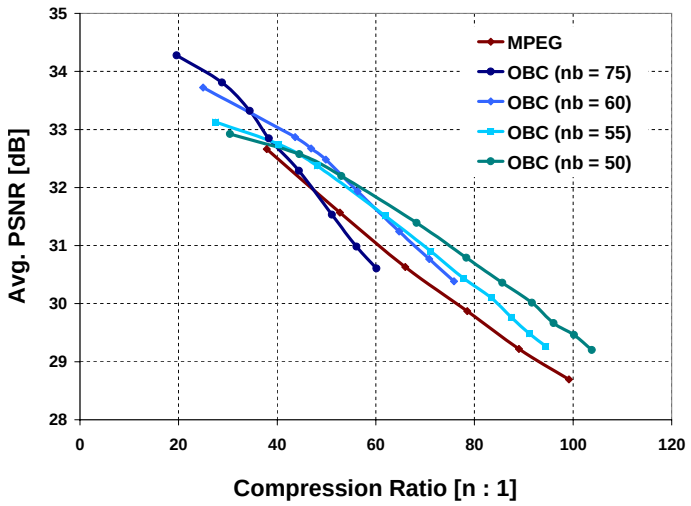


FIG. *8.10:* CLAIRE *sequence : Avg. PSNR in [dB] vs. Compression Rate in n : 1 for MPEG and OBC encoding in various configurations.* nb *indicates the number of basis images used.*

Figure 8.11 illustrates the artifacts of MPEG and OBC coding by displaying the frames that have been reconstructed with the highest and lowest PSNR, plus their originals. We can easily identify the blocking artifacts of DCT/DPCM-based MPEG compression. The OBC reconstruction error consists of a more or less pronounced fuzziness of the decoded frame.

TAB. *8.5: Sequence* CLAIRE*, face region 80 x 80 pixels : Some encoding parameters and results from figure 8.10 in numbers*

OBC coding			
Basis size	Basis quality	Compression Rate [n : 1]	Avg. PSNR [dB]
60	75	75.84	30.39
60	80	70.83	30.77
60	85	64.72	31.24
60	90	56.28	31.93
60	95	43.64	32.87
60	100	24.98	33.72
MPEG coding			
	P : B : I quantizer scales	Compression Rate [n : 1]	Avg. PSNR [dB]
	14 : 12 : 9	78.55	29.87
	12 : 10 : 7	65.94	30.63
	10 : 8 : 5	52.70	31.57
	8 : 6 : 3	37.88	32.66

OBC is performing better on CLAIRE than MPEG, which is due to an efficient normalization of the input image stream by tracking. Up to very high compression ratios, OBC has a margin of 1+ dB. Varying the basis size and compression, we can efficiently increase compression while maintaining a decent image quality.

Figures 8.12 and 8.13 show the compression rate and the average PSNR reconstruction quality in [dB] versus an increasing number of frames. The underlying experiment is coding the first 100, 200, 300, 400, and all 494 frames. Basis sizes are adapted in order to keep the compression rate at about the same value. We see that for comparable reconstruction quality, OBC yields a compression rate topping that of MPEG by about 6.6, which in this case corresponds to an improvement of 17.5 %. Figure 8.11 then illustrates the artifacts of MPEG and OBC.

FIG. *8.11: Example frames from sequence* CLAIRE *showing the coding artifacts, illustrated by the best (maximum PSNR) and worst (minimum PSNR) reconstruction : MPEG encoding with P : B : I quantizer scales of 14 : 12 : 9 (avg. PSNR 29.87 ; CR 78.55 : 1), and for OBC with 60 basis images and quality parameter 75 (avg. PSNR 30.39 ; CR 75.84 : 1).*

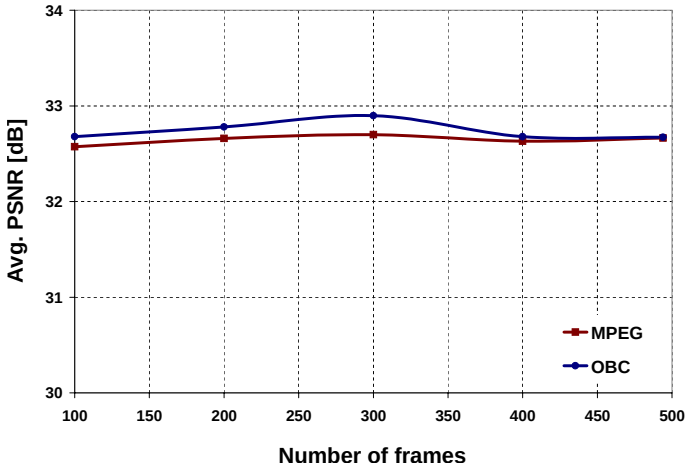


FIG. *8.12:* CLAIRE *sequence : Average PSNR in dB versus number of frames for MPEG and OBC encoding.*

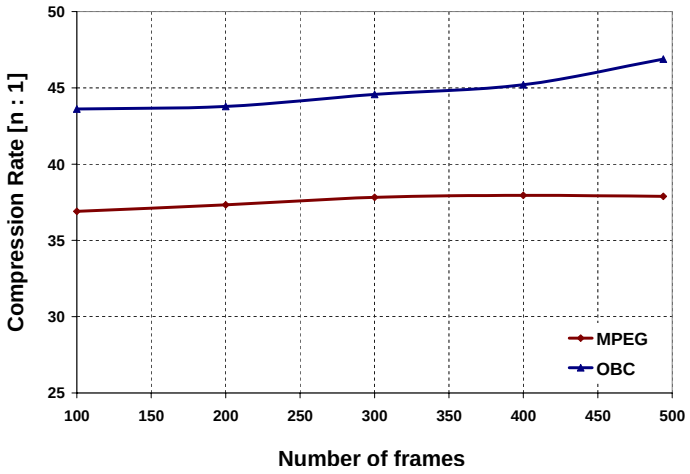


FIG. *8.13:* CLAIRE *sequence : Compression Rate in n : 1 versus number of frames for MPEG and OBC encoding.*

8.2.4 Sequence GRANDMA

The GRANDMA sequence is another standard test sequence [Vid01a, Vid01b] showing an old lady sitting in an armchair, speaking. It contains 870 frames and lasts about 29 seconds (30 fps). The length of the sequence is of particular interest since assuming a frame rate of 6 fps, for instance for a mobile communication scenario, 870 frames represent about $2\frac{1}{2}$ minutes. The head of the speaker is moving only little over the sequence. The is realistic as a person communicating over a small (mobile) video telephone is supposed to hold his head relatively still in order to watch his own (local) screen. Since the head of the speaker in the GRANDMA sequence stays calm, generating a sequence of 80 x 80 pixels with her face tracked is easy.

The sequence normalized to the head of the talking person enables OBC to perform better than MPEG in terms of compression as well as reconstruction quality. Figure 8.14 compares MPEG and OBC drawing the average PSNR versus the compression rate for various encoding parameters such as varying quantizing scale for MPEG or number of basis images for OBC, some of which are listed in table 8.6 together with their corresponding results . Figure 8.15 shows the artifacts of OBC and MPEG for the case of high compression (making the artifacts more visible).

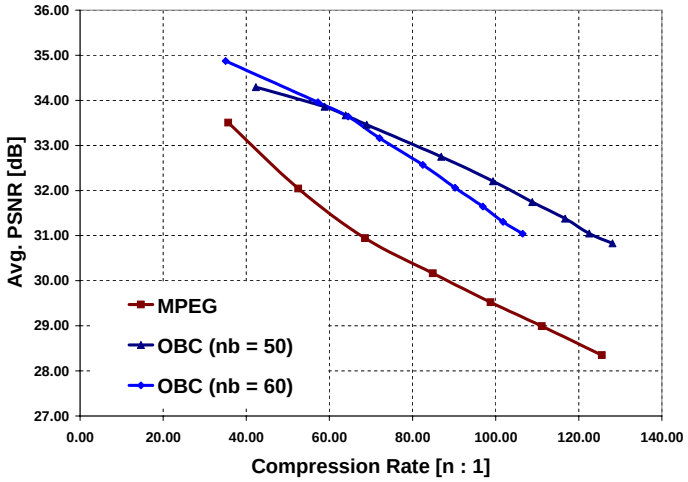


FIG. *8.14:* GRANDMA *sequence : Avg. PSNR in [dB] vs. Compression Rate in n : 1 for MPEG and OBC encoding in various configurations.* nb *indicates the number of basis images used.*

Figures 8.17 and 8.16 show the compression rate and the PSNR reconstruction quality in [dB] versus the number of frames. The underlying experiment is coding the first 100, 200, 400, and all 870 frames. Of course, basis sizes are adapted in order to keep the compression rate at about the same value. We see that at 870 frames for comparable reconstruction quality, OBC yields a compression rate topping that of MPEG by about 28.8, which in this case corresponds to an

TAB. *8.6: Sequence* GRANDMA*, face region 80 x 80 pixels : Some encoding parameters and results from figure 8.14 in numbers*

OBC coding			
Basis size	Basis quality	Compression Rate [n : 1]	Avg. PSNR [dB]
50	80	108.80	31.75
50	85	99.37	32.20
50	90	86.88	32.75
50	95	68.97	33.46
50	100	42.28	34.29
MPEG coding			
	P : B : I quantizer scales	Compression Rate [n : 1]	Avg. PSNR [dB]
	18 : 16 : 13	111.13	28.99
	16 : 14 : 11	98.77	29.52
	14 : 12 : 9	84.88	30.16
	12 : 10 : 7	68.58	30.94
	8 : 6 : 3	35.67	33.51

improvement of 81 %. The lesson from the experiments with the GRANDMA sequence is that with a well-normalized sequence, OBC gains even more advantage over MPEG as the sequence gets longer.

FIG. *8.15: Example frames from sequence* GRANDMA *showing the coding artifacts, illustrated by the best (maximum PSNR) and worst (minimum PSNR) reconstruction : MPEG encoding with P : B : I quantizer scales of 18 : 16 : 13 (avg. PSNR 28.99 ; CR 111.13 : 1), and for OBC with 50 basis images and quality parameter 80 (avg. PSNR 31.75 ; CR 108.80 : 1).*

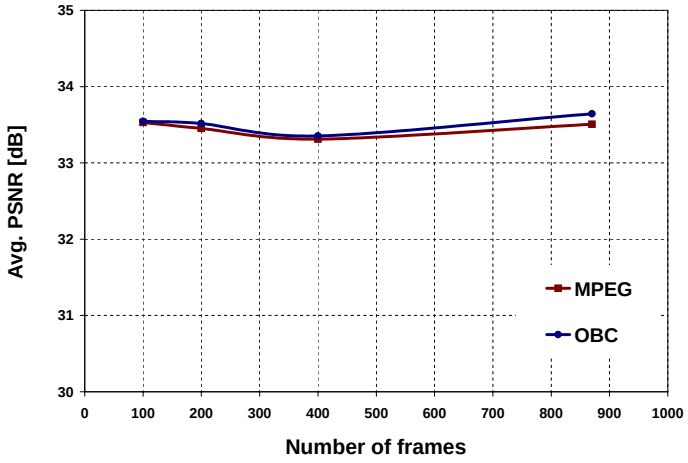


FIG. *8.16:* GRANDMA *sequence : Average PSNR in dB versus number of frames for MPEG and OBC encoding.*

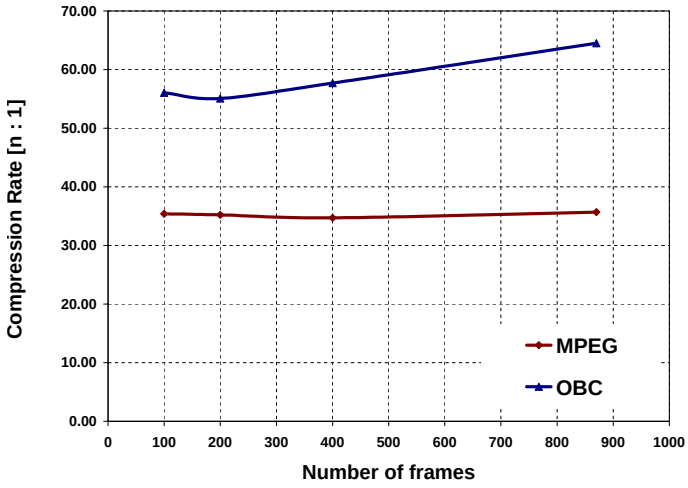


FIG. *8.17:* GRANDMA *sequence : Compression Rate in n : 1 versus number of frames for MPEG and OBC encoding.*

8.2.5 Sequence JEANBAPTISTE

JEANBAPTISTE is a sequence recorded in our laboratory. It shows a head-and-shoulders scene with the head making some movements while the shoulders stay relatively calm. It is therefore easy to track, and the resulting 80 x 80 pixel video sequence is very well centered. The normalized sequence is an excellent input for the OBC codec, and indeed, the OBC codec easily outperforms MPEG in terms of compression as well as reconstruction quality. This actually shows how crucial good tracking is for OBC.

Figure 8.18 shows that for low as well as high compression rate, OBC yields a substantially better quality. Table 8.7 shows some of the values of figure 8.18 in numbers, whereas figure 8.19 illustrates the artifacts of OBC and MPEG.

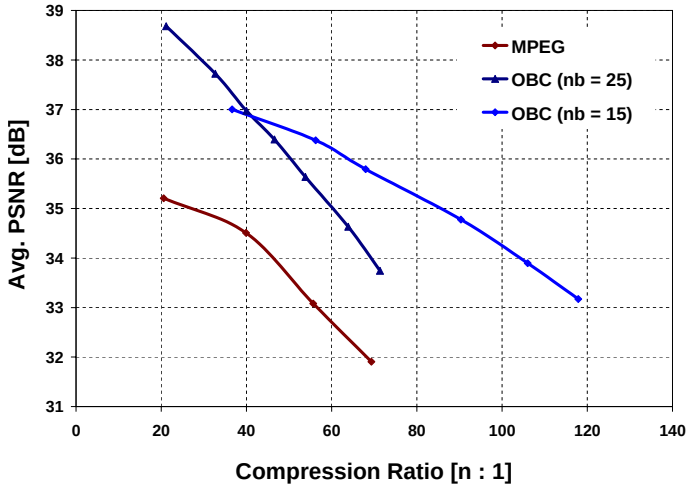


FIG. *8.18:* JEANBAPTISTE *sequence : Avg. PSNR in [dB] vs. Compression Rate in n : 1 for MPEG and OBC encoding in various configurations.* nb *indicates the number of basis images used.*

Figure 8.20 contains the PSNR for each frame in the sequence for OBC and MPEG encoding at comparable compression rates. We note that the curve for OBC has a greater variance than the one for MPEG. In general, a high PSNR for OBC represents an image that is close to its cluster center. Deep dips in the OBC curve occur when a frame is far from all the cluster centers. This usually corresponds to features that have not been captured by the sample selection process. Choosing a bigger basis size often remedies those effects up to a certain point.

TAB. *8.7: Sequence* JEANBAPTISTE*, face region 80 x 80 pixels : Some encoding parameters and results from figure 8.18 in numbers*

OBC coding			
Basis size	Basis quality	Compression Rate [n : 1]	Avg. PSNR [dB]
25	80	71.34	33.74
25	93	46.58	36.29
25	95	39.99	36.97
25	100	21.14	38.68
MPEG coding			
	P : B : I quantizer scales	Compression Rate [n : 1]	Avg. PSNR [dB]
	12 : 10 : 7	69.33	31.91
	10 : 8 : 5	55.67	33.08
	8 : 6 : 3	39.88	34.51
	5 : 4 : 1	20.59	35.21

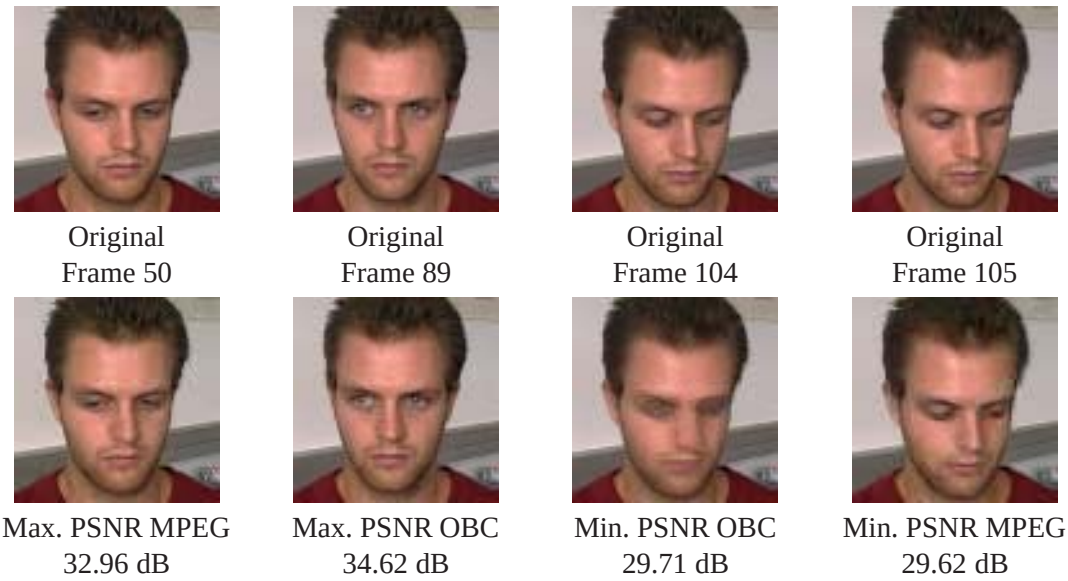


FIG. *8.19: Example frames from sequence* JEANBAPTISTE *showing the coding artifacts, illustrated by the best (maximum PSNR) and worst (minimum PSNR) reconstruction : MPEG encoding with P : B : I quantizer scales of 12 : 10 : 7 (avg. PSNR 31.91 ; CR 69.33 : 1), and for OBC with 25 basis images and quality parameter 80 (avg. PSNR 33.74 ; CR 71.34 : 1).*

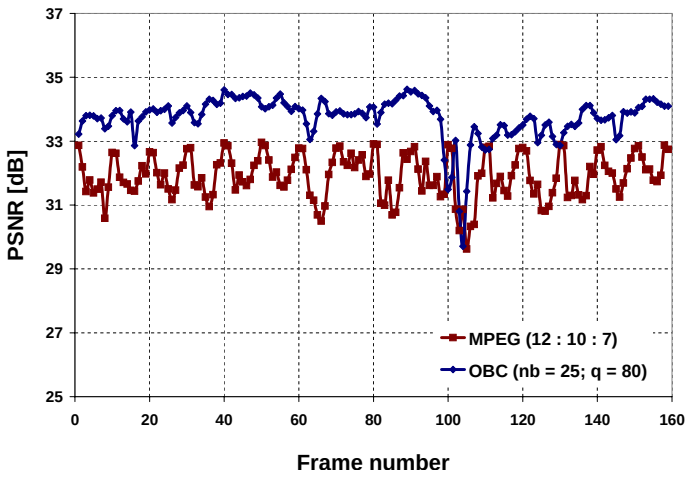


FIG. *8.20: PSNR versus number of frame for sequence* JEANBAPTISTE *for MPEG encoding with P : B : I quantizer scales of 12 : 10 : 7 (Compression Rate 69.33 : 1 ; avg. PSNR = 31.91) and OBC encoding with 25 basis images and quality parameter 80 (Compression Rate 71.34 : 1 ; avg. PSNR = 33.74).*

8.2.6 Sequence MOM

The sequence MOM is again one of the standard sequences used by the video compression. The video clip in QCIF color format shows a lady speaking, probably a news reader, containing 494 frames and lasting about 16.5 seconds (30 fps). The speaker is making mostly nodding movements with her head, and her facial expression ranges from serious to smiling. The background colors are relatively close to face color, which is why the tracker was not as precise as with, e.g. , the JEANBAPTISTE sequence.

Figure 8.21 shows the interesting fact that OBC starts to outperform MPEG at high compression rates. As we saw from the BILLS2 sequence on, OBC has a better average PSNR / Compression rate ratio at higher compression rates. This may come handy for certain applications such as mobile video telephony.

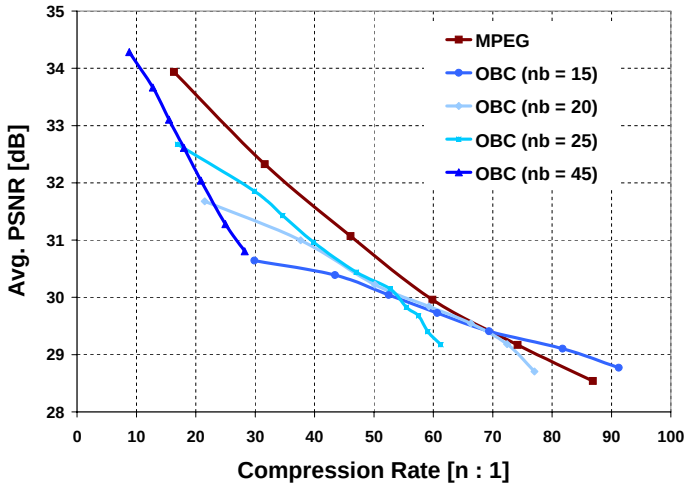


FIG. *8.21:* MOM *sequence : Avg. PSNR in [dB] vs. Compression Rate in n : 1 for MPEG and OBC encoding in various configurations.* nb *indicates the number of basis images used.*

We see in figure 8.22 that both techniques have their best performance (OBC for 15 basis images and quality 80 ; MPEG for P : B : I quantizer values of 16 : 14 : 11) at frame 59. Their worst reconstruction quality falls almost on the same frame, too. In case of OBC compression the reason is that this image is an outlier with respect to the cluster centers computed in the sample selection process. The reason for a bad MPEG performance may be that the forward-backward-prediction captures a movement of the head only poorly. A frame by frame display of the PSNR is corresponding to figure 8.22 is shown in figure 8.23.

TAB. *8.8: Sequence* MOM*, face region 80 x 80 pixels : Some encoding parameters and results from figure 8.21 in numbers*

OBC coding			
Basis size	Basis quality	Compression Rate [n : 1]	Avg. PSNR [dB]
15	80	91.25	28.77
15	85	81.77	29.10
15	90	69.38	29.41
15	95	52.48	30.04
15	100	29.88	30.64
MPEG coding			
	P : B : I quantizer scales	Compression Rate [n : 1]	Avg. PSNR [dB]
	16 : 14 : 11	86.88	28.54
	14 : 12 : 9	74.22	29.17
	12 : 10 : 7	59.86	29.96
	10 : 8 : 5	46.06	31.07
	8 : 6 : 3	31.61	32.32

FIG. *8.22: Example frames from sequence* MOM *showing the coding artifacts, illustrated by the best (maximum PSNR) and worst (minimum PSNR) reconstruction : MPEG encoding with P : B : I quantizer scales of 16 : 14 : 11 (avg. PSNR 28.54 ; CR 86.88 : 1), and for OBC with 15 basis images and quality parameter 80 (avg. PSNR 28.77 ; CR 91.25 : 1).*

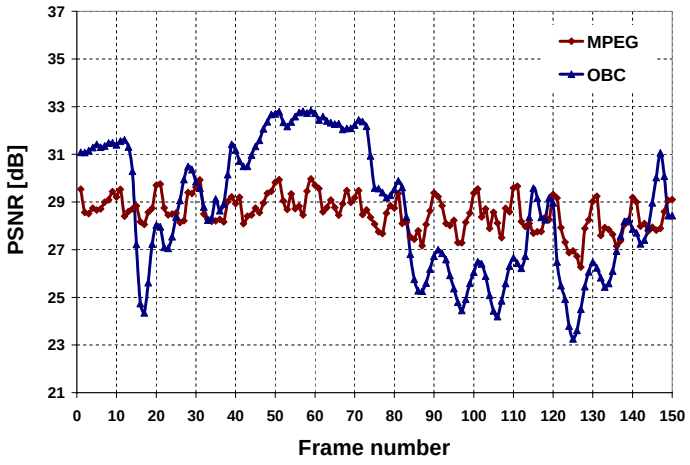


FIG. *8.23: PSNR versus number of frame for sequence* MOM *for MPEG encoding (Compression Rate 86.88 : 1 ; avg. PSNR = 28.54) and OBC encoding (Compression Rate 91.25 : 1 ; avg. PSNR = 28.77).*

8.2.7 Sequence TALKINGHEAD

The TALKINGHEAD sequence was, just like the BILLS2 and the JEANBAPTISTE sequences, recorded in our laboratory. It shows a person doing some relatively big movements with his head. The movements are perpendicular to the camera, so the head virtually always has the same size. As with the CARPHONE sequence, tracking creates a changing background. Again we notice that OBC performs the better, the higher the compression rates, although it has the handicap of a changing background. The results of OBC are comparable with those of MPEG for the entire sequence, as can be seen in figure 8.26.

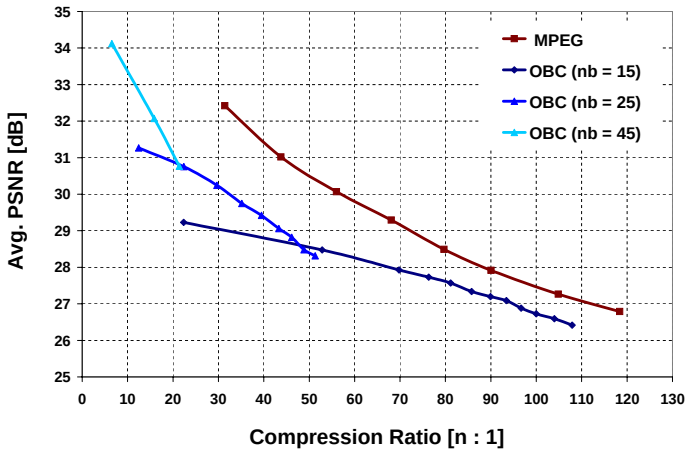


FIG. *8.24:* TALKINGHEAD *sequence : Avg. PSNR in [dB] vs. Compression Rate in n : 1 for MPEG and OBC encoding in various configurations.* nb *indicates the number of basis images used.*

Figure 8.26 shows the PSNR for each frame for high compression. We note that the lowest PSNR for OBC (frame # 1, see also figure 8.25) is extremely low. The reason is that the face tracker was run on a pre-recorded sequence where the start and end position of the face are not exactly the same. The face tracker sees thus an extremely saccadic movement of the head, compromising its precision. In live situations this would be an unlikely event. The face tracker could be adapted by keeping the values of the covariance matrix artificially big, which would give up some precision. However, we preferred to accept the low PSNR of at the turnover of the TALKINGHEAD sequence as an exception.

TAB. *8.9: Sequence* TALKINGHEAD*, face region 80 x 80 pixels : Some encoding parameters and results from figure 8.24 in numbers*

OBC coding			
Basis size	Basis quality	Compression Rate [n : 1]	Avg. PSNR [dB]
15	40	104	26.59
15	60	89.91	27.20
15	70	81.13	27.57
15	80	69.77	27.92
15	90	52.85	28.48
MPEG coding			
	P : B : I quantizer scales	Compression Rate [n : 1]	Avg. PSNR [dB]
	21 : 19 : 16	104.88	27.27
	18 : 16 : 13	90.05	27.92
	16 : 14 : 11	79.68	28.49
	14 : 12 : 9	68.06	29.29
	12 : 10 : 7	56.02	30.07

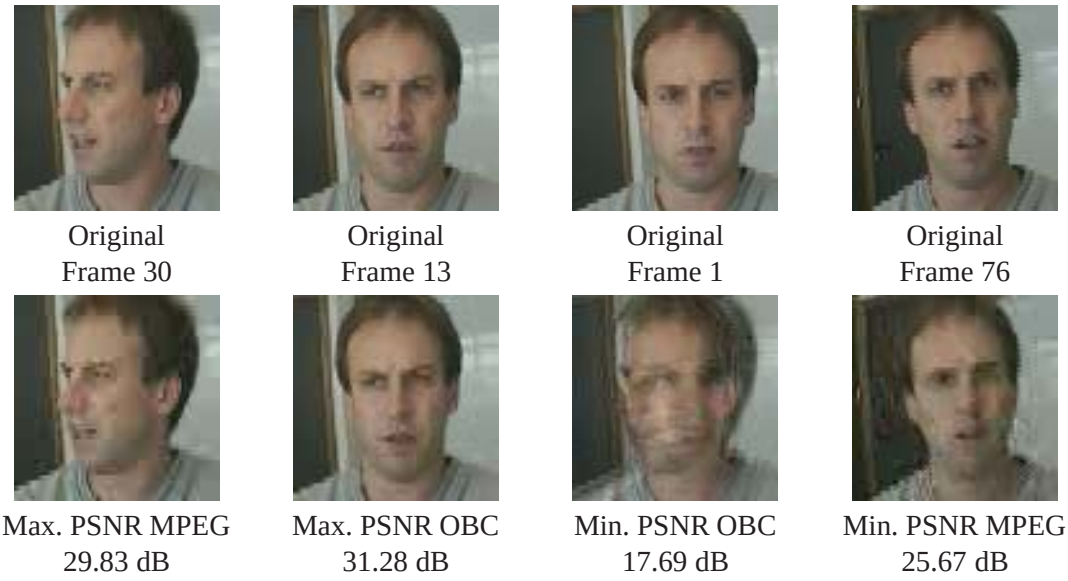


FIG. *8.25: Example frames from sequence* TALKINGHEAD *showing the coding artifacts, illustrated by the best (maximum PSNR) and worst (minimum PSNR) reconstruction : MPEG encoding with P : B : I quantizer scales of 18 : 16 : 13 (Compression Rate 90.05 : 1 ; avg. PSNR = 27.92) and OBC encoding with 15 basis images and quality parameter 60 (Compression Rate 89.91 : 1 ; avg. PSNR = 27.20).*

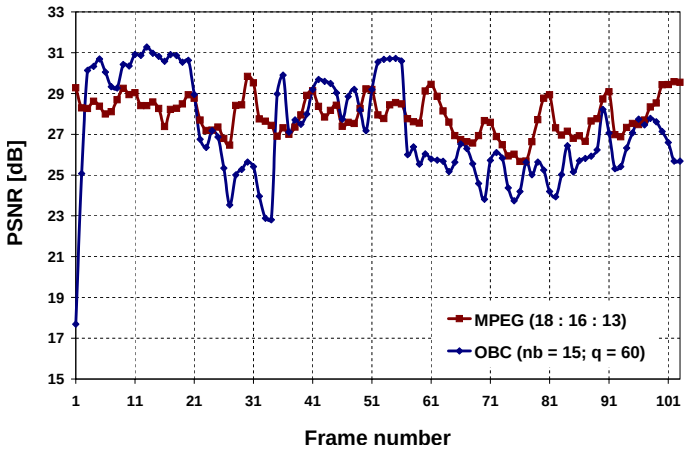


FIG. *8.26: PSNR versus number of frame for sequence* TALKINGHEAD *for MPEG encoding with P : B : I quantizer scales of 18 : 16 : 13 (Compression Rate 90.05 : 1 ; avg. PSNR = 27.92) and OBC encoding with 15 basis images and quality parameter 60 (Compression Rate 89.91 : 1 ; avg. PSNR = 27.20).*

8.2.8 Computing Speed

Computing performance is important when we talk about video compression which is supposed to be in real-time. For MPEG as well as for OBC it is possible to trade off quality for computing time. Variations of computing time for MPEG are related to the motion vector search algorithm, and the number of I, B, and P frames [Ber97]. Table 8.10 shows that changing the motion vector search algorithm for P-frames has no impact on quality or compression. For all other measurements in this section, we used the TWOLEVEL algorithm. For OBC, the parameters affecting computing time are the size of the images to be processed, the number of basis images, and the number of images in the sequence to be encoded. All following calculations were performed on a personal computer equipped with a 600 Mhz Pentium III processor, 512 kByte cache, and 1 GByte RAM.

TAB. *8.10: Sequence* GRANDMA *: Computing time for MPEG motion vector search for P-frames with different search algorithms ; values are averaged over measurements for 100, 200, 400, 600, and 870 encoded frames.*

Algorithm	Comput. time [msec/frame]	Compr. Rate [n : 1]	Avg. PSNR [dB]
EXHAUSTIVE	199	11.8	33.5
SUBSAMPLE	163	11.8	33.5
TWOLEVEL	61	11.8	33.5
LOGARITHMIC	17	11.8	33.5

The numbers shown in the figures below for OBC are for limited use, since, as has been said above, the code for OBC has not been optimized yet. Measurements for MPEG are shown for the Berkeley MPEG encoder. There are probably faster MPEG software codecs out there, but even though we did not do a comparative study of MPEG codecs in terms of speed, we would not expect them at this point to be more than 10 times faster than `mpeg_encode`. So the reader can well get an impression of the computing time range current MPEG software codes are in. In any case, since they follow the MPEG standard, any MPEG encoder must produce the same compression results for equal parameter configuration. Pre-processing steps for MPEG-4 or MPEG-7 such as image content extraction and modeling are of course not considered. Note that since OBC is an appearance-based technique, this pre-processing is already implicitly included.

Figure 8.27 compares OBC and MPEG computing time per frame for the TALKINGHEAD sequence for various image sizes. OBC as it is implemented today loads the entire sequence into memory before processing it. Unfortunately, this limits the possible image size being processed, since each pixel is converted into a 32 bit floating point number. We see that MPEG is about ten times faster than OBC.

Figure 8.28 illustrates the dependence of OBC on the number of basis images. At the beginning for 5 basis images the sample selection process is dominated by computing the new cluster centers,

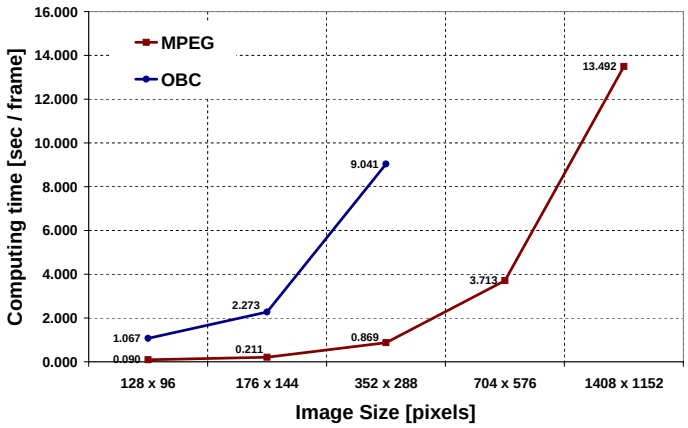


FIG. *8.27: Sequence* TALKINGHEAD *: Computing speed per frame in [sec/frame] vs. images size*

i.e., the average of all samples in each cluster. From 15, 20 basis images on, its the assigning of the samples to the clusters, which is using most of the computing time. For selection algorithms other than the EQUIDISTANT algorithm, finding the cluster centers may dominate the computing time consumption.

Figure 8.29 and 8.30 shows the dependence of the computing time as a function of the number of images to be encoded. We see that the actual encoding step, i.e., the mapping of the images into the created orthonormal basis space, is the most consuming computing step. This is mostly due to the increasing basis size.

8.3 Critical discussion of experimental results

This chapter presented first results on OBC coding. In order to give reproducible results, OBC was used on pre-recorded sequences. We saw that OBC performed differently on different input sequences meaning that OBC coding is sensitive to its input data. This is an expected result, since the entire encoding procedure is relying on normalization of the input stream. OBC is implicitly collecting information about the image content, and it must perform worse than an algorithm not dependent on input data, if this image content is not well accessible. This is the case when the principal object in the scene to be encoded is not well normalized.

This puts the focus on the face tracker. As of today the face tracker is a one-module color tracker. The results of the encoding of the BILLS2, CARPHONE, and TALKINGHEAD sequences show that imprecise tracking severely compromises compression results. These are the sequences

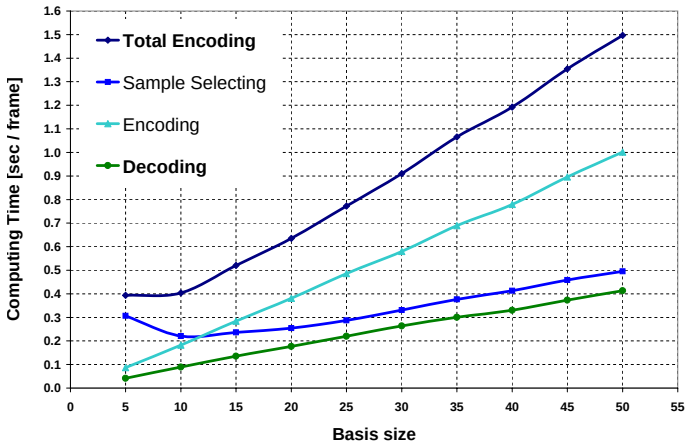


FIG. *8.28: Sequence* TALKINGHEAD *: Computing time per frame in [sec / frame] vs. basis size in number of basis images.*

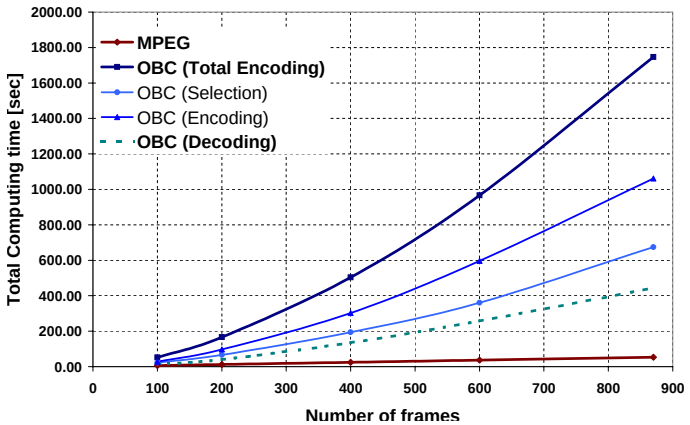


FIG. *8.29: Sequence* GRANDMA *: Computing time in [sec] vs. number of frames to be encoded The basis size is adapted to yield an approximatively constant reconstruction quality.*

where the head moves considerably over the image. It is up to further investigation to find out if OBC needs sub-pixel precision or scaling. Adding more modules such as a background subtractor

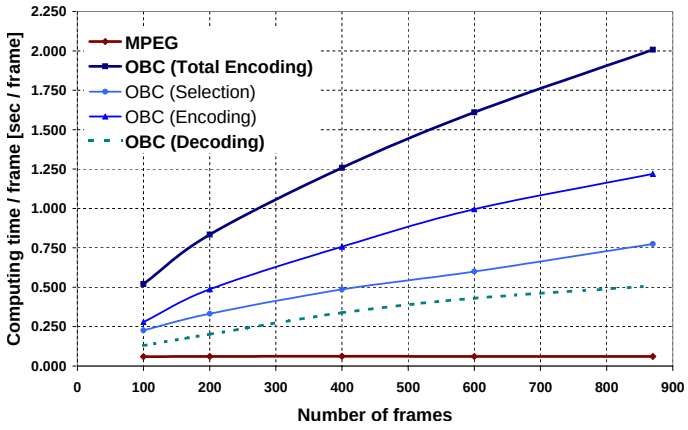


FIG. *8.30: Sequence* GRANDMA *: Computing time per frame in [sec / frame] vs. number of frames to be encoded*

or a motion detector to the face tracker should further contribute to precision and stability. Especially the tracked object's extent should carefully be reconsidered, as the results of the CARPHONE sequence suggest.

Another big drawback of the results above are the reconstruction quality metrics used. Degradations in reconstructed images are of different nature for MPEG and OBC, which actually makes them incomparable. We saw in section 7.4 that none of the currently used image quality measures reflects human perception. A series of experiments with humans using psychological methods to capture their impressions of image quality should be the way to go. This research should be conducted with the goal of finding a mathematical description of human perception in mind.

A considerable disadvantage of OBC compared to MPEG is currently its computing requirements. This is basically due to an implementation which was designed for flexibility and extensibility rather than efficiency. Here are some measures to compensate for complexity of the algorithm :

a. Using a pixel step parameter in order to adapt the internal resolution of the codec.
b. Avoid `byte` to `float` to `int` conversions etc.
c. Masking the input sequence with the tracker output
d. Avoid using temporary files
e. Using `inline` functions and look-up tables.

Similar improvements applied to the face tracker's code yielded a reduction of computing time by a factor of 5 (compare the results in [SC00] and in figure 6.5).

8.4 Conclusion

This chapter compared OBC to MPEG for a number of video sequences. These sequences contain head and shoulder scenes, which is the scenario for which OBC was designed. The comparison criteria were *compression rate*, *reconstruction quality*, and *computing speed*. The results showed that OBC is a valid alternative to DCT/DPCM type compression such as MPEG. In terms of compression rate and reconstruction quality, OBC performs on a level at least comparable to MPEG. If the prerequisites of OBC-coding such as normalizing and a one-principal-object-scenario like head-and-shoulders scenario are fulfilled, then OBC can show significantly better reconstruction quality at the same compression rate, or vice versa, than MPEG coding.

The computational cost of OBC, however, is currently higher, which is due to a code not yet optimized for computing efficiency. We proposed a number of improvement measures to reduce this problem. An improvement factor of 10, based on the experience with the optimization of the face tracker code, should be realistic, which would make OBC very competitive compared to MPEG.

A second field of improvement is the enhancement of OBC with features and functionality. Some new features of OBC are currently being tested or designed but are to be considered too immature to be presented here. These contain

Online OBC : An incremental version using the Gram-Schmidt procedure produced comparable results to those reported above in an online version of OBC. Major problem here is the basis updating algorithm.

Offline, open universe coding : Privacy protection or the wish to use an animated talking head on web pages or for user help suggest the offline creation of a basis. Studies about the required completeness of that basis need to be conducted. Short sequences mdae from such a basis could be mapped onto phonemes, contributing to new human-computer interfaces.

Quality of feature reconstruction : Make OBC sensitive to movements of the eyes and mouth by multiplying these regions by an extra weighting factor during the comparison of images with the cluster centers. The information for this could come from the face tracker. Preservation of edges may be another quality criteria for the OBC codec producing fuzziness as principal artifact.

Add modules to face tracker : In order to improve tracking precision, adding further modules such as background subtraction and motion estimation to the face tracker could improve precision. Sub-pixel precision may be necessary.

Lossless compression of encoding parameters : From a certain basis size on it is worthwhile considering a lossless compression of the encoding parameters, e.g., by ZIP compression. This is valid only for closed universe coding.

Precision of encoding parameters : 32 bits for each encoding parameter may not be necessary. Experiments to find the minimum required precision for the encoding parameters should be conducted.

Reconstruction quality measures modeling the human visual system could give new insight into the entire coding process.

Chapitre 9

Conclusions

This thesis presents a new approach to the encoding of video data for communication. The approach is based on the *appearance*[1] of one principal object in a stream of video data. In this context, this object is a talking head in a head-and-shoulders scenario. Appearance based methods require normalization by tracking, which in turn uses techniques from computer vision. We used computer vision techniques to create a fast and robust face tracking system based on the detection of skin colored pixels in an image. This tracker is part of a video codec based on *Orthonormal Basis Coding (OBC)*.

Two conditions have to be fulfilled so that OBC can be efficient :

a. (Face) tracking must be precise : The tracked object should always have the same position and size within the image.

b. The recorded sequence is repetitive in its content ; this is not a *conditio sine qua non*, but it increases greatly the efficiency of OBC

We could show in this work that if tracking is sufficiently precise, OBC outperforms by far conventional encoding schemes such as MPEG in terms of reconstruction quality at comparable compression rates. OBC performs particularly well at high to very high compression rates. This is due to the fact that OBC does not slice up the images into little blocks, but treats them as a whole vector. This resembles intentionally vector quantization or, if we speak in terms of pattern recognition, clustering. We even make explicit use of algorithms from those fields.

For some years now research in video compression was busy producing incremental improvements rather than re-thinking video compression as a whole. Standardization work in this area, notably on MPEG-4 and MPEG-7, build multimedia codecs around MPEG-1 or MPEG-2 compression since there is simply nothing out there to replace the DCT/DPCM/VLC based compression scheme. The work has been passed to the computer vision community, who is trying to extract and model image information, which is supposed to be parameterized and used for interactive video. Using Wavelets in stead of DCT may produce better results every now and then, but its nothing more than replacing one fixed-data-model based transform coding algorithm by another, where the complexity on the coding side is even further increased.

[1] see page 27

We claim that in order to re-think the entire problem, we need to go one abstract level higher. This is why we restarted the design process from the fundamental concepts of information theory. We think that we could show with this work that it is still worthwhile doing so. Appearance-based video compression *can* do both, efficiently compression video data *and* collect and use image information. The drawback of being restricted to a certain scenario is relative since image content extraction *always* needs assumption about the image content. Moreover, there is plenty of applications which follow exactly this scenario, from (mobile) video telephony to animated talking heads for human-computer interaction.

9.1 Contributions

This thesis has two main contributions to the area of video compression :

⇒ The use of tracking methods from computer vision to enable appearance-based coding actually makes sense. We could show that if a sequence is well normalized to an object, its eigenspace representation is denser than that of a random sequence.

⇒ Vector quantization (or clustering) can be used as an approximation to principal components analysis.

Principal components analysis (PCA) is a technique used (or from) multivariate data analysis [Gif90] and is the interpretation of the eigenspace representation of vector data. It tells us how representative the first *n* eigenvectors are of the data represented. A handy property of PCA is the exact knowledge of any error made. The eigenspace is usually computed by the Karhunen-Loève expansion. It is the densest possible representation of vector data and thus implicitly appropriate for compression. Calculating an eigenspace representation is usually too complex for being used on a *n* x 100 image vector set, but it becomes feasible if a representative set of 1 to 75 vectors are selected. This selection is done for this work by clustering.

At every step of OBC, there is still room for improvement. Current and future research tries to better OBC by trying out different algorithms for basis image selection. This is important considering the computing times of OBC compression (see chapter 8). Other improvements include code optimization and adding features and functionality such as online coding, offline, but open universe coding, design appropriate image reconstruction quality measures, lossless compression and reduced precision of encoding parameters (see section 8.4).

Almost as a side-product, an efficient tracking algorithm based on the detection of skin color has been developed. This algorithm uses the first and second moments of the probability distribution of skin colored pixels in an image and subsequent weighting of new input data with the distribution of the previous image. This approach is inspired by robust statistics and can actually applied to any pixel-based tracking algorithm. It has successfully applied to background subtraction and motion history tracking.

9.2 Critical evaluation of the approach

It is clear that a doctoral dissertation cannot be more than a feasibility study for a new approach. There currently are still some question marks behind some points, the biggest one perhaps of which is complexity and computing power consumption. There is a reason why the – theoretically optimal – Karhunen-Loève expansion has not been used for energy compaction in video compression, and this reason is its complexity and the need to update the basis with new incoming data. The KLE has therefore been replaced by fixed-basis transform coding algorithms relying on a pre-determined data model. The most widely used transform, the Discrete Cosine Transform, is based on a Markov model of the image data and is a good approximation of the KLE for data following that model. This was the state of the art when the currently used video compression algorithms were designed, i.e., in the 1980s and early 1990s. Today's computer generation allows us to use more complex but more powerful algorithms.

The efficiency of OBC is dependent on its input data. We have shown with this work that OBC can be very efficient with an input video stream normalized in position and size to one object. Fortunately, this can be applied to a head-and-shoulders scenario, which is the basic scenario for any point-to-point human to human communication. Video E-Mail, (Mobile) Video Telephone, an interactive graphical user interface for instance a talking head on a web page are only a few application examples which are to play an increasingly important role in future communication technology.

For everything else than very long, repetitive sequences, the basis size will be much more important in data size than the reconstruction parameters. Fifteen basis vectors, for instance, produce currently only 60 Bytes per frame, so we would need 320 frames, that the parameter data amounts to one single 80 x 80 pixel RGB image. In order to reduce the amount of basis data, we can chose either a lossless or a lossy compression algorithm. Since the basis vectors are images – even as eigenvectors[2] –, we can compress them with algorithms used for image compression. As we saw in section 8.2, ZIP compression has a much lower compression rate than JPEG. The fact that JPEG compression of the basis data does not compromise the results of image reconstruction allows for its use, which greatly enhances the efficiency of OBC.

The biggest drawback of OBC so far is its computing speed[3]. OBC exhibits a considerable lag to `mpeg_encode`, where we have to consider that `mpeg_encode` is not the fastest MPEG encoding program available. On the other hand, the source code of OBC has not been optimized for computing speed yet. Images are converted back and forth from `unsigned char` pointers to `float` vectors, and temporary files are used during the encoding process. This is due to the fact that flexibility for development and easy integration of new modules was given priority when the used libraries were selected or written. These and many other things can be improved to efficiently reduce computing time while, of course, maintaining the same results in terms of image quality and compression. A hardware (DSP) version of the encoder should definitely push the OBC codec to real-time performance.

[2] That is, they have the properties of images in terms of statistical correlation of pixels

[3] It is, however, fast compared to model-based methods [EWG00].

Here are some more advantages of OBC :

⇒ Compression rate is easy controllable through variation of basis size.

⇒ Implicit motion detection through face tracking.

⇒ Based on an eigenspace-like representation of the input video stream, OBC offers possibilities for a seamless integration of applications such as recognition, animated talking head, or privacy protection.

9.3 Perspective for further research

Although OBC as it is presented in this dissertation is an already well-working video compression algorithm, many solutions for further improvement are to be tested until we can give an answer about its potential. The most critical improvements to be made are those to improve speed considerably. Once this problem is solved in a satisfactory way, applications can be designed and tested. Here are some ideas about how to further explore and exploit the concepts presented in this thesis :

Create offline, closed universe applications : Develop a video electronic mail system, as a stand alone version and as plug-in for common communication front-ends such as the Netscape© Navigator.

Build an online video telephony system : This requires an incremental version of OBC. Also, the face tracker must probably be more precise than in its current version to avoid a reconstruction quality loss such as observed with the CARPHONE sequence. The face tracker would enhance the usability of the video communication system by allowing the user to freely move in front of the camera while communicating.

Develop reconstruction quality measures based on the human visual system in collaboration with psychologists. Design a test series to quantify human visual perception in order to eventually create a mathematical description of human image quality rating.

Offline, open universe applications : These require experiments about the completeness of a basis space. Sufficiently complete appearance space representation may be used to efficiently protect privacy. A talking head for human-computer interaction on web pages or with user help functions would be another application. For the latter, the synchronization of phonemes and very short video sequences has to be explored. This may be considerably easier than doing the same with a 3D model.

Bibliographie

[3D-01] 3D model-based facial analysis and synthesis for virtual conferencing. http://dewww.epfl.ch/~horbelt/PR/DEMO95/, 2001.

[AH00] The American Heritage Dictionary of the English Language. http://www.bartleby.com/61, 2000.

[BBF+01] L. Bagot, C. Bernard, A.-L. Fitère, C. Vincent, and M.-P. Virard. Internet mobile : 2000 milliards pour un triple pari. *Enjeux-les-Echos*, (166) :46–64, February 2001. http://www.lesechos.fr/publications/enjeux/enjeux.htm.

[BCL99] S. Battista, F. Casalino, and C. Lande. MPEG-4 : A multimedia standard for the third millenium, part 1. *IEEE Multimedia Magazine*, 6(4) :74–83, 1999.

[BCL00] S. Battista, F. Casalino, and C. Lande. MPEG-4 : A multimedia standard for the third millenium, part 2. *IEEE Multimedia Magazine*, 7(1) :74–83, 2000.

[Ber97] Berkeley Multimedia Research Center, http://bmrc.berkeley.edu. *Berkeley MPEG-1 Video Encoder User's Guide*, 1997.

[BFY98] M.J. Black, D.J. Fleet, and Y. Yacoob. Framework for modeling appearance change in image sequences. In *Proc. of the 6th IEEE Conf. on Computer Vision*, pages 660–667, Mumbai, India, January 1998.

[BI98] A. Blake and M. Isard. *Active Contours*. Springer-Verlag, Berlin Heidelberg New York, 1998.

[BJ96] M.J. Black and A.D. Jepson. Eigen tracking : Robust matching and tracking of articulated objects using a view-based representation. In ECCV'96 Conf. Proc. [ECC96], pages 329–342. Volume I.

[Bre99] L. Bretzner. *Multi-Scale Feature Tracking and Motion Estimation*. PhD thesis, Kungl Tekniska Högskolan, October 1999.

[CB90] G. Casella and R.L. Berger. *Statistical Inference*. Wadsworth & Brooks/Cole, Pacific Grove, CA, 1990.

[CBC97] J.L. Crowley, F. Bérard, and J. Coutaz. Multi-modal tracking of faces for video communications. In *Proc. of the 16th IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pages 640–645, Puerto Rico, June 1997. http://www.computer.org/proceedings/cvpr/7822/7822toc.htm.

[CCB00] J.L. Crowley, J. Coutaz, and F. Bérard. Things that see. *Communications of the ACM*, 43(3) :54–64, March 2000.

[CCBS97] J. Coutaz, J.L. Crowley, F. Bérard, and D. Salber. Eigenspace coding as a means to support privacy in computer mediated communication. In *Proc. of the 6th IFIP Conf. on Human-Computer Interaction*, pages 640–645, Sydney, July 1997.

[Cha95a] L. Chariglione. The development of an integrated audiovisual coding standard : MPEG. *Proc. of the IEEE*, 83(2) :151–157, February 1995.

[Cha95b] L. Chariglione. MPEG : A technological basis for multimedia applications. *IEEE Multimedia Magazine*, 2(1) :85–89, Spring 1995.

[Cla00] R. Clark. An Introduction to JPEG2000 and Watermarking. `http://www.jpeg.org/JPEG2000.htm`, 2000.

[CS99] J.L. Crowley and K. Schwerdt. Robust tracking and compression for video communication. In ICCV'99 Workshop Proc. [ICC99].

[CT65] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19 :297–301, April 1965.

[CTI01] Compression Technologies Inc. home page. `http://www.cinepak.com`, March 2001.

[CWS98] J.L. Crowley, F. Wallner, and B. Schiele. Position estimation using principal components of range data. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, May 1998.

[CWT00] T.F. Cootes, K. Walker, and C.J. Taylor. View-based active appearance models. In FG'2000 Conf. Proc. [FG'00], pages 227–232.

[Deu96] P. Deutsch. *GZIP file format Specification version 4.3*. InterNIC, `http://www.cis.ohio-state.edu/htbin/rfc/rfc1952.html`, May 1996.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B39 :1–38, 1977.

[DVD01] DVD-Forum. `http://www.dvdforum.org`, March 2001.

[ECC96] *Proc. of the 4th European Conf. on Computer Vision*, volume 1065 of *Lecture Notes in Computer Science*. Springer, April 1996.

[ECC98] *Proc. of the 5th European Conf. on Computer Vision*, volume 1406 of *Lecture Notes in Computer Science*, Freiburg, Germany, June 1998. Springer.

[Eis00] P. Eisert. *Very Low Bit-Rate Video Coding using 3-D Models*. Doktorarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg, November 2000. `http://www.nt.e-technik.uni-erlangen.de/~eisert/publications.html`.

[EWG00] P. Eisert, T. Wiegand, and B. Girod. Model-Aided Coding : A New Approach to Incorporate Facial Animation into Motion-Compensated Video Coding. *IEEE Trans. on Circuits and Systems for Video Technology*, 10(3) :344–358, 2000.

[FG'00] *Proc. of the Fourth IEEE Int. Conf. on Automatic Face and Gesture Recognition*, Grenoble, France, March 2000.

[For82] E.W. Forgy. Cluster analysis of multivariate data : efficiency vs. interpretability classifications. *Biometrics*, 21 :768–769, 1982.

[Fra81] L.E. Franks. *Signal Theory*. Dowden & Culver, Inc., Stroudsbourg, PA, revised edition, 1981.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2nd edition, 1990.

[Fur94] B. Furht. Multimedia systems : An overview. *IEEE Multimedia Magazine*, 1(1) :47–59, Spring 1994.

[GA96] J.-L. Gailly and M. Adler. *DEFLATE Compressed Data Format Specification version 1.3*. InterNIC, `http://www.cis.ohio-state.edu/htbin/rfc/rfc1951.html`, May 1996.

[Gal91] D. Le Gall. MPEG : A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4) :46–58, April 1991.

[Gif90] A. Gifi. *Nonlinear Multivariate Analysis*. John Wiley & Sons, Inc., West Sussex, U.K., 1990.

[GOSC00] C. Le Gal, A.E. Ozcan, K. Schwerdt, and J.L. Crowley. A sound magic board. In *Proc. of the Third Int. Conf. on Multimodal Interfaces*, Beijing, China, October 2000.

[GR94] K.L. Gong and L.A. Rowe. Parallel MPEG-1 video encoding. In *Proc. of the Picture Coding Symposium*, Sacramento, CA, September 1994. `http://www.bmrc.berkeley.edu/frame/resources`.

[GSM01] GSM World. `http://www.gsmworld.com`, 2001.

[Har28] R.V.L. Hartley. Transmission of Information. *Bell Systems Technical Journal*, page 535, July 1928.

[HHD98] I. Haritaoglu, D. Harwood, and L.S. Davis. W^4S : A Real-Time System for Detecting and Tracking People in $2\frac{1}{2}$D. In ECCV'98 Conf. Proc. [ECC98], pages 877–892. Volume I.

[HL96] T.S. Huang and R. Lopez. Computer vision in next generation image and video coding. *Lecture Notes in Computer Science*, 1035 :13–22, 1996.

[HN98] S. Huwer and H. Niemann. 2D-Object Tracking Based on Projection-Histograms. In ECCV'98 Conf. Proc. [ECC98], pages 861–876. Volume I.

[HTT00] HTTP home page. `http://www.w3.org/Protocols/HTTP`, May 2000.

[Huf52] D.A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proc. of the IRE*, 40(9) :1098–1101, September 1952.

[IB96] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In ECCV'96 Conf. Proc. [ECC96], pages 343–356. Volume I.

[IB98] M. Isard and A. Blake. ICONDENSATION : Unifying Low-level and High-level Tracking in a Stochastic Framework. In ECCV'98 Conf. Proc. [ECC98], pages 893–908. Volume I.

[ICC99] *IEEE Computer Society Int. Conf. on Computer Vision, Workshop on Face and Gesture Recognition*, Corfu, Greece, September 1999.

[ICV99] *Proc. of the First Int. Conf. on Computer Vision Systems*, volume 1542 of *Lecture Notes in Computer Science*, Las Palmas, Spain, January 1999. Springer.

[ISO93] ISO/IEC 11172-2 (MPEG-1). Coding of moving pictures and associated audio – part 2 : Video. Technical report, International Organisation for Standardisation, Geneva, Switzerland, http://www.iso.ch, 1993.

[ISO94] ISO/IEC 10918 (JPEG). Digital compression and coding of continuous-tone still images. Technical report, International Organisation for Standardisation, Geneva, Switzerland, http://www.iso.ch, 1994.

[ISO96a] ISO/IEC 13818-2 (MPEG-2). Generic coding of moving pictures and associated audio information : Video. Technical report, International Organisation for Standardisation, Geneva, Switzerland, http://www.iso.ch, 1996.

[ISO96b] ISO/IEC 14496-2 (MPEG-4). Information technology – coding of audio-visual objects – part 2 : Visual. Technical report, International Organisation for Standardisation, Geneva, Switzerland, http://www.iso.ch, 1996.

[ISO00a] ISO/IEC 14495 JTC1/SC29 WG1 (JPEG/JBIG). Information technology – Lossless and near-lossless compression of continuous-tone still images (JPEG-LS). Technical report, International Organisation for Standardisation, Geneva, Switzerland, http://www.iso.ch, 2000.

[ISO00b] ISO/IEC N2729 (MPEG-7). MPEG-7 content and objectives. Technical report, International Organisation for Standardisation, Geneva, Switzerland, http://www.darmstadt.gmd.de/mobile/MPEG7/Documents/N2729.html, 2000.

[ITU80] ITU-T Study Group VIII. *Recommendation T.4* : Standardization of group 3 facsimile terminals for document transmission. Technical report, Telecommunication sector of the International Telecommunications Union, Geneva, Switzerland, http://www.itu.int, 1980.

[ITU93] ITU-T Study Group XV. *Recommendation H.261* : Video codec for audiovisual services at px64 kbit/s. Technical report, Telecommunication sector of the International Telecommunications Union, Geneva, Switzerland, http://www.itu.int, March 1993.

[ITU94] ITU-R Study Group 11. *Recommendation BT.601-4* : Encoding parameters of digital television for studios. Technical report, Radiocommunication Sector of the International Telecommunication Union, Geneva, Switzerland, http://www.itu.int, 1994.

[ITU96] ITU-T Study Group XV. *Recommendation H.263* : Video coding for low bit rate communication. Technical report, Telecommunication sector of the International Telecommunications Union, Geneva, Switzerland, http://www.itu.int, 1996.

[JP99] T. Jebara and A. Pentland. Action reaction learning : Automatic visual analysis and synthesis of interactive behaviour. In ICVS'99 Conf. Proc. [ICV99], pages 273–292.

[KDN93] D. Koller, K. Daniilidis, and H.-H. Nagel. Model-Based Object Tracking in Monocular Image Sequences of Road Traffic Scenes. *Int. Journal of Computer Vision*, 10(3) :257–281, 1993.

[KHS99] V. Krüger, A. Happe, and G. Sommer. Affine real-time face tracking using a wavelet network. In ICCV'99 Workshop Proc. [ICC99].

[KP96] F. Kirouche and F. Planchon. Suivi d'Objets par le Son et l'Image pour la COmmunication Médiatisée. Technical report, Laboratoire CLIPS-IMAG, Equipe IIHM, Grenoble, France, June 1996. `http://www.imag.fr`.

[KSK90] G. Klinker, S.A. Shafer, and T. Kanade. A physical approach to Color Image Understanding. *Int. Journal of Computer Vision*, 4 :7–38, 1990.

[Lar92] LAROUSEE Encyclopedique en couleurs. Édition du Club France Loisirs, 1992.

[LBG80] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, COM-28 :84–95, January 1980.

[LCL+97] M.-C. Lee, W. Chen, C.B. Lin, C. Gu, T. Markoc, S.I. Zabinsky, and R. Szeliski. A layered video object coding system using sprite and affine motion model. *IEEE Trans. on Circuits and Systems for Video Technology*, 7(1) :130–145, February 1997.

[Lim90] Jae S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1st edition, 1990.

[Lio91] M. Liou. Overview of the p x 64 kbit/s Video Coding Standard. *Communications of the ACM*, 34(4) :59–63, April 1991.

[MPE01] MPEG-7 Frequently asked questions. `http://www.darmstadt.gmd.de/mobile/MPEG7/FAQ.html`, 2001.

[MW00] Merriam-Webster Collegiate Dictionary. `http://www.m-w.com/netdict.htm`, 2000.

[Nel92] M. Nelson. *The Data Compression Book*. M&T Books, New York, N.Y., 1992.

[NL99a] F. Nack and A.T. Lindsay. Everything you wanted to know about MPEG-7 : Part 1. *IEEE Multimedia Magazine*, 6(3) :65–77, 1999.

[NL99b] F. Nack and A.T. Lindsay. Everything you wanted to know about MPEG-7 : Part 2. *IEEE Multimedia Magazine*, 6(4) :64–73, 1999.

[OBK95] T. Özcelik, J. Brailean, and A.K. Katsaggelos. Image and video compression algorithms based on recovery techniques using mean field annealing. *Proc. of the IEEE*, 83(2) :304–316, February 1995.

[Pap91] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 3rd edition, 1991.

[Pen00] A. Pentland. Perceptual intelligence. *Communications of the ACM*, 43(3) :35–44, March 2000.

[Rea01] RealNetworks home page. `http://www.realnetworks.com`, March 2001.

[RY90] K.R. Rao and P. Yip. *Discrete Cosine Transform : Algorithm, Advantages, Applications*. Academic Press Inc., New York, 1990.

[SAG00] M. Störring, H.J. Andersen, and E. Granum. Estimation of the illuminant colour from human skin colour. In FG'2000 Conf. Proc. [FG'00], pages 64–69.

[Say95] K. Sayood. *Introduction to Data Compression, draft version*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1995.

[Say00] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2nd edition, 2000.

[SB91] M.J. Swain and D.H. Ballard. Color indexing. *Int. Journal of Computer Vision*, 7(1) :11–32, 1991.

[SC00] K. Schwerdt and J.L. Crowley. Robust face tracking using color. In FG'2000 Conf. Proc. [FG'00], pages 90–95.

[SCD98] K. Schwerdt, J.L. Crowley, and J.-B. Durand. Robustification of detection and tracking of faces. In *Proc. of the Joint TMR Workshop on Computer Vision and Mobile Robitics*, pages 155–161, Santorini, Greece, September 1998.

[Sha48] C.E. Shannon. A Mathematical Theory of Communication. *Bell Systems Technical Journal*, 27 :379–423,623–656, October 1948. Also available at `http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html`.

[Sik97] T. Sikora. MPEG digital video-coding standardsŤ. *IEEE Signal Processing Magazine*, 14(5) :82–101, September 1997.

[SW95] B. Schiele and A. Waibel. Gaze tracking based on face color. In *Proc. of the Int. Workshop on Automatic Face and Gesture Recognition*, pages 344–349, Zurich, Switzerland, June 1995.

[TG74] J.T. Tou and R.C. Gonzalez. *Pattern Recognition Principles*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1974.

[TLS93] W.B. Thompson, P. Lechleider, and E.R. Stuck. Detecting Moving Objects Using the Rigidity Constraint. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15 :162–167, February 1993.

[TP91] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1) :71–86, March 1991.

[UMT01] UMTS forum. `http://www.umts-forum.org`, 2001.

[Vid01a] Video sequence links source page. `http://www.av.it.pt/collaborators/personal/navarro/links.html`, February 2001.

[Vid01b] Video sequences source page. `ftp://dspftp.ece.ubc.ca/pub/tmn/qcif_source`, February 2001. sequences in QCIF format.

[VSC99] W.E. Vieux, K. Schwerdt, and J.L. Crowley. Face-tracking and coding for video compression. In ICVS'99 Conf. Proc. [ICV99], pages 151–161.

[Wal91] G.K. Wallace. The JPEG Still Picture Compression Standard. *Communications of the ACM*, 34(4) :30–44, April 1991.

[Wal97] F. Wallner. *Estimation de position d'un robot mobile par utilisation des composantes principales des données d'un capteur télémétrique laser*. PhD thesis, Institut National Polytechnique de Grenoble, October 1997.

[WAP00] WAP Forum home page. `http://www.wapforum.org`, May 2000.

[WNC87] I.H. Witten, R.M. Neal, and J.G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6) :520–540, June 1987.

[ZL77] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. on Information Theory*, 23(3) :337–343, May 1977.

[ZL78] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. on Information Theory*, 24(5) :530–536, September 1978.

Index

Résumé

Cette thèse présente une nouvelle technique pour la compression de données vidéo numériques, appelée le *Codage de Bases Orthonormales (CBO)*. Des algorithmes de vision par ordinateur, de compression de données, et d'identification de configuration sont combinés pour donner une méthode de codage en trois étapes. CBO recueille des informations sur le contenu d'une image sans utiliser de modèles. Au lieu de cela, il est basé sur l'apparence d'objets. Les techniques basées sur l'apparence utilisent des représentations orthonormales de l'espace de base des objets, habituellement dans l'espace propre, et exploitent les propriétés géométriques de ces représentations d'objet. Dans une séquence d'images d'un objet, chaque image est un point dans l'espace engendré par la base orthonormale utilisée. Une concentration sur un objet représente une normalisation des données visuelles d'entrée d'un objet principal. Étant donné les domaines d'application pour la compression vidéo, le visage d'un locuteur comme objet principal est un choix normal. Nous démontrons que CBO est une alternative valide aux techniques de compression vidéo conventionnelles. En fonction de la précision de la normalisation sur l'objet principal, le CBO montre une performance bien supérieure a celle des techniques conventionnelles.

Mots–clés

Compression vidéo, vision par ordinateur, apparence, Codage de Bases Orthonormales

TITRE

COMPRESSION VIDÉO FONDÉE SUR L'APPARENCE

Abstract

This thesis describes a new technique for the compression of digital video data, called *Orthonormal Basis Coding (OBC)*. Algorithms from computer vision, data compression, and pattern recognition are combined to form its three-step encoding scheme. OBC gathers information about image content without using models. Instead, it is based on appearance. Appearance-based techniques use orthonormal basis space representations of objects, usually in eigenspace, and exploit geometrical properties of these object representations. From a sequence of images of an object, each image contributes a point in the orthonormal basis space used. Focusing on one object from a sequence means that we need to normalize our video input data to one principal object. Given the application areas for video compression, a talking head as principal object is a natural choice. We show that OBC is a valid alternative to conventional video compression techniques. If the input video stream is well normalized to the principal object, OBC outperforms conventional compression techniques.

Key words

Video compression, computer vision, appearance, Orthonormal Basis Coding