UCL
Université
catholique
de Louvain

Ecole Polytechnique de Louvain
Laboratoire de Télécommunications
et
Télédétection

B - 1348 Louvain-la-Neuve
Belgique

Seamless Remote Browsing and Coarse-to-fine Compressed Retrieval using a Scalable Image Representation

Antonin Descampe

Thèse présentée en vue de l'obtention du grade de docteur en sciences de l'ingénieur

Composition du jury :

Benoit Macq (UCL/TELE) - *Promoteur*
Philippe Chevalier (UCL/IAG-CORE)
Jean-Didier Legat (UCL/DICE)
Christophe De Vleeschouwer (UCL/TELE)
Justus Piater (ULg, Belgique)
Pierre Vandergheynst (EPFL, Suisse)
Luc Vandendorpe (UCL/TELE) - *Président*

Mars 2008

Remerciements

Si une thèse est souvent décrite comme une aventure solitaire, il ne s'agit pas pour autant d'un travail d'ermite. J'ai eu la chance au cours de ces 4 années de recherche d'être entouré, tant professionnellement que personnellement, d'un grand nombre de personnes que je tiens à remercier chaleureusement. C'est avec beaucoup de reconnaissance que je leur dédie ce travail.

Je remercie tout d'abord mon promoteur, Benoit Macq, pour m'avoir encouragé dans la voie du doctorat, pour m'avoir accordé sa confiance sans faillle tout au long de mon parcours, et pour avoir su créer pendant ma recherche les opportunités et les impulsions qui m'ont fait avancer. Je remercie également tout particulièrement Christophe De Vleeschouwer : sa disponibilité, son écoute, ses commentaires éclairés et ses encouragements m'ont systématiquement remotivé quand j'en avais besoin et ont été réellement déterminants dans l'accomplissement de cette thèse. Merci aussi au FNRS d'avoir financé ces 4 années de recherche.

Les 6 mois que j'ai eu la chance de passer à Lausanne furent très enrichissants, tant professionnellement que humainement : merci à Pierre Vandergheynst de m'avoir si agréablement accueilli et intégré dans son équipe. Merci aussi à tous ceux qui peuplent les si bons souvenirs que je garde de ce séjour.

Merci également aux autres membres de mon jury, Philippe Chevalier, Justus Piater et Jean-Didier Legat, pour le temps qu'ils ont consacré à mon travail et pour leurs remarques constructives. Je remercie aussi Raphaël Marée pour sa promptitude à répondre à mes questions concernant les arbres de décision.

L'ambiance de travail est un facteur déterminant pour le doctorant qui tient à conserver (une partie de) sa santé mentale. Sur ce point, travailler au laboratoire TELE m'a non seulement permis de rester sain d'esprit mais m'a aussi donné l'occasion de rencontrer des personnes de grande qualité. Merci à tous et en particulier à Damien, Pedro, FOD, Jean-Ju, Jacek, Jerôme, Jef, sans oublier mes chers collègues de bureau Remy, Teh et Suzanne.

Ces 4 dernières années auront aussi été l'occasion de co-écrire le début d'une belle histoire : celle d'intoPIX. Je tiens à remercier ici Jean-François,

Gaël, François, et François-Xavier pour avoir ensemble rendu cette aventure possible. Puisse la suite de cette histoire continuer à être aussi passionnante! Un merci tout particulier à François-Olivier qui a fort heureusement autant de surnoms que de statuts dans mon réseau social.

Merci enfin à tous ceux et celles qui me rappellent, chacun à leur manière, que le bonheur n'est réel que s'il est partagé. Je pense d'abord à ma famille qui m'a supporté (dans les deux sens du terme) de manière inconditionnelle tout au long de mon parcours. Merci aussi à tous mes amis, avec une pensée particulière pour la bande de théâtreux qui m'a confirmé que gesticuler sur des planches est un complément parfait à un travail cérébral.

Last but not least, merci à Anouar Brahem pour avoir quotidiennement accompagné mon travail de la beauté de sa musique.

Abstract

In today's information society, two important trends can be observed : (1) the digital universe -particularly still or moving images- is growing exponentially and (2) computer networks are becoming more heterogeneous. The former raises the need for tools to guide the user within large data spaces, while the latter requires a representation of information that is sufficiently flexible to adapt itself optimally to various network constraints or user requirements. In this thesis, we focus on image information and address these needs by investigating three complementary aspects of a remote image server.

First, a scalable image representation, namely JPEG 2000, is studied. This compression standard organizes the image information in such a way that almost any spatial area can easily be extracted at any resolution and bit-depth. This scalability implies a high computational load that may require a hardware implementation when dealing with real-time constraints. The main characteristics of such hardware architecture are also presented.

Secondly we consider the issues of scheduling and caching JPEG 2000 data in client/server interactive browsing applications under memory and channel bandwidth constraints. We evaluate several strategies to schedule data packets that are likely to become relevant to expected future Windows-of-Interest ("pre-fetching" techniques), and show how the system reactivity can be improved.

Finally, we investigate how to perform categorization and retrieval tasks directly in the compressed domain. Thanks to the JPEG 2000 scalability the discriminant level of a compressed image characterization is directly related to the amount of extracted data. Taking this finding into account, an original representation, called integral volumes, is introduced to store such characterization. Combining integral volumes with random decision trees, we propose a JPEG 2000 image classifier that achieves performances similar to the best uncompressed image classification results obtained on several freely available databases. Eventually, to address the image retrieval problem, a cascade of such classifiers is used, enabling a cost-optimized coarse-to-fine retrieval process.

Contents

List of Figures

Introduction 1

It is a press, certainly, but a press from which shall flow in inexhaustible streams... Through it, God will spread His Word. A spring of truth shall flow from it: like a new star it shall scatter the darkness of ignorance, and cause a light heretofore unknown to shine amongst men.

Gutenberg.

The library is a labyrinth ?

William of Baskerville,
in *The Name of the Rose* [33], p.157.

1.1 Motivations and context of this work

The invention of movable type printing, attributed to German goldsmith Johannes Gutenberg in 1439, was a revolution in Europe: spreading information became much easier and abbeys were no more the only places to gain access to knowledge. Although we should let History tell, the beginning of the digital era and the development of the Internet appear to bring to our world as important changes than the printing press did [86]. The biggest of these changes is that the binary representation of information, whether visual, audio or textual, allows for the first time to dissociate the information from its original medium (film, vinyl, paper, etc). Until now, the way we were storing, accessing, protecting or distributing information was intrinsically linked with this physical medium. But now, as information is dematerialized in a stream of 0's and 1's [40] and as computer networks can efficiently process it, almost all tasks related to information management have to be re-invented. A good illustration of this is the obsolescence

of the music sales business model, based on the sale of a medium (tapes, CDs) that includes author's retribution. Such model is obviously not valid anymore, as evidenced by the amount of songs daily shared on the Internet. In this context of digital revolution, two trends can in particular be observed, and are the main motivations of the present work.

The Big Bang of the digital universe

First, the amount of digital data increases drastically. YouTube, this online video hosting company created 3 years ago, streams 100 million videos a day. Studies have shown that around a billion songs a day are shared over the internet in MP3 format. 64 trillion bits a day are transmitted from London's traffic surveillance cameras to data centers[1]. Usual consumer digital cameras "spend" nowadays around 10 million pixels on a single picture. Examples are numerous. This increase is due on one hand to the fact that more and more analogical measurements from the real world are digitized and then replicated for purposes of storage, transmission or any other processing. On the other hand, as accuracy requirements become more constraining, and as sampling structures get finer, a given analogical measure results in even more digital data. Recently, IDC, a company specialized in studies on information technology markets trends, released a report [39] on this "expanding digital universe". Amongst the key findings, one will mention:

- From 2006 to 2010, the information added annually to the digital universe will increase more than 6 folds, from 161 to 988 exabytes[2].
- Images, captured by more than 1 billion devices in the world, from digital cameras and camera phones to medical scanners and security cameras, comprise the largest component of the digital universe.
- By 2010, 70% of the digital universe will be created by consumers (taking pictures, talking on the phone, working at home computers, etc). However, organizations (business of all sizes, agencies, governments, etc) will be responsible for the security, privacy, reliability and compliance of at least 85% of that same digital universe.

[1] These three examples come from [39].
[2] 1 exabyte = 1 billion of gigabytes = 2^{60} bytes.

These findings and predictions give us some hints on the kind of tools that should be developed to manage this growing world.

First of all, *compression is still relevant.* As storage devices are getting cheaper and bandwidths broader, one could reasonably question the actual need to compress data. However, analysts show in [39] that the growing pace of the digital world is actually greater than the one of the global storage capacity. Being able to store all these images, movies and music in a raw format remains therefore an utopia, and smart algorithms are still needed to efficiently format information. Moreover, as we shall see below, compression becomes only one requirement among others in the design of such algorithms.

Second, *solutions are needed to avoid a digital chaos.* Or at least to find our way in this chaos. Having access to terabytes of data is useless if we cannot efficiently find the information we are looking for. Regarding textual information, the problem is easy (in comparison with visual or audio information) and has already been smartly and successfully solved by companies like Google. But as indicated above, the largest part of the data is still or moving images. And the only way to deal with this increase of the amount and diversity of image data is the development of techniques to guide and assist the user within these large data spaces. Unfortunately, as "a picture is worth a thousand words", this challenge is a thousand times more complex than with textual information (due in particular to the so-called semantic gap [127]). Consequently, a single fully-automatic algorithm might not be optimal to address the complexity and the diversity of the problem. On the contrary, several complementary techniques will likely have to be combined to produce good results: meta-data generation, content-based search, efficient browsing techniques, relevance feedback, etc.

Finally, *organizations are the custodians of users data.* Although individuals will create most of the future digital information, they will not simply keep it on their personal hard drives. The data will be touched at some point by an organization: in a data center, an images server, a hosting site or a remote back-up system. These organizations therefore need appropriate and efficient infrastructures to manage, host, secure, and transmit this digital universe.

Scale to your needs

The second trend concerns the access to this huge amount of digital data. The digital universe is growing, for sure, but its access roads are also getting more heterogeneous. Development and diversification of computer

networks allow us to require access to any information at any time in any place and from any device. Mobile phones, PDAs, GPS, laptop and desktop computers, etc: all are different kinds of devices with different resources, notably in terms of bandwidth and display size. And when delivering information, these resources have to be taken into account for obvious usability reasons. A first solution would be to store on a server several quality levels of the same information so as to be able to answer appropriately the requests from different devices. However, a smarter, more efficient, and less space-wasting solution is to find a single representation of information that would be sufficiently *flexible* to adapt itself optimally to various network constraints, terminal devices or user requirements.

Moreover, while an ever larger fraction of people from industrialized countries are getting a broadband access whatever their location, this is not at all the case for everybody, particularly in developing countries. This gap between the technologically rich and poor is called the *digital divide* [82], which is nothing more than a new instantiation of the social divide. Taking this network heterogeneity into account and ensuring that most people can efficiently access information whatever their resources, is therefore not only a smart move on a diversifying market but it also brings social benefits.

If we now focus on images, the "flexible representation" mentioned above means that a single format should be able to optimize the visualization under the constraint of available resources. This property is also often called *scalability* and is lacking in several image compression standards, such as JPEG. In the DCT[1]-based approaches (JPEG, MPEG) any change of resolution or any new rate-quality trade-off is reached through decoding and re-encoding of the images. On the contrary, the latest image compression standard from the JPEG committee, called JPEG 2000 [6], enables inherently such scalability: according to the available bandwidth, computing power and memory resources, different resolutions and quality levels can be extracted from a single bit-stream. Moreover, as we shall see in this work, the scalability of an image representation can not only be used to adequately adapt itself to available resources but it can also efficiently be exploited in retrieval tasks. Additionally, it is interesting to point out that this concept of scalability is actually a very *natural* concept. The analysis of basic processes involved in animal (or human) vision has shown that the eye automatically makes a multi-scale decomposition of the observed image, even before any intervention of the brain (*early visual system*). In

[1]Discrete Cosine Transform, a transformation of the image that reveals its frequency content but looses the spatial information.

front of a beautiful "postcard" mountains landscape, we will therefore first get a global coarse view of the whole landscape (mountains, valleys) before focusing on details like creeks or fir trees distribution[1].

1.2 Goals and thesis organization

If we summarize the key points of the previous section, we can basically identify two needs: (*i*) the need for tools able to guide and assist a user when "traveling" in the growing digital universe, (*ii*) the need for a compressed and flexible representation of information. Moreover, we have seen that images (still or moving) will represent the largest part of this information. In this context, our work focuses on

> **the development of systems using flexible bit-streams to browse and search information within high resolution images.**

Practically, we consider the existence of a remote image server and we investigate three aspects of such server:

- the representation used to efficiently store the images,
- techniques to enable a smooth and seamless remote navigation in these images,
- algorithms allowing to retrieve (parts of) images similar to areas of interest submitted by the user.

We explicitly consider high resolution images (also called *mega-images*) but it is not a restrictive choice: smaller images or even moving images (videos) could also be handled. In this last case, our system would adequately be combined with the approach proposed in [76], that generates a video summary in the form of a single high resolution mosaic of key-frames.

Regarding the flexible representation used for the images, the above mentioned JPEG 2000 algorithm has been chosen. This choice is motivated by several reasons: it is an efficient compression algorithm, it has the status of International Standard (and is therefore royalty-free), and a growing number of professionals show a real interest in its use[2]. But most importantly, this algorithm is highly scalable, and this property lies at the

[1] Thanks to Laurent Jacques [47] for this nice illustration.

[2] See next chapter for a more detailed description of the JPEG 2000 adoption.

basis of all results obtained in the proposed browsing and retrieval techniques. As such, the implementation of our work deeply depends on the JPEG 2000 standard, but the underlying concepts only assume the scalability of the image representation format. Should JPEG 2000 be dethroned by a "JPEG 3000", these concepts would therefore remain valid as long as this new algorithm remains scalable. And we believe scalability is such a natural and powerful property that it would necessarily be the case.

Many applications could draw benefit from such remote image server. Among them, we could cite at least medical applications, audio-visual archives, and remote sensing, as professionals from these fields have explicitly shown us their interest in this kind of tool.

In the following, each of the three aspects enumerated above is investigated in a separate chapter, that we now briefly introduce. We invite the reader to refer to Figure 1.1 to get a visual illustration of the thesis organization.

In **Chapter 2**, we describe in more details the JPEG 2000 image compression standard. Both a functional and a technical description are given, which are the prerequisites to fully understand the two next chapters[1]. Then, the main algorithm drawback -complexity- is highlighted, together with its implications in terms of implementation. When a high-processing rate is required to encode or decode images (to meet a real-time constraint for example), a hardware implementation might be considered. The main characteristics and performances of such hardware architecture are presented. This system, that we partly developed in the context of this thesis, was initially designed for Digital Cinema. However, an adaptation is under development by a spin-off so that it can be exploited in heavily loaded image servers like the one considered here.

Chapter 3 deals with JPEG 2000 remote browsing. It considers the issues of scheduling and caching JPEG 2000 data in client/server interactive browsing applications, under memory and channel bandwidth constraints. It analyzes how the conveyed data has to be selected at the server and managed within the client cache so as to maximize the reactivity of the browsing application. In particular, several *pre-fetching* techniques are proposed and compared. Pre-fetching consists in sending data in advance to the client because it is likely to be requested in a near future.

[1]It should be noted however that each chapter is as self-contained as possible and a reminder of important concepts is always given when necessary.

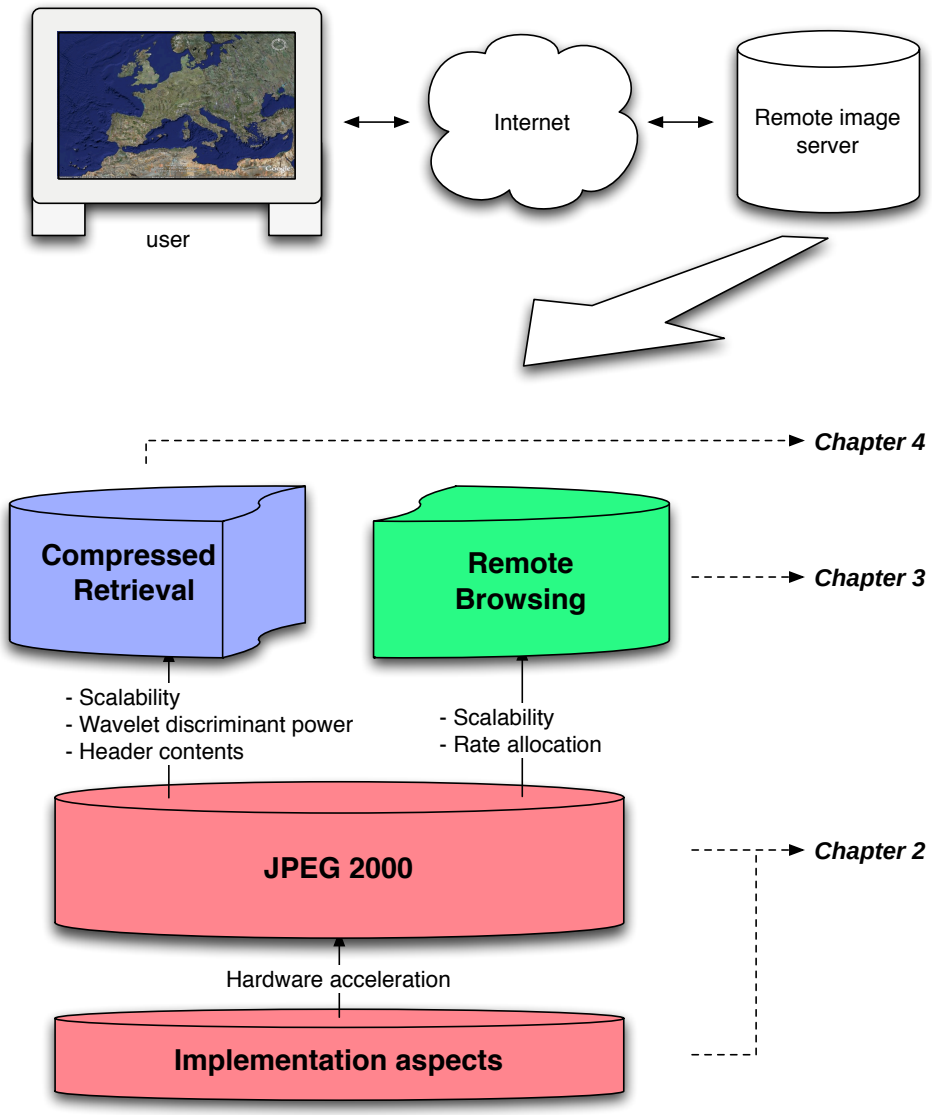


Figure 1.1: *Thesis organization. Three aspects of a remote image server are studied: the flexible representation of images (and some implementation issues), a remote browsing module, and compressed retrieval techniques. The two last components rely on the first one in several ways, as indicated in the figure.*

In **Chapter 4**, we address the issues of analyzing and classifying JPEG 2000 code-streams, which can be useful in any application that involves (semi) automatic image categorization or retrieval tasks. Being able to mine into compressed code-streams is obviously of great interest when dealing with a huge amount of data, as heavy decoding operations can be avoided. As we shall see in this chapter, such compressed retrieval operation in the JPEG 2000 domain is not only possible but actually also very efficient, as its scalability allows for a coarse-to-fine approach that guarantees a good trade-off between computational load and performances.

In the **last chapter**, we summarize the contributions of each chapter and present several future research directions. In particular, synergies could be exploited between hardware aspects, browsing and retrieval. Although they are presented separately in each chapter, these components have indeed to be thought globally, as making part of a unified framework whose cement is the JPEG 2000 algorithm.

JPEG 2000, a flexible image compression standard

2

Introduction to a flexible image compression standard: functional, technical, and implementation aspects of JPEG 2000

This chapter presents in more details the JPEG 2000 image compression standard. First, a functional description explains why this algorithm can be considered as flexible. The various scalability levels are illustrated and other useful features are detailed. Then, a technical description is given: this rich features set is made possible thanks to the combination of a discrete wavelet transform and a context-based adaptive binary arithmetic coder. The main drawback of this algorithm is that it requires a high computational load, in comparison with other still image compression standards, like JPEG. This leads to several implementation issues when addressing applications with real-time constraints, like Digital Cinema. These issues are described in the last part of this chapter, together with a brief presentation of an FPGA-based hardware architecture of a JPEG 2000 decoder that we partially developed in the context of our work. The implementation aspects mentioned in this chapter have been discussed in two conference papers [17, 18] and a journal paper [21], and has been successfully commercialized by the intoPIX *spin-off as an IP (Intellectual Property) [20].*

2.1 Introduction

As explained in the previous chapter, the development of digital technologies has drastically modified the requirements and constraints that a good image representation format should meet. In the early development of digital imaging applications, the requirements were to achieve a good compression efficiency while keeping the computational complexity low. This has lead in 1992 to the standardization of the JPEG format, which is still very widely used today. Over the years however, many things have evolved: more computing power is available and the development of Internet has required image representation formats to be more flexible and network-oriented.

In this context, the JPEG committee worked on a more scalable, but also more complex, image compression algorithm, called JPEG 2000. It became an International Standard from the ISO[1] in 2001. Since then, it is being adopted on a growing number of markets. However, its high complexity (in comparison with JPEG) has somewhat slowed down its adoption. Moreover, several other compression formats have been developed and JPEG 2000 has to "share" its potential users with them. JPEG for instance is still popular in consumer pictures handling applications. There is also HD Photo [105], a new compression format from Microsoft, that reveals itself to be simple but better than JPEG and which could replace it on the consumer market. On the video side[2], MPEG-2, MPEG-4 (also called H.264-AVC [120]) or soon SVC (Scalable Video Coding [102]) will on their side remain well suited candidates for HD-DVD's, tv-on-demand and all the end-user segment of broadcast. JPEG 2000 is therefore focusing on other markets, which are mainly professional markets that require a high quality level and intrinsic scalability. Among them, we should cite Digital Cinema (DC) that has adopted JPEG 2000 as its official standard. Broadcast also shows a growing interest for it, at least on its contribution[3] segment [106]. Other potential applications are Medical Imaging [36, 116], Remote Sensing [126], and audio-visual archives [48].

Since its release in 2001 (and even already before), JPEG 2000 gave rise to a huge amount of articles, from technical descriptions to short and popularized overviews or market-flavoured presentations. Among others,

[1] International Organization for Standardization.

[2] JPEG 2000 could indeed also be used as a video compression algorithm, without however exploiting the temporal redundancy.

[3] The *contribution*, in broadcast, is the transfer of high quality versions of the distributed media between different broadcast providers.

we should obviously cite the official document provided by the JPEG committee [6]. As it defines the standard, it is logically exhaustive, but not easily readable. More accessible are the papers [92] and [104] that give a good technical and functional overview of the standard. A detailed and exhaustive description, together with deep explanations of the underlying concepts, is given in [112], a book written by two "fathers" of JPEG 2000, D. Taubman and M.W. Marcellin.

In the remainder of the chapter, we give in Sections 2.2 and 2.3 a brief description of the standard. In Section 2.4, we address the main implementation issues and present the performances reached by an hardware implementation that we partially developed and that is now being commercialized.

2.2 Functional description

The JPEG 2000 image compression standard has been split in several parts, the first one being the core coding system. We first introduce the features of this baseline standard, before briefly describing some of the enhancements provided by the other parts.

2.2.1 Basic features

The core coding system defined in JPEG 2000 part 1 defines a compression algorithm and a code-stream syntax. One of the main requirements of this system was to define a single algorithm that could be used in many different applications and use cases. Indeed, the JPEG 2000 authors wanted to avoid the main drawback of JPEG: its lack of modularity due to the existence of too many incompatible modes[1]. With this in mind, they designed JPEG 2000 in a very flexible way. Any JPEG 2000 compliant code-stream inherently benefits from many useful features, much beyond what the JPEG core system was proposing. Here are the most important ones.

Handling of a wide variety of images

JPEG 2000 can handle many different kinds of images: from small ones to mega-images, with 1 up to 16 384 components, each one with up to 38 bits per pixel.

[1] JPEG has 44 modes: most of them are designed for very specific applications and many JPEG codec only know a few of them.

High compression efficiency

As detailed in [100], the rate-disortion curve achieved by a JPEG 2000 compressed image is very good, compared to other coding schemes. This is largely due to the use of an adaptative arithmetic entropy coding (described in Section 2.3), which is more efficient than a Huffman entropy coding system used in JPEG or MPEG standards.

Moreover, at low bit-rates, the coding artefacts that appear are usually less annoying for the human visual system than the blocking effects commonly found when compressing images with DCT-based standards. Indeed, these "DCT-blocks" manifest themselves as horizontal and vertical straight lines -virtually unknown in nature- while wavelets artefacts have more natural curved shapes and are therefore less visible. Fig. 2.1 illustrates this high compression efficiency by comparing a JPEG and a JPEG 2000 image at two different compression ratios. As we see, the visual quality of JPEG 2000 is better.

Lossy and lossless compression

An important advantage of JPEG 2000 in comparison with other standards is to enable both a lossless and a lossy compression, using the same algorithm. It further increases the genericity of the standard as the same compression algorithm can be used in a wide variety of applications, from medical diagnosis (that requires lossless compression) to consumer images distribution (that can afford losses). Moreover, within a given application, a losslessly compressed image can be used to extract lossy versions of it, as explained in the next paragraph.

Scalability

What fundamentally makes JPEG 2000 a flexible image compression algorithm is its ability to seamlessly adapt itself to the user needs and available resources, which is often called *scalability*[1]. It means that within a JPEG 2000 code-stream, the image data is distributed in such a way that one can easily extract the exact portion of information that corresponds to his needs (in terms of image area, resolution, etc) and that takes into account the available resources (in terms of bandwidth, display, etc). This scaling can be done at four different levels (see Fig. 2.2):

[1] In a more general sense, scalability is defined as *the ability of a computer application or product (hardware or software) to continue to function well when it (or its context) is changed in size or volume in order to meet a user need.* `http://www.whatis.com`.

Figure 2.1: *Subjective comparison between JPEG (left) and JPEG 2000 (right) compression efficiency. First row: original image. Second row: compression ratio of 170. Third row: compression ratio of 65.*

1. *the resolution*: the wavelet transform (briefly described below) reorganizes the information in so-called *resolution levels*. Starting from the original full resolution, each successive level divides the image width and height by 2, leading to a 4-times smaller version of the image from the previous level.

2. *the bit-depth*: data is entropy-encoded on a "per bit-plane" basis. This means that most significant bits of all pixels are encoded before less significant ones. By grouping encoded bits of equal significance, we obtain *quality layers*. The first quality layer gives a coarse version of the image (only the most significant bits of each pixel are used), which is further refined by subsequent quality layers. Note that the lossless compression ability mentioned above corresponds to a case where all bits from each wavelet coefficient are kept (no quantization) and entropy-encoded[1].

3. *the spatial location*: any spatial area inside an image can easily be extracted from a JPEG 2000 code-stream without having to process other parts of this image.

4. *the color component*: when an image is made of several components (like in color images or, more generally, multi-modal images), each component is coded independently and can therefore be extracted separately.

Practically, a JPEG 2000 code-stream is made of a succession of packets, each one containing the information related to a certain quality layer from a certain resolution, in a certain spatial location of one of the image components. This scalable way of organizing information inside the code-stream provides several useful features, like:

- *Random access*: the scalability levels can be mixed and used all together. It is therefore possible for example, starting from a compressed color image, to extract the two first quality layers from the lowest resolution of the upper left corner of component Y.

- *Single compression, multiple decompression*: a single JPEG 2000 code-stream can be decompressed in a lot of different ways, depending on the user needs or available resources. Fig. 2.2 illustrates this

[1]To be accurate, we have to mention that a lossless compression also implies to use reversible color and wavelet transforms, as described below.

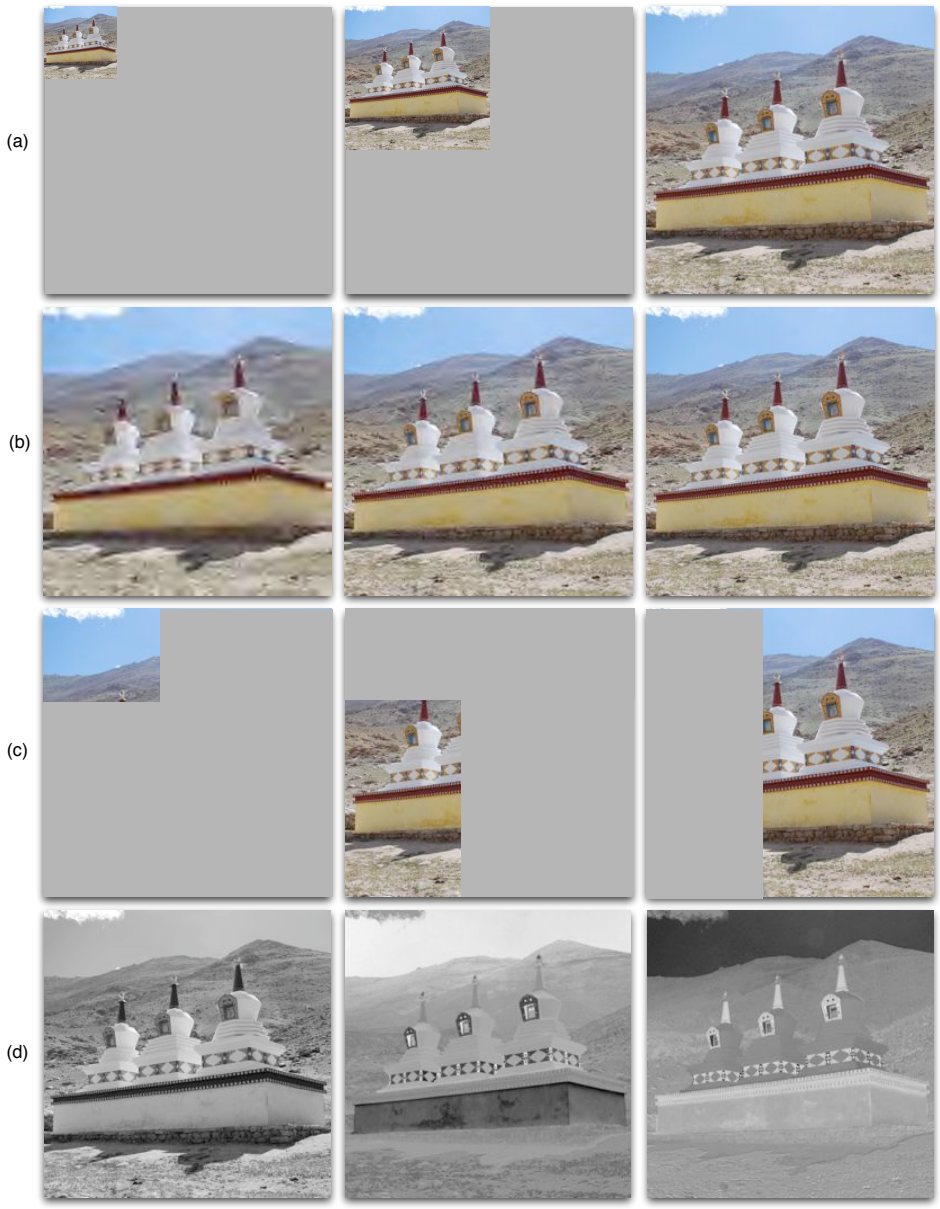


Figure 2.2: *Scalability in JPEG 2000. Starting from a JPEG 2000 codestream, image information can be extracted in several different ways: (a) by resolution, (b) by quality layer, (c) by spatial location, (d) by component (here: Y, Cb and Cr components, represented with a grayscale colormap).*

as all twelve versions of the image are extracted from the same code-stream. This feature is very useful in streaming applications for example: based on a single version of the images stored on a server, various quality levels can be delivered to different clients, according to their interest and available resources.

- *Various kinds of progressive transmission*: packets can be ordered in several ways, leading to different kinds of progressive transmissions when the code-stream is sent over a network. We can for instance decide to transmit all information related to the lowest quality layer first. Moreover, switching from a progression order to an other does not require any decompression operation: a simple packet re-ordering is sufficient.

As we shall see in Chapter 3 and 4, these features are of great interest for browsing or retrieval applications. When remotely browsing an image, the user can pan and zoom while limiting the transmission to the packets related to his moving window-of-interest. In a retrieval context, we will see that the JPEG 2000 scalability enables a coarse-to-fine approach that allows to automatically retrieve areas of interest while limiting the required computational load.

Region-of-Interest coding

When processing an image, JPEG 2000 allows to specify certain areas that have to be compressed with a better quality, i.e. with a smaller compression ratio. Indeed, it can happen that all regions of an image do not have the same importance and that some areas have to be preserved from any compression artefact. With this mechanism, it is still possible for example to get a globally highly compressed image while having some areas that are almost losslessly compressed.

Error Resilience

When transmitting an image over a network, some errors or packet losses can occur. They decrease the overall quality of the displayed image or even make the received stream unusable. JPEG 2000 has several built-in mechanisms that avoid error propagation by enabling re-synchronization operations during the code-stream decoding. In this way, a single error cannot alter the whole image but will rather damage only a small part of it.

Moreover, the scalability levels detailed above can be adequately combined with a smart error protection strategy. Indeed, unequal error protection can easily be implemented by inserting stronger forward error correction (i.e. more redundancy) in parts of the code-stream that are of great importance (headers, low resolutions or low quality layers). By doing so, we increase the probability to at least receive a low quality version of the image, without saturating the bandwidth. This kind of mechanism is used for instance in a set of tools specifically designed for transmission of JPEG 2000 images over wireless networks, called JPEG Wireless (JPWL).

2.2.2 Additional features

The features described above are all available in the JPEG 2000 core coding system, which is the part 1 of the standard. In addition to this, several extensions have been defined, each one being described in a specific part. For the sake of completeness, we provide in Table 2.1 a brief description of the different additional parts of the JPEG 2000 standard.

Among these extensions, the part 9, JPIP [89], that standardizes the communication between a client and a server for the transmission of JPEG 2000 images is of high interest in the context of our work. Indeed, in Chapter 3, we address the problem of JPEG 2000 remote browsing and assume that a similar protocol is available to forward client requests to the server and send back image data to the client. Based on this assumption, we can focus on the issues related to the selection and caching of conveyed image data units in order to improve the browsing system reactivity.

Another standardization effort that is worth mentioning is called JPSearch [54]. It has begun in 2004 (it is currently still under way) and is not part of the above mentioned extensions as it is not intended to be restricted to JPEG 2000 images. JPSearch aims to provide a standard for interoperability for still image search and retrieval systems. This includes the specification of a meta-data format, a common query language, etc. In Chapter 4, we introduce a JPEG 2000 retrieval system and we assume the existence of such JPSearch query language, so that we can focus on the performances and accuracy of our system.

As we see, JPEG 2000 is completed with several additional tools that facilitate its use in various application frameworks. The existence of such tools for browsing (JPIP) and retrieval (JPSearch) applications allows us to rely in both cases on a strongly formalized framework. It also comforts us in the interest raised by such applications.

Part	*Name*	*Description*
1	J2K, JP2	JPEG 2000 core coding system.
2	JPX	Algorithm extensions (like user-defined wavelet transform).
3	MJ2	Motion JPEG 2000: encapsulation format to store a set of successive JPEG 2000 frames (like a movie).
4	-	Compliance procedures: test-images sequences in order to verify the correct behavior of a given codec.
5	-	Reference software.
6	JPM	Compound image format: extension enabling to efficiently code documents containing both text and images.
7	-	Hardware implementation interoperability (*abandoned*)
8	JPSEC	Tools in order to allow applications to generate, consume, and exchange secure JPEG 2000 codestreams.
9	JPIP	JPEG Interactive Protocol: client-server protocols for remote browsing of JPEG 2000 images.
10	JP3D	Extensions for 3D-data and floating point data (volumetric imaging)
11	JPWL	Additional data structures and error protection techniques for transmission of JPEG 2000 images on wireless networks.
12	-	Base Media File Format: common encapsulating format for JPEG 2000 and MPEG4.
13	-	Baseline encoder guidelines.

Table 2.1: *The different parts of the JPEG 2000 standard. Concerning the part 3 "MJ2", note that although a movie can be stored with such format, it does not mean that motion compensation is done: frames are coded separately and encapsulated in a single file.*

2.3 Algorithm overview

In this Section, concepts and vocabulary useful for the understanding of the next chapters are presented. For more details, [6] or [92] should be referred to. Readers familiar with the JPEG 2000 algorithm should skip this section. Fig. 2.3 presents the successive coding steps that have to be applied to an image in order to obtain a JPEG 2000 code-stream. Each step is briefly described below.

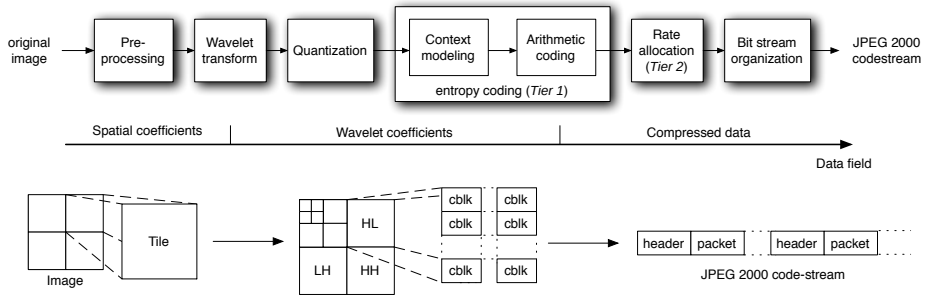


Figure 2.3: *The JPEG 2000 coding steps. Spatial coefficients (aka* pixels*) are first transformed in wavelet coefficients. Then, the different subbands from each resolution level are divided in code-blocks that are independently entropy-coded. Eventually, entropy-coded data is distributed in packets that compose the final JPEG 2000 code-stream.*

Pre-processing

Three operations can be done during this first step. First of all, the image can be split into rectangular blocks called tiles, that will be compressed independently. This is particularly useful for applications with limited memory resources. Then, if pixels are represented by unsigned values, they are shifted to get the corresponding signed values. Eventually, in case of an image made of several components, an inter-component decorrelation can be applied. For instance, in part 1, the RGB to YCbCr transformation is specified. Such decorrelation increases the subsequent compression efficiency as components are encoded independently from each other.

Discrete Wavelet Transform

After this pre-processing step, an intra-component decorrelation is performed on the tile: on each component a *discrete wavelet transform* is carried out. Wavelets have been developed long before JPEG 2000: the theory of wavelets has been written in the mid-eighties (see for instance the founding paper [42] by Grossman and Morlet) but it actually formalizes some concepts that were already in use at the beginning of the 20th century, like *Haar* functions. We invite the reader to refer to [43] for a complete chronology of the development of this theory.

Like many signal processing tools, wavelets aim at changing the representation of a signal (i.e. the association of a time or a position with a certain value) so as to reorganize the information contained in the signal and reveal some properties that were not appearing clearly in the initial representation. Those properties can then subsequently be used in the targeted application, which could be signal compression for instance. Among commonly used transformations, we should cite the ones based on the Fourier transform (like the Discrete Cosine Transform, used in JPEG). Such transforms unveil the frequency content of a signal but by doing so, they lose the spatial information. In comparison, the wavelet transform has the great advantage to reveal the frequency content of a signal while keeping its spatial consistency. This is called a "time-frequency" representation[1]. Such representation is obtained by comparing the signal to a set of functions, or *atoms*, each atom being obtained by applying simple transformations (translation, dilatation, rotation, etc) to a *mother-function* ψ. In the case of wavelets, ψ has to verify some properties, one of them being that $\int_{\mathbb{R}} \psi = 0$. This implies that ψ oscillates around 0, which explains the term "wavelet". Among other advantages, this transform enables a multi-resolution analysis of the studied signal [65], obtained by successive dilatation of the function ψ. As we know, this ability will be extensively used in JPEG 2000. Getting into a deeper description of this theory is beyond the scope of this thesis but we invite the interested reader to refer to [67] for more information on this powerful tool and the underlying mathematical concepts.

From an "engineer" point of view, a wavelet transform can actually be seen as a filter applied on the signal. In the case of JPEG 2000, the goal of this filter is to separate high and low frequencies of the image. By doing so, the image information (mostly contained in the low frequencies) is concentrated in a very localized zone of the image, allowing to more

[1] The "time" has here to be understood in a broad sense and means the position of a value in the space of the signal being analyzed [47].

efficiently compress the remaining areas. Practically, successive dyadic decompositions are applied. Each uses a bi-orthogonal filter bank, followed by a decimation by a factor 2 in both directions. Each decomposition splits high and low frequencies in the horizontal and vertical directions into four subbands. The subband corresponding to the low frequencies in the two directions (containing most of the image information) is used as a starting point for the next decomposition, as shown in Fig. 2.4. By default, the JPEG 2000 algorithm performs five successive decompositions but this number can vary from 0 to 32. Two filter banks have been standardized: the *Le Gall* (5,3) filter bank, for lossless encoding, and the *Daubechies* (9,7) filter bank, that usually performs better than the first one but is not strictly reversible so that it can only be used for lossy encoding.

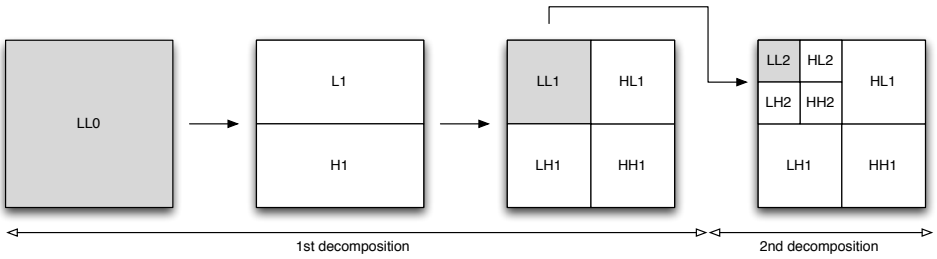


Figure 2.4: *2-levels Discrete Wavelet Transform.*

At the output of the wavelet transform step, the pixels of the image have been transformed in so-called wavelet coefficients, as illustrated in Fig. 2.5. It is important to understand here that no compression has been done up to this point. The image information has just been reorganized and decorrelated so as to concentrate the image energy in the upper left corner. By doing so, the image has been "prepared" for compression: most high-frequency subbands coefficients are indeed close to zero (grey points in Fig. 2.5) and will be therefore very easily compressed.

Quantization

In the case of a lossy compression, a uniform scalar *quantization* with deadzone at the origine is applied to the wavelet coefficients. If lossless compression is required, no quantization is performed. It should be noted that during the rate allocation step described hereunder, a second kind of

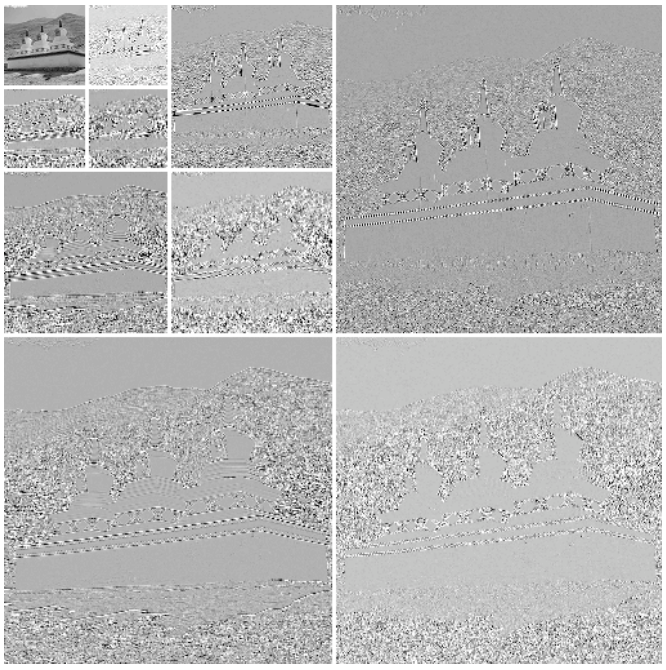


Figure 2.5: *Example of Discrete Wavelet Transform on an image. 3 successive decompositions are carried out. Note that the white borders are only there for visual convenience and are not produced by the DWT.*

quantization is performed, as some parts of the entropy-coded least significant bits of coefficients might be dropped in order to reach a certain compression ratio.

Entropy coding

Every (quantized) subband is then split into rectangular entities called code-blocks. Each code-block is compressed independently using a *context-based adaptive entropy coder*. It reduces the amount of data without losing information by removing redundancy from the original binary sequence. "Entropy" means it achieves this redundancy reduction by using the probability estimates of the symbols. Adaptability is provided by dynamically updating these probability estimates during the coding process. And "context-based" means that the probability estimate of a symbol depends on its neighborhood (its "context"). Practically, entropy coding consists of

- *Context Modeling*: the code-block data is arranged in order to first encode the bits which contribute to the largest distortion reduction for the smallest increase in file size. In JPEG 2000, the Embedded Block Coding with Optimized Truncation (EBCOT) algorithm [111] has been adopted to implement this operation. Unlike JPEG that encodes data coefficient by coefficient, the coefficients in the code-block are *bit-plane* encoded, starting with the most significant bit-plane. The most significant bits from each coefficient allow indeed to rapidly get a coarse idea of its final value and contribute therefore to a larger extent to the global distortion reduction than less significant bits. This way of encoding data is also what enables the bit-depth scalability described above. Moreover, instead of encoding the entire bit-plane in one coding pass, bits from a given bit-plane are distributed in three different categories, according to the distortion reduction they are likely to provide. The neighbors of the bit to encode are used to estimate this reduction and decide to which group the bit belongs. A bit-plane is thus encoded in three successive passes (respectively called significance, refinement and clean-up passes), with the provision of truncating the bit-stream at the end of each pass. During a pass, the modeler successively sends each bit that needs to be encoded in this pass to the arithmetic encoder described below, together with its context.

- *Arithmetic Coding*: the Context Modeling step outputs are entropy coded using a MQ-coder, which is a derivative of the Q-coder [77].

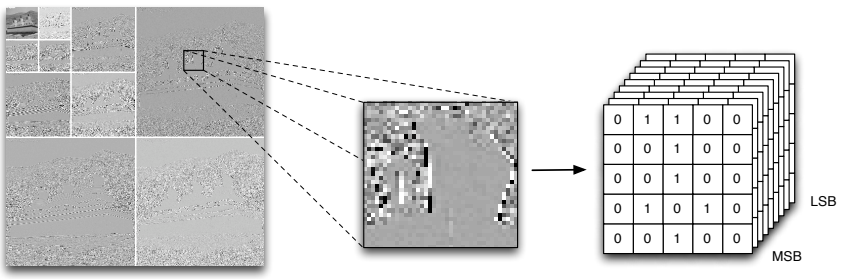


Figure 2.6: *Encoding of a code-block: bit-plane by bit-plane. Code-blocks are usually* 32x32 *or* 64x64 *blocks of wavelet coefficients (here, only 25 coefficients have been represented for visual convenience).*

According to the provided context, the coder chooses a probability for the bit to encode, among predetermined probability values supplied by the JPEG 2000 Standard and stored in a look-up table. Using this probability, it encodes the bit and progressively generates code-words, called segments. Note that when *decoding* an image, the context modeling part only provides the context to the arithmetic coding part and waits for the decoded bit. This bit is computed by the MQ-decoder, that progressively consumes the compressed bit-stream.

Rate allocation

During the *rate allocation*, segments from each code-block are scanned in order to find optimal truncation points to achieve various targeted bit-rates. As mentioned above, each code-block has been encoded so as to encode first the bits that bring the highest reduction in distortion. This leads to a representation that is (close-to-)optimal in the rate-distortion sense. As illustrated in Fig. 2.7, several truncation points are defined in this representation, each one corresponding to a certain slope of the rate-distortion (R-D) curve. When a targeted bit-rate R^* is specified, the rate allocation process tries to find the smallest slope $\lambda^* = \frac{\Delta D}{\Delta R}^*$ such that the sum of all code-blocks increments verifying $\frac{\Delta D}{\Delta R} \geq \lambda^*$ still meets the rate constraint $R \leq R^*$. Once λ^* has been found, all incremental contributions satisfying the above condition are grouped in a so-called quality layer. Several quality layers can of course be defined for a single image, corresponding to successive different rate-distortion trade-offs. Note that a symmetrical mechanism can be applied if a given maximum distortion is specified, in-

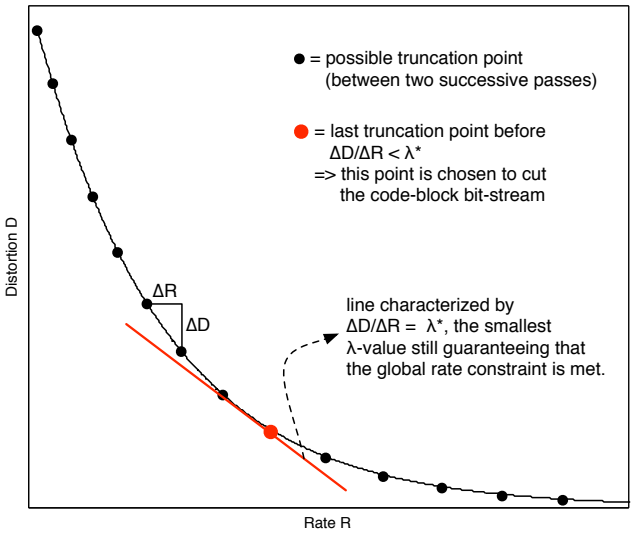


Figure 2.7: *Truncation points on a rate-distortion curve for a given code-block. Each incremental contribution (i.e. each coding pass) is characterized by a certain slope* $\frac{\Delta D}{\Delta R}$ *that will determine to which quality layer the contribution belongs (or if it is dropped and not included at all in the final code-stream).*

stead of a maximum bit-rate.

Packetization and bit-stream organization

Once the rate allocation step has distributed the incremental contributions from all code-blocks over the specified quality layers (and, in case of a lossy encoding, once some of these contributions have been dropped), the compressed data is divided in packets. A packet contains the information related to a certain quality layer from a certain resolution, in a certain spatial location of one of the image components. A spatial location is called a *precinct* and corresponds to the set of code-blocks from all subbands of a given resolution, and covering a given spatial area (Fig. 2.8). Packets, along with additional headers, form the final JPEG 2000 code-stream. As we will see in Chapter 4, useful information is stored in the packet headers, like the length in bytes of the packet body or the number of most significant bit-planes without any bit equal to 1.

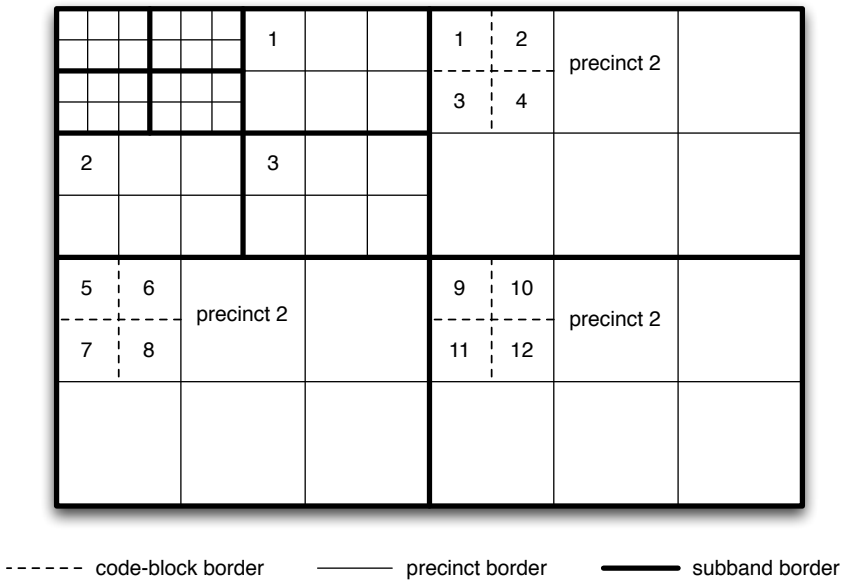


Figure 2.8: *Precinct and code-blocks subdivisions. A precinct includes all code-blocks belonging to a given resolution level and covering a given spatial area. For example, in the highest resolution, the first precinct is made of 12 code-blocks, 4 from each subband. The size (i.e. the number of pixels) of the area covered by a precinct can vary from one resolution level to another. This is the case in the figure where this area size has been adapted in each resolution level so as to have the same number of precincts (6) in each level.*

2.4 Implementation issues

All the nice features described in Section 2.2 are made possible thanks to the algorithm summarized in Section 2.3. However, they come at a certain price, and imply several drawbacks, mainly in terms of complexity. An overview of these weaknesses is given in Section 2.4.1. Depending on the targeted application, various implementation choices can be made to take into account both these drawbacks and the application constraints. The main different kinds of implementation are reviewed in Section 2.4.2. Eventually, Section 2.4.3 briefly describes the characteristics and performances of a hardware architecture we partly developed during this thesis.

2.4.1 Algorithm drawbacks

The main weakness of JPEG 2000 is its complexity. As shown in [57], this complexity is mainly due to the entropy coder, which requires over half the computation time. It can be explained by the bit-level processing of the JPEG 2000 entropy coder, as opposed to the JPEG Huffman coder, which only deals with entire samples. An N-bits sample is processed as N distinct samples by the EBCOT. Moreover, each bit-plane is entirely scanned by three successive passes, while each bit of this bit-plane is encoded only once, in one of the three passes. Finally, both the encoding and decoding schemes have their specific drawback. As the rate allocation step occurs after the entropy-coding step, the encoder has often to encode data that will eventually be dropped, which is not efficient. And concerning the decoder, a feedback loop that does not exist at the encoder side forces the EBCOT to wait for the MQ-decoder answer before being able to further process the code-block. This means that main parts of the EBCOT and MQ algorithm cannot be executed concurrently, which implies additional idle time.

Concerning the DWT part, the computational complexity is much lower, but the memory requirements might be very high as tiles are handled entirely (in comparison with the 8×8 DCT-blocks in JPEG).

Eventually, the system level also raises several problems. In particular, resources required to interface the entropy coding and DWT sub-systems are non-negligible. In a decoder for instance, samples are produced by the entropy coder, bit-plane by bit-plane, one code-block at a time. Those samples are then processed by the Inverse DWT, coefficient by coefficient, one line at a time. This difference in the way each sub-system processes data either implies tight synchronization between those sub-systems, or additional memory resources to let them work independently ([112], p.690).

2.4.2 FPGAs: a powerful, yet flexible solution

As mentioned above, JPEG 2000 can address a variety of different applications, each with its own constraints in terms of available resources and required performances. For many of them, a software implementation of the algorithm will do the job. For instance, this is the case for common still image processing tools, or remote browsing applications (at least the client side of them). The great advantage of such implementations is their high flexibility: they are easily upgradable and customizable, and are cheap to develop and distribute. Among such realizations, we would like to mention the opensource C-library OpenJPEG[1], that we maintain and develop in the UCL Telecommunications Lab. As it is freely available, it is used in a growing number of projects (GDCM[2] or SecondLife[3] to cite some of them) and is continuously being optimized thanks to external contributions.

However, some applications have specific constraints that a software implementation does not adequately fulfill. In those cases, hardware might be considered. These constraints are:

- *High processing rate*: due to the high complexity described above, applications with real-time constraints or high processing load often require a dedicated hardware. A typical example of such application is Digital Cinema, for which JPEG 2000 has been chosen as official standard. On-board image acquisition (in video cameras or satellites) is another example. Nevertheless, as the complexity and power of microprocessors increase, software programs reach higher and higher performances. The fastest JPEG 2000 software implementation, Kakadu[4], has for instance recently announced that it was able to satisfy current Digital Cinema specifications (up to 250 Mbits/sec input rate, up to 4 Gbits/sec output rate) on a Dual Quad Core Xeon x64 from Intel. Therefore, choosing hardware based on a high processing rate requirement will remain relevant in the future depending on the relative evolution of such constraint compared to microprocessors performances.

- *Security*: in some applications, the security aspect -avoid the media content to be stolen or corrupted- is of high importance. Again, this is for instance the case in Digital Cinema, that handles brand new

[1] http://www.openjpeg.org
[2] http://www.creatis.insa-lyon.fr/Public/Gdcm/
[3] http://www.secondlife.com
[4] http://www.kakadusoftware.com

and unseen movies that have consequently a high intrinsic commercial value. Hardware should also in this case be preferred as it is a lot more secure than software programs. And, unlike the "high processing rate" constraint, this is not very likely to change in a near future. Indeed, software programs rely on general purpose operating systems that in turn depend on microprocessors designed to address a lot of different problems. As these microprocessors become more powerful and complex, this inevitably opens new doors for attacks, intrusions, etc. A good example of this is the processor virtualization ability [98] that allows the creation of new kinds of malware. On the contrary, dedicated hardware circuits have a reduced set of functionalities and are, as such, inherently less vulnerable.

When dealing with dedicated hardware, there are basically two kinds of circuits that might be considered. The first one is an ASIC (Application Specific Integrated Circuit). Once such circuit has been designed and tested, the production cost is low. The development process however might be long and expensive, as any new prototype implies a step at the foundry. It therefore targets (very) large markets with well-known and stable technology. On the other hand, FPGAs (Field Programmable Gate Arrays) are entirely reconfigurable. The unit production cost of such chip is higher than for an ASIC but the development of a new architecture is easier and cheaper as it can be modified at any time in the process. Such solution will therefore be used in markets that are smaller or for which upgrade opportunities are of high interest.

It should be noted however that as FPGAs are getting cheaper and more powerful, and as they combine flexibility, performances and security, they become appropriate for an increasing number of applications. Moreover, in addition to a huge amount of high-frequency logic gates, new generations of FPGAs (like the Virtex-5 family from Xilinx [123]) include PowerPC microprocessors (that enable co-design systems), DSPs, Ethernet controllers, high-bitrates interfaces, etc. Such additional features allow to build a complete system into a single component.

2.4.3 Developed architecture

A JPEG 2000 decoder hardware architecture has been developed in the context of our work but it has been decided to not include its detailed description in this text for confidentiality reasons. As such, and because it has not been published, it should not be considered as making part of the original academic contributions of this thesis. However, for the sake of completeness, we give hereunder a brief presentation of this architecture.

Targeted application: Digital Cinema

The application initially targeted by the developed system is Digital Cinema (DC). JPEG 2000 is indeed suited for this emerging market for several technical reasons, including:

- it allows to seamlessly display various resolutions of a movie based on a single code-stream,
- for the very high quality level required by such application, making use of the temporal redundancy (inter-frame compression) does not bring significant improvement, so that a highly-efficient still-image compression algorithm like JPEG 2000 can compete with video standards like those from the MPEG family.

For all these reasons, and for some other political ones[1], the seven major Hollywood studios[2] have decided in 2004 that JPEG 2000 would be the official compression standard used for DC. They released requirements about images in a digital movie [16], summarized in Table 2.2.

Characteristics and performances

The high bit-rates requirements (Table 2.2), the limited market size (around 150 000 projection rooms throughout the world), and the potential evolution of specifications (given the youth of the market), make of the FPGA the ideal platform to be used for DC. The developed architecture has therefore been designed in VHDL, and synthesized and implemented in various Xilinx FPGAs from the Virtex family.

[1] Political reasons include for instance the royalty-free international standard status of JPEG 2000.

[2] These are Disney, Fox, Paramount, Sony Pictures Entertainment, Universal and Warner Bros. Studios. They formed a joint venture, called DCI, whose primary purpose was to establish the specifications of Digital Cinema.

Compression algorithm	JPEG 2000 (ISO/IEC 15444-1)
Max. compressed rate	250 Mbits/sec
Uncompressed rate	2k / 24 fps (= 1.9 Gbits/sec) 2k / 48 fps (= 3.8 Gbits/sec) 4k / 24 fps (= 7.6 Gbits/sec)
Bit-depth	12 bits / component
Color space	XYZ
Tiles	*no tiling*
Code-block size	32x32
DWT	9-7 lossy filter
Internal precision	18 bits

Table 2.2: *Image requirements for Digital Cinema. "2K" means* 2048x1080 *and "4K" means* 4096x2160*.*

The first version of our architecture has actually been designed before the release of the DCI requirements and does not therefore comply with these specifications. However, it allowed us to get familiar with the details of the JPEG 2000 algorithm and the implementation issues described above. This first architecture is detailed in [21].

Next versions of our JPEG 2000 decoding architecture do comply with the DCI specifications. In the context of this thesis, we focused in particular on the MQ-decoder, as described in our technical report [20]. We proposed several optimizations that allow to:

- solve the feedback loop mentioned in Section 2.4.1,
- tightly synchronize the MQ with the context modeling step,
- avoid idle resources so as to output 1 decoded bit at each clock cycle.

The MQ-decoder design has been implemented on a Virtex-4 FX20 (X4VFX20-12) where it occupies only 7% of the chip (600 slices). It reaches an output rate of 222 Mbits/sec, which is higher than the one achieved by equivalent ASIC architectures proposed in the literature, like [109] (172 Mbits/sec) or [83] (175 Mbits/sec).

Partly thanks to the small size of the MQ-decoder design, we were able to place a complete DCI-compliant JPEG 2000 decoder architecture [44, 45], in a medium-sized FPGA, like the Virtex-4 SX35 (X4VSX35-10). Because it makes use of several additional FPGA features, this design is able to

propose a complete JPEG 2000 decoding solution for DC. Among these features, we should cite the use of an included microprocessor to handle some lightweight operations in software (co-design system), or the use of RocketIO interfaces that enable direct HD-SDI outputs, up to 7.6 Gbits/sec. To the best of our knowledge, it is the only complete DCI-compliant system that fits in a single component of this size.

Exploitation

The design briefly described above has lead to the creation of the intoPIX company and has been commercialized as an IP (Intellectual Property) on the DC market. The demonstration board is shown in Figure 2.9. Up to this point, it has reached around 70% of this market. Its main competitor is an ASIC architecture proposed by Analog Devices [26]. As an ASIC, it draws benefit from a low unit cost. However, as initially developed for video surveillance, its main drawback is that several chips have to be parallelized to reach DC throughputs.

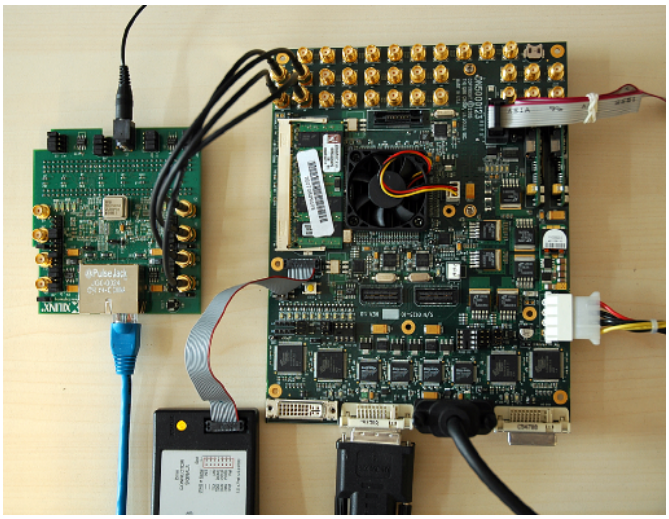


Figure 2.9: *intoPIX demonstration board.*

As JPEG 2000 raises a growing interest in the broadcast field, in particular in the contribution segment, the corresponding encoder architecture has been developed so that a high-performance JPEG 2000 hardware codec [46] can be proposed to address this specific application.

2.5 Conclusion

In this chapter, we have seen that JPEG 2000 is a very flexible and powerful image compression standard. Besides a high compression efficiency, its various scalability levels make it suited for plenty of different applications. Among them, remote browsing and image retrieval will be further investigated in the next chapters.

This rich features set comes at the price of a higher complexity, in comparison to other still image compression standards, like JPEG. When dealing with real-time constraints, it is therefore often required to use dedicated hardware resources. Initially implemented for Digital Cinema, an FPGA-based hardware architecture of a JPEG 2000 decoder has been developed and is now part of a complete JPEG 2000 hardware codec. Such hardware module could be exploited in a lot of applications, including for instance a remote image server, like the one considered in this thesis.

Remote Browsing 3

Prefetching and caching strategies for remote and interactive browsing of JPEG 2000 images

This chapter considers the issues of scheduling and caching JPEG 2000 data in client/server interactive browsing applications, under memory and channel bandwidth constraints. It analyzes how the conveyed data have to be selected at the server and managed within the client cache so as to maximize the reactivity of the browsing application. Formally, to render the dynamic nature of the browsing session, we assume the existence of a reaction model *that defines when the user launches a novel command as a function of the image quality displayed at the client. As a main outcome, our work demonstrates that, due to the latency inherent to client/server exchanges,* a priori *expectation about future navigation commands may help to improve the overall reactivity of the system. The expectation about future navigation commands is formalized based on a stochastic* navigation model*, which defines the probability that a given Window of Interest (WoI) is requested next, knowing previous WoI requests. Based on that knowledge, several scheduling scenarios are considered. Our results demonstrate that, for predictable navigation commands, scheduling data expected to be requested before all the current WoI data have been sent out improves the overall reactivity of the system by up to 30 % compared to conventional scheduling approach. They also reveal that an accurate knowledge of the reaction model is not required to get these significant improvements. The work presented in this chapter has lead to two conference papers [19, 23] and a journal paper [24] on which the following text is based.*

3.1 Introduction

The development and diversification of computer networks as well as the availability of devices able to produce very high resolution images have created the potential for remote access to large scale images [34, 88]. However, despite mobile hardware developments, remotely browsing large scale images remains particularly challenging because of (i) the small size of mobile displays, and (ii) the cost and latency associated to data transmission. The first issue has been extensively studied by image browsing interface designers [13, 87, 95, 96]. For cases where the image resolution is far larger than the display resolution, these works encourage the use of interactive browsing, for which the user interactively defines his/her region and resolution of interest in the image [49, 55, 93]. They also end-up in recommending the adaptation of the display and interaction mechanisms to the application needs [11, 94]. Our work does not consider the design of the browsing interface. We adopt a standard zoom-and-pan browsing interface, and rather study the generic communication and memory management problems raised by interactive remote browsing applications. That is, we consider the scheduling of image data transmitted from the server to the client, and their storage at the client. By data scheduling, we mean the selection of data sent out by the server at any point in the transmission session. Our ultimate goal is to minimize the consumption of channel and client memory resources while increasing the navigation smoothness. At least two components of the browsing system are generally investigated to address these two antagonist goals.

The first aspect is the compression and representation of large scale images. As images can be as large as several gigabytes, the user accessing a remote server wants to be able to access various spatial parts of an image, at various resolutions, without having to download the entire file locally. As explained in Chapter 2, flexible bit-streams such as JPEG 2000 fulfill these requirements. Our work takes advantage of the scalability levels of this image compression standard to limit the access to the parts and the resolutions of the image that are relevant to the browsing session.

The second aspect, which will be investigated in this chapter, is the management of user requests, and the optimal selection of downloaded data. Recent standardization efforts in JPEG 2000 Part 9 [78, 89, 110, 113] have been oriented towards the development of a protocol, named JPIP, to support remote interactivity in client-server based applications. In our work, we assume that a similar protocol is available to forward client requests to the server and send back image data to the client, and we focus on the

issues related to the selection and caching of conveyed image data units. At any transmission opportunity, JPEG 2000 data are selected so as to provide the largest gain in quality per cost unit. Transmitted data are then cached at the client, in a way that best uses the available memory while limiting its fragmentation. As an outcome, the proposed scheduling and caching solutions are expected to pave the way for the design of seamless remote browsing systems characterized by low latency and efficient memory usage.

3.1.1 Contributions and related work

Earlier contributions related to the remote browsing of JPEG 2000 images either address the definition of protocols for interactive browsing, or the optimal selection of JPEG 2000 code-stream units with respect to remote user requests.

Regarding the definition of interactive browsing protocols, in [110, 113], David Taubman presents the JPIP protocol for remotely browsing JPEG 2000 images. Our browsing system architecture, presented in Section 3.2, follows Taubman's proposal in many aspects. That is, all data returned by the server are self-describing, and do not depend on previous requests made by the client. Moreover, user requests are formulated in terms of the geometric and resolution attributes of the region the user is interested in seeing, not in terms of low-level JPEG 2000 constructs. Actually, the main difference between JPIP and our architecture lies in the fact that we advocate the use of an index file to characterize the JPEG 2000 code-stream, i.e. to define the byte range and gain in quality corresponding to each JPEG 2000 data unit. As a consequence, the system component in charge of converting user requests into packet schedules can be located either on the server or on the client side. Similar ideas had been proposed in [25] and [55]. Hence, our browsing architecture directly results from [25, 55, 84, 113], and should not be considered as an original contribution of this chapter.

Regarding the selection of the JPEG 2000 data units to transmit, earlier contributions generally assume the knowledge of the attributes of the region to display at the current time at the client side. Typically, the attributes correspond to the spatial location, resolution, and level of quality of an image rectangular window. Given the attributes of the current window of interest (WoI), JPEG 2000 data units are scheduled so as to rapidly increase the WoI displayed quality. In practice, the transmission order is defined either based on a pre-defined prioritization of JPEG 2000 data units [55, 96], or based on a rate-distortion criterion [114]. In that last

case, Taubman and Rosenbaum define how to schedule data packets so as to maximize the displayed image quality within a region of interest, at each point in the transmission. Despite the dynamic nature of the interactive browsing session, we note that all these approaches are static in essence. That is, only the current state of the WoI is taken into account to define the JPEG 2000 data transmission order. In contrast, our work proposes data scheduling and client memory management strategies that take into account the fact that the WoI definition changes interactively along the browsing session.

We formalize the user behavior based on two complementary models, and propose scheduling and caching methods that best support seamless image browsing. Based on *a priori* knowledge of the user behavior, the central objective of our study is thus to define scheduling and cache management mechanisms that maximizes the system responsiveness, i.e. minimizes the average time needed by the user to decide about future navigation commands. Regarding the user, we first assume the knowledge of a (stochastic) navigation model. The navigation model defines how the browsing commands are structured and correlated along time. We then assume the existence, but not necessarily the knowledge, of a user reaction model, which defines when the navigation commands are triggered, as a function of the received image data. Our work does not pretend to define the "ultimate" navigation and reaction models, which we believe are tightly connected to the envisioned application. Rather, our objective is to evaluate whether the partial or complete knowledge of the user behavior may help to improve the browsing system reactivity, i.e. to reduce the average time between consecutive navigation commands. In particular, we estimate the advantage to draw from data pre-fetching, defined as the transmission of data that are expected to become highly relevant in a near future without necessarily improving the current WoI. As a main outcome, our work demonstrates that, due to the latency inherent to client/server exchanges, *a priori* knowledge about expected future navigation commands may help to improve the overall reactivity of the system, when combined with appropriate pre-fetching mechanisms. Intuitively, that important result is motivated as follows. Because of the system latency, some time elapses between the instant at which all the information needed by the client to decide about the next command has been sent out by the server and the instant at which the next command is received by the server. During that period of time, the information relevant to the current request is conveyed, decoded, and interpreted at the client. Meanwhile, the server might have an opportunity

to anticipate future requests of the user by pre-fetching promising future image data rather than going on forwarding current WoI packets. In that context, our work is primarily interested in defining the conditions under which the navigation and reaction models knowledge significantly improves the browsing system performance. These conditions are related to the accuracy of the knowledge about user behavior, to the deterministic nature of the browsing commands, and to the delay the user needs to access and interpret received data. Our belief is that our contribution may support the design of specific and dedicated browsing applications in that it helps the designer to rapidly figure out whether some envisioned specific models of the user behavior are likely to improve the browsing session or not.

Few works have already proposed to pre-fetch data based on the expectation of the user behavior. In [60], Lin and Zheng predict the user navigation commands and propose heuristic mechanisms to improve the browsing system responsiveness. In contrast, we formulate and solve the pre-fetching problem in a formal rate-distortion (RD) framework. In [19], the authors combine JPEG 2000 flexibility with a user navigation model to manage the client cache and pre-fetch data in a rate-distortion optimal manner. However, the authors in [19] assume that user navigation commands are regularly spaced. So, they neglect the interactive nature of the browsing system. They ignore the fact that the data displayed at the client play a key role on the release of a novel request by the user. In particular, they omit the delay the user needs to receive and interpret image data, and to decide to trigger a novel navigation command. In contrast, our work considers the latency induced by data transmission and interpretation, and analyzes its impact on data pre-fetching strategies. For completeness, we also note that in [62] the authors propose an image attention model to define user-dependent image browsing paths. The proposed technique can be used to define navigation models, and thus provides a valuable complement to our study.

3.1.2 Chapter organization

The chapter is organized as follows. In Section 3.2 we describe the client/server architecture that supports our interactive and remote browsing application. In Section 3.3, we review the work by Taubman and Rosenbaum [114] to serve a static window of interest (WoI) in a rate-distortion way. In Section 3.4, we consider that the user has the capability to dynamically adjust the requested WoI as a function of the scene displayed at the client. In particular, we make the realistic assumption that there is

a non-zero delay in the client/server interaction, and we propose original scheduling mechanisms to take advantage of that browsing system latency when future navigation commands can be predicted. We also consider improved cache management to increase the browsing system responsiveness. Finally, Section 3.5 compares and evaluates the proposed solutions based on simulations, and defines a number of guidelines to design an interactive remote image browsing system. Section 3.6 draw conclusions.

3.2 Our JPEG 2000 image navigation system

The architecture of the navigation system studied in this chapter is largely inspired by the works in [25, 55, 113], and is presented in Figure 3.1. On the left, the client defines a window of interest (WoI) through a *graphical user interface* (GUI), and decodes the parts of the *JPEG 2000 code-stream* received from the server. On the right, the server stores the JPEG 2000 code-stream, interprets the client request in terms of JPEG 2000 packets, and forwards the requested data to the client side. Those parts of the code-stream that are received on the client side are stored in a cache memory. We now detail the architecture components.

On the client side, the user drives the browsing system using a set of commands that are captured and understood by a *graphical user interface* (GUI). As in [110, 113], these commands identify user requests in relative terms (typically zooming and panning), i.e. the commands define the way the current WoI attributes have to be modified to characterize the subsequent WoIs. The WoI attributes define the spatial localization and the resolution level of the image that the user wants to visualize. The *Request Manager* module is in charge of converting WoI increments into actual and absolute WoI features.

Given the WoI features, the JPEG 2000 parser generates and addresses appropriate requests to the server in order to provide the client with the parts of the JPEG 2000 code-stream that are the most relevant to the WoI. As explained in Chapter 2, JPEG 2000 packets are the basic access units to the JPEG 2000 code-stream. They are characterized by their spatial location, resolution, component and quality layer. They allow for progressive and flexible access to the image content [114]. Hence, the JPEG 2000 code-stream parser has the capability to provide fine granular answers to the user requests. It involves a scheduler to select the right packets to send at any point of the browsing session. The scheduler is optimized so as to

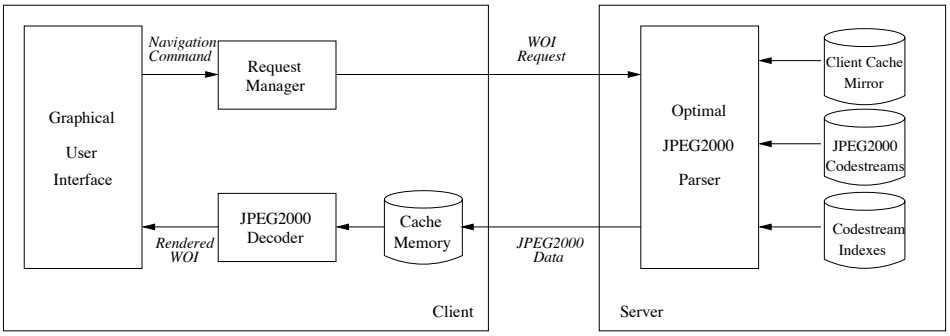


Figure 3.1: *Architecture of our interactive system to browse JPEG 2000 images.*

maximize the browsing system reactivity. The scheduler optimization certainly constitutes the key contribution of this chapter. Here we just note that, similar to [25, 55], the parser uses a *code-stream index* to characterize and access JPEG 2000 packets efficiently. The code-stream index file is generated by the OpenJPEG library (http://www.openjpeg.org), and defines the gain in quality -see Section 3.3- and the range of bytes corresponding to each packet. Besides, from a functional point of view, we observe that the *JPEG 2000 parser* can be located either on the client or on the server side, depending on the resources available on each side [75]. A client application with limited resources can rely on the server to deal with the JPEG 2000 parser. In that case, to avoid retransmission of packets that are already available at the client, the scheduler maintains a *client cache mirror* [110] at the server (as depicted in Figure 3.1). On the contrary, a server that is expected to face a large number of clients can distribute the JPEG 2000 parser load to its clients. The code-stream index is then forwarded to the client during the connection phase, before launching the browsing session. To limit the transmission overhead, the gain in quality provided by each packet can be replaced by a set of Lagrange multipliers that characterize the gain in quality per unit of packet size for each quality layer.

To complete this section, we note that the communication channel envisioned by our navigation system is characterized by a constrained -possibly fluctuating- bandwidth, but is assumed to be lossless. For interested readers, error protection of JPEG 2000 packets has been considered in [80], and media content scheduling in presence of losses has for example been formal-

ized in [12]. In this work, we are interested in the dynamic and interactive nature of the WoIs definition, and not in the robustness to transmission losses. Hence, the main contributions of this work describe how the scheduler (Section 3.4.2) and the cache memory (Section 3.4.3) are managed to cope with a dynamic browsing environments.

3.3 Packet scheduling for a static WoI

This section first summarizes the notions detailed in Chapter 2 that enable a flexible and progressive access to remotely stored JPEG 2000 images. It then considers JPEG 2000 image browsing and surveys the rate-distortion optimal solution proposed in [114] to download JPEG 2000 packets in an order that maximizes, at any point of transmission, the quality displayed in a pre-defined WoI. Finally, Section 3.3.3 describes how the RD optimal solution in [114] applies to other cost functions than the packet size in bytes. We review these earlier works because they support our own contributions presented in Section 3.4.

3.3.1 JPEG 2000 code stream abstraction

As explained in Chapter 2, JPEG 2000 is based on a discrete wavelet transform, whose subbands are partitioned into *code-blocks* that are coded independently. Each code-block is coded into an embedded bit-stream, i.e. into a stream that provides a representation that is (close-to-)optimal in the rate-distortion sense when truncated to any desired length. To achieve rate-distortion (RD) optimal scalability at image level, the embedded bit-stream of each code-block is partitioned into a sequence of increments based on a set of truncating points that correspond to the various rate-distortion trade-offs [111] defined by a set of Lagrange multipliers. A Lagrange multiplier λ translates a cost in bytes in terms of distortion. It defines the relative importance of rate and distortion. Given λ, the RD optimal truncation of a code-block bit-stream is obtained by truncating the embedded bit-stream so as to minimize the Lagrangian cost function $\mathcal{L}(\lambda) = D(R) + \lambda R$, where $D(R)$ denotes the distortion resulting from the truncation to R bytes. Different Lagrange multipliers define different rate-distortion trade-offs, which in turns result in different truncation points. For each code-block, a decreasing sequence of Lagrange multipliers $\{\lambda_q\}_{q>0}$ identifies an ordered set of truncation points that partition the code-block bit-stream into a sequence of incremental contributions [111]. Incremental contributions from the set of image code-blocks are then collected into so-called quality layers, $\mathcal{Q}_q$.

Because the rate-distortion trade-offs targeted during truncation are the same for all the code-blocks, for any quality layer index l the contributions provided by layers $\mathcal{Q}_1$ through $\mathcal{Q}_l$ constitute a rate-distortion optimal representation of the entire image. It thus provides distortion scalability at the image level. Resolution scalability and spatial random access to the image result from the fact that each code-block is associated to a specific subband and to a limited spatial region.

Although they are coded independently, code-blocks are not identified explicitly within a JPEG 2000 code-stream. Instead, the code-blocks associated to a given resolution are grouped into *precincts*, based on their spatial location [6, 114]. Hence, a precinct corresponds to the parts of the JPEG 2000 code-stream that are specific to a given resolution and spatial location. As a consequence of the quality layering defined above, a precinct can also be viewed as a hierarchy of *packets*, each packet collecting the parts of the code-stream that correspond to a given quality among all code-blocks matching the precinct resolution and position. Hence, packets are the basic access unit in the JPEG 2000 code-stream.

3.3.2 Rate-distortion optimal scheduling for a static WoI

In this section, we review the solution proposed by Taubman and Rosenbaum in [114] to schedule JPEG 2000 packets so as to maximize the quality of the image visualized at the client, at each point along the packet transmission sequence. The image visualized at the client is defined by a static window of interest (WoI), which identifies the spatial region and the resolutions that are relevant to the display. The rate-distortion (RD) optimal solution described in [114] simply selects the packets to transmit in decreasing order of WoI distortion reduction per transmitted bytes. The solution is in accordance with the dependency relationship existing between packets because, due to the embedded nature of the code-stream, the reduction of distortion per byte decreases as the packet layer index increases within a precinct[1].

We have seen in Chapter 2 that, for a given image component, a JPEG 2000 packet corresponds to a specific quality layer and to a specific precinct. Formally, we define a precinct with two indices r and p, respectively characterizing its resolution and spatial location. A JPEG 2000

[1] In practice, the reduction of distortion per unit of transmission can be locally non-decreasing, due for example to empty packets [114]. In that case, consecutive packets are aggregated until absorption of the local maximum.

packet is thus completely defined by its precinct indices r and p, and by its layer index q, s.t. $0 < q \leq Q$, with Q denoting the total number of quality layers. Following [114], we introduce $s_b(r, p, q)$ to be the size in bytes of the q^{th} packet of the (r, p) precinct, and denote $d(r, p, q)$ to be the amount by which the distortion, measured on the whole original image, is decreased if the packet (r, p, q) is decoded compared to the distortion if only the packets (r, p, i), $i < q$, are decoded. As a consequence of that definition, the metric $d(r, p, q)$ has to be additive, i.e. the gain in quality provided on the entire image by multiple packets has to be equal to the sum of the gain provided by each individual packet. In our work, the additive distortion $d(r, p, q)$ is derived from the approximation of the Mean Square Error (MSE) defined in [111].

Formally, the distortion D_i^q associated to the reconstruction of code-block B_i from its first q quality layers is defined by

$$D_i^q = w_{b_i}^2 \sum_{(x,y)\in B_i} [\hat{c}_i^q(x, y) - c_i(x, y)]^2 \tag{3.1}$$

There, $c_i(x, y)$ denotes the subband coefficient in code-block B_i, $\hat{c}_i^q(x, y)$ denotes the quantized representation of these coefficients associated to the first q quality layers, and w_{b_i} denotes the L2-norm of the wavelet basis functions for the subband to which the code-block B_i belongs. It has been shown that, although the wavelet basis functions used in JPEG 2000 are not strictly orthogonal, D_i^q reasonably approximate the MSE and is thus additive [99]. Hence, letting $\Gamma(r, p)$ denote the set of code-blocks belonging to precinct (r, p), the incremental reduction of distortion $d(r, p, q)$ associated to the decoding of packet (r, p, q) is defined by

$$d(r, p, q) = \sum_{i\in\Gamma(r,p)} D_i^{(q-1)} - \sum_{i\in\Gamma(r,p)} D_i^q. \tag{3.2}$$

We now follow [114] to explain how the reduction of distortion $d(r, p, q)$ provided on the whole image is adjusted to take into account the fact that only the WoI, defined by its position and its resolution, is displayed at the client. In [114], the authors have proposed to approximate the distortion reduction provided by a packet on a static WoI as the product of the distortion reduction provided on the whole image with the fraction of pixels that are in the WoI, from the set of pixels affected by the packet subband coefficients. We use the same approximation, but extend it to cases where the resolution $\mathcal{R}(w)$ of the displayed WoI might be lower than the original

image resolution, denoted r_{max}. Letting $d_\xi(r,p,q,w)$ denote the distortion reduction provided by packet (r,p,q) on the WoI w, we have thus

$$d_\xi(r,p,q,w) = \xi(r,p,w).d(r,p,q).4^{(\mathcal{R}(w)-r_{max})} \tag{3.3}$$

where (i) the factor $4^{(\mathcal{R}(w)-r_{max})}$ accounts for the decrease of the spatial support of the precinct displayed at lower resolution[1], and (ii) $\xi(r,p,w)$ denotes the fraction of pixels affected by precinct (r,p) that belong to WoI w. Letting $\mathcal{P}_{(r,p)}$ denote the set of pixels that are affected by the subband coefficients of precinct (r,p), and $\mathcal{P}_w$ the set of pixels in the WoI w, $\xi(r,p,w)$ is defined by

$$\xi(r,p,w) = \frac{||\mathcal{P}_w \cap \mathcal{P}_{(r,p)}||}{||\mathcal{P}_{(r,p)}||} \tag{3.4}$$

Based on the above definitions, to maximize at each point of transmission the quality displayed in the static WoI w, packets have to be forwarded to the client in decreasing order of $\xi(r,p,w).4^{(\mathcal{R}(w)-r_{max})}.d(r,p,q)/s_b(r,p,q)$ [114]. Keeping in a compact form the sequence of $d(r,p,q)/s_b(r,p,q)$ for all image packets is unpractical. This is especially the case when this information has to be transmitted to the client in the form of a code-stream index file (see Section 3.2). An alternative, proposed in [114], consists in approximating the distortion reduction per unit of transmission, i.e. $d(r,p,q)/s_b(r,p,q)$, by the Lagrange multiplier λ_q used to define the truncation points of the q^{th} quality layer. Based on that approximation, the quality displayed in the WoI is maximized at any packet transmission by forwarding packets to the client in decreasing order of $\xi(r,p,w).\lambda_q$, with $0 < q \leq Q$ [114]. In the rest of the chapter, the reduction of distortion per unit of transmission associated to a packet is estimated based on actual $d(r,p,q)$ and $s_b(r,p,q)$ measurements, but our simulations have revealed that a Lagrange multiplier approximation does not significantly affect the browsing performance.

3.3.3 From rate/distortion to cost/distortion optimality

In this section, we explain how the solution in [114] generalizes to other packet cost function than the packet size in bytes.

[1]Note that this factor is a constant when a static WoI is considered, but is required in the dynamic case, where distortion decrease expectations are computed over multiple WoI's at multiple resolutions. The power of 4 arises from an approximation, and is only strictly valid provided the wavelet transform's basis functions are orthogonal.

Let $C(r,p,q)$ denote a monotonically increasing function of q that defines the cost associated to the first q packets of precinct (r,p). Obviously, in contrast to the packet size in bytes, an arbitrary cost function does not necessarily guarantee that the cost/distortion characteristics defined by the truncation points of a precinct do lay on a convex-hull. In that case, selecting the packets in decreasing order of distortion reduction per cost unit does not ensure cost/distortion optimality, i.e. it does not necessarily maximize the distortion reduction obtained in the WoI for a given cost constraint. To achieve optimality, it is first required to identify the subset of truncation points that actually lay on the cost/distortion convex-hull for each precinct. Cost/distortion optimality is then obtained by restricting the admissible truncation points of a precinct to the only convex-hull optimal truncation points, and by forwarding the corresponding subsets of consecutive packets in decreasing order of benefit per cost unit. Importantly, it is worth noting that the convex-hull analysis can be performed a priori for each precinct, independently of the (dynamic) WoI features. This is because the factor $\xi(r,p,w)$ in Equation (3.3) is constant for all packets within a precinct.

We now introduce two cost functions that are used later in the chapter. They define possible alternatives to the packet size in bytes. The first one is used when the transmission network forces the scheduler to pack each JPEG 2000 packet into an integral number of transmission units with padding. In that case, letting ζ_t denote the transmission unit size in bytes, the cost $C_t(r,p,q)$ in transmission units for the first q packets of precinct (r,p) is defined by

$$C_t(r,p,q) = \sum_{l=1}^{q} \left\lceil \frac{s_b(r,p,l)}{\zeta_t} \right\rceil \tag{3.5}$$

The second alternative cost function occurs when the resource constraint imposed to the scheduler is expressed in terms of storage rather than transmission capacity. Fundamentally, the memory cost of a packet depends on how JPEG 2000 packets are mapped onto the physical memory. Here, we anticipate the next section, and consider the dynamic release of multiple and consecutive navigation commands by the user. In that case, obsolete JPEG 2000 packets have to be removed to leave space for incoming packets. In practice, packets are removed as a function of the navigation commands, and not as a function of their order of arrival. Such pseudo-random packet removal order, combined with the variable size of JPEG 2000 packets, results in memory fragmentation. This is because an incoming packet might have to be split in several fragments to be stored in the memory slots that have been released by the former removal of obsolete packets. Memory

fragmentation spreads the packets across the physical memory, and slows down the access to the data. To mitigate the problem, a dynamic memory manager generally allocates memory space in terms of memory units. A memory unit is a set of bytes that are jointly allocated or freed, and that can be accessed efficiently, for example because they are adjacent in the physical memory. To define the exact cost of a packet in terms of memory units, we point out some consistency in the arrival/removal order of the precinct components. Due to the dependency between layers, the precinct packets arrive in increasing order of layers, and are removed in decreasing order of layers. So, within a precinct, packets are removed in reverse order of arrival. This feature motivates the definition of a specific memory management strategy, for which only the first packet of a precinct is synchronized with the beginning of a memory unit, while other packets are stored immediately after the last byte of the packet corresponding to the previous quality layer of the same precinct. This management strategy is in accordance with the syntax of the JPIP protocol, which considers precinct data streams as the fundamental data unit to be accessed and processed by the decoder [113]. Based on that management strategy, and letting ζ_m denote the memory unit size in bytes, the cost $C_m(r,p,q)$ in memory units for the first q packets of precinct (r,p) is defined by

$$C_m(r,p,q) = \left\lceil \frac{\sum_{l=1}^{q} s_b(r,p,l)}{\zeta_m} \right\rceil \tag{3.6}$$

Note that for unitary sizes of both transmission and memory units, both costs defined in (3.5) and (3.6) correspond to the packet size in bytes $s_b(r,p,q)$. In that case, the optimal scheduler simply forwards packets in decreasing order of distortion reduction per byte, without having to perform a convex hull analysis, since the size/distortion characteristics of any truncation point of a precinct already lay on a convex-hull. For non-unitary transmission or memory sizes, both schedulers might differ and a convex-hull analysis is required.

In the next section, we present the original contributions of our work. The envisioned framework differentiates from the work in [114] by the fact that we consider a dynamically evolving WoI. It raises a number of important and fundamental issues going from the modelization of the user behavior, to the definition of optimal image pre-fetching strategies, passing by the study of the impact of the client cache on the browsing system responsiveness.

3.4 Dynamic packet scheduling

In this section we are primarily interested in the changes of WoI, and in their consequence in terms of packet scheduling decisions. The purpose of our study is to figure out whether it is possible to reduce the navigation system latency by proper anticipation of the WoI updates or by an improved management of the client cache memory.

Fundamentally, the anticipation of future commands is motivated by the fact that there exists a delay between the emission of a packet by the server and the feedback it causes in terms of WoI definition. That delay includes the transmission delay (in both directions), the decoding latency, and the period of time the user needs to analyze the newly displayed visual information to decide about the future navigation command. We use the term *reaction delay* to refer to the aggregation of these multiple sources of latency. Evidently, the reaction delay degrades the responsiveness of the browsing system. In particular, it causes a waste of resources if current WoI data are still forwarded between the instant at which the server has sent out all the data needed by the user to decide about the next navigation command, and the instant at which the server gets informed about that next command. During that period, it is definitively better to pre-fetch data that are likely to be requested in the future, rather than transmitting useless quality increments for the current WoI. In practice, the anticipation of future WoI relies on two components. First, it depends on the *a priori* knowledge (or expectation) of future requests. Second, it requires the definition of schedulers that favor the download of promising JPEG 2000 packets.

Section 3.4.1 addresses the first component of the problem. It formalizes the release of navigation commands by defining which requests are sent out by the user, and when they are triggered in response to the content displayed in the WoI. Our intention is not to propose universal models to describe the user behavior. Rather we fix these models *a priori* and are interested in the analysis of how appropriate scheduling mechanisms are able to take advantage of their knowledge. These schedulers anticipate future navigation commands by favoring JPEG 2000 packets that are likely to become crucial for future WoIs. Hence, in some occasions they are expected to forward image data before they are actually requested by the user. For this reason they are referred to as pre-fetching mechanisms. Our work compares three pre-fetching strategies, presented in Section 3.4.2, and which differentiate by the knowledge they have about the user behavior models.

To complete our investigation of the factors affecting the responsiveness of a dynamic browsing session, Section 3.4.3 considers the management of the client cache memory. It observes that efficient cache management is likely to increase the system responsiveness, simply because it increases the amount of data available at the client side. Hence it considers client cache usage optimization, and raises a fundamental question regarding the browsing responsiveness: is it worth optimizing the transmission schedule in terms of storage rather than transmission cost? The first solution maximizes the initial quality available from the cache upon release of a novel navigation request, while the second maximizes the WoI quality increment per unit of transmission. Section 3.4.3 describes how both solutions are implemented. Later in Section 3.5.3, simulation results demonstrate that maximizing the usage of channel resources also maximizes browsing responsiveness in most practical cases.

3.4.1 User navigation and reaction models

We characterize the user by two components. First, a navigation model defines which command is expected to be released next. Second, a reaction model defines when the next command is triggered, in response to data transmitted for the current WoI.

To define our navigation model, we adopt the formalism introduced in [19]. We assume that the user only browses through various image resolutions and spatial locations. The number of points displayed in the WoI is constant, so that the image area covered by the WoI changes with the zoom factor. For simplicity, we limit the possible commands to 6 actions, which select the next WoI based on zoom-in/zoom-out actions or down/up/right/left translations[1]. We then assume that the sequence of user's actions is defined by a first order Markov process. Specifically, an action is chosen identical to the previous one with probability δ, while all other actions have the same probability $(1-\delta)/5$. The δ parameter reflects the predictable nature of the navigation process, i.e. a δ value close to one means that future WoIs are reliably predicted. Based on that model, the server can calculate the probability that a packet appears in the WoI at any future step of the browsing session. However, for our practical implementation, we decided to restrict our prediction of the future to a "myopic" one step look-ahead. So, the probability that a packet becomes relevant in

[1] Zoom-in and zoom-out actions increase or decrease the resolution by a factor of two. Translations magnitude equals the size of the WoI.

the future is simply computed as the probability that it belongs to the next WoI.

To model the reaction of the user, we adopt a rudimentary formalism. Fundamentally, the model has to reflect that (i) a user command is triggered in response to the content displayed in the current WoI, and (ii) packets sent out by the server need some time to be conveyed, displayed (that is, decoded and rendered) and interpreted at the client. In the rest of the chapter, we assume that the next navigation command is triggered τ seconds after the PSNR in the current WoI went beyond a threshold Θ, in dBs. Whilst minimalist, this user reaction model captures the two fundamental features listed above. Its simplicity makes it easy to handle and interpret. Note that, for convenience and without loss of generality, when analyzing our simulation results in Section 3.5, the reaction delay τ is expressed in terms of the number of bytes B_τ that the server-client channel conveys in τ seconds. In our simulations also, Θ has been defined to be constant at all resolutions.

3.4.2 Anticipation of future navigation commands

This section presents the three pre-fetching methods proposed to anticipate future navigation commands. The first one, named *oracle scheduling*, assumes the complete knowledge of the user navigation and reaction models, and maximizes the responsiveness of the browsing system by anticipating future WoIs as soon as the data sent out for the current WoI allow for deciding about the next navigation command. The second and third strategies only assume a partial knowledge of the behavior to expect from the user. They consider that the user navigation model is known, but that the reaction model is unknown. Such scenario is realistic in the sense that the navigation model captures the browsing logic, while the reaction model captures more complex user features, such as its ability to interpret the image content.

Oracle scheduler

The *oracle scheduler* knows all about the user behavior models. It knows the δ parameter of the navigation model, and the Θ and τ parameters of the reaction models. Based on that knowledge, the oracle scheduler selects the packets that best contribute either to the current WoI or to future WoIs, depending on whether data sent out by the server reach the quality threshold Θ in the current WoI or not. Hence, the oracle feeds the

current WoI until the Θ quality threshold is achieved. Upon that point, the oracle selects the JPEG 2000 packets so as to maximize the gain in quality expected per cost unit for future WoIs.

The gain in quality (or reduction of distortion) expected for future WoIs is computed based on a one step look-ahead horizon, and only depends on the navigation model. Specifically, at a given time t, we denote $\mathcal{W}_t$ to be the set of possible future WoIs, and therefore, in our simulations $\mathcal{W}_t$ just contains the 6 possible next WoIs (see Section 3.4.1). Let W_t denote the random variable that defines the next WoI selected within $\mathcal{W}_t$ by the user navigation command. Then, for each element $w \in \mathcal{W}_t$, $p_{W_t}(w)$ denotes the probability for w to be the next WoI, and $d_\xi(r,p,q,w)$, which is defined in Equation (3.3), denotes the reduction of distortion provided on w by the q^{th} quality layer of the (r,p) precinct. Based on these definitions, the reduction of distortion $E[d_\xi(r,p,q,W_t)]$ expected for packet (r,p,q) in the future is defined by

$$E[d_\xi(r,p,q,W_t)] = \sum_{w \in \mathcal{W}_t} p_{W_t}(w).d_\xi(r,p,q,w). \tag{3.7}$$

Hence, based on Equation (3.7) and (3.3), the reduction of distortion per packet size unit can be written as the product of a navigation-dependent term, i.e. $\sum_{w \in \mathcal{W}_t} p_{W_t}(w).\xi(r,p,w).4^{(\mathcal{R}(w)-r_{max})}$, and an image data dependent term, i.e. $d(r,p,q)/s_b(r,p,q)$. Note that the same conclusion holds when an alternative cost function is used. From the system architecture point of view, we note that the first term can be computed equivalently at the client or server side. In contrast, the information associated to the second term is initially only available at the server, and needs to be conveyed across the network when the scheduler is implemented at the client, e.g. for complexity distribution purposes (see Section 3.2). In that case, as commented in Section 3.3 and suggested in [114], to limit the amount of information to transmit to the client, it is recommended to approximate the $d(r,p,q)/s_b(r,p,q)$ ratios by their corresponding Lagrange multipliers. We now propose schedulers that attempt to take advantage of the navigation model without knowing the reaction model.

Natural pre-fetching

When the user reaction model is unknown, i.e. when the scheduler ignores the impact of the current WoI quality on the instant of release of the next navigation command, a natural way to schedule data consists in

feeding the current WoI as long as possible. By *natural pre-fetching*, we thus refer to the pre-fetching that is initiated once all packets that are relevant to the current WoI have been sent out by the server. As for the oracle scheduler, the natural pre-fetching method works in two steps. The first step feeds the current WoI in decreasing order of distortion reduction per cost unit. That initial step goes on till all data that are relevant to the current WoI have been forwarded (and not till the threshold Θ has been achieved, as for the oracle). Then the scheduler selects the JPEG 2000 packet that maximizes the gain in quality expected per cost unit for future WoIs, as defined based on Equation (3.7). This mechanism goes on until the next navigation command is triggered, as long as memory space is available in the client buffer. Memory management is considered in Section 3.4.3.

Anticipated pre-fetching

The natural pre-fetching strategy has defined how data are pre-fetched once the entire current WoI has been sent out by the server. In contrast, this section proposes to consider pre-fetching *before* all the information relevant to the current WoI has been transmitted to the client. Our investigation is motivated by (i) the observation that there exists a delay between the emission of image data by the server and their interpretation by the user, and (ii) the fact that the current WoI quality increments are obtained at increasing cost as the downloading process goes ahead. As a consequence, at some point, it might be more beneficial to download a JPEG 2000 packet that has a high probability to bring a significant gain in the future, rather than wasting resources for non-significant increments of the current WoI. For this reason, *anticipated pre-fetching* schedules data packets according to the gain they provide on the current WoI, but also as a function of the gain expected for the next WoI. We propose a weighted sum to trade-off current and future expected benefits. Formally, as in Section 3.4.2, let $w(t)$ denote the WoI at current time t, and let W_t denote the random variable associated to the selection of the next WoI. The gain in quality foreseen for packet (r, p, q) at time t is then defined as the weighted average of the benefits in distortion respectively provided on the current WoI and expected for the next WoI. Specifically, letting α denote a weighting factor between 0 and 1, we define the gain $d_\alpha(r, p, q, t)$ associated to packet (r, p, q) by

$$d_\alpha(r, p, q, t) = (1 - \alpha).d_\xi(r, p, q, w(t)) + \alpha.E[d_\xi(r, p, q, W_t)], \qquad (3.8)$$

where $d_\xi(r, p, q, w(t))$ and $E[d_\xi(r, p, q, W_t)]$ are defined as in Equation (3.3) and Equation (3.7), respectively. Hence, anticipated pre-fetching transmits

JPEG 2000 packets in decreasing order of weighted gain in quality per cost unit[1]. Section 3.5 considers the sensitiveness of the proposed algorithm to the α parameter, and shows that for most cases of interest, anticipated pre-fetching significantly improves natural pre-fetching and keeps similar performance over a large range of α values, typically $0.05 < \alpha \leq 0.3$. In that case, the choice of α is thus not crucial. A possible practical approach to select α when the reaction model is unknown consists in giving the user the possibility to adjust α according to its feeling about the reactivity of the system. Large α values slow down the quality increments in the current WoI, but better anticipate future WoIs, which in turns increases the initial quality in the next WoI. The best trade-off can be found by user adjustments.

It is also worth mentioning that we deliberately consider a completely generic approach to drive anticipated pre-fetching, because one of our purposes is to estimate whether a formal and accurate definition of the user reaction model is needed to achieve browsing performance closer to the ones obtained by the oracle scheduler. Hence, in defining anticipated pre-fetching, we do not make any assumption about the nature of the user reaction model. For example, we do not exploit the fact that there is an (unknown) quality threshold beyond which additional data for the current window become useless. Interestingly, the simulation results presented in Section 3.5 demonstrate that, whilst rudimentary, the anticipated pre-fetching mechanism defined based on Equation (3.8) is able to exploit the user reaction delay efficiently.

3.4.3 Client cache management and browsing responsiveness

As a primary motivation for this section, we first observe that the content stored in the client cache memory of a dynamic browsing session impacts the system responsiveness because it determines the initial quality rendered upon release of a novel navigation command. An efficient usage of the memory is thus expected to increase the responsiveness as it potentially increases the quality available from the cache upon release of a novel command. For this reason, the section considers the addition and removal of packets to/from the memory.

[1]Note that because JPIP allows the client to issue multiple requests concurrently [113], it can also support the above formulation, in terms of weighted sums of quality gains.

First, we consider the addition of packets to the cache. The selection of incoming packets obviously affects the memory content, and directly depends on the scheduling decisions. Besides the scheduler also determines the benefit obtained per transmission unit, which defines the time period required to bridge the gap between the initial WoI quality and the quality threshold triggering the next command. Hence, the scheduler affects both the memory and channel usage, which in turn determine the browsing system responsiveness. This argument motivates the comparison between memory- and channel- optimized schedulers. Both solutions transmit JPEG 2000 data in decreasing order of distortion reduction per cost unit, but differ in the way they estimate the cost of a packet. The first solution measures the cost in memory units, while the second defines it in terms of transmission units. The first solution maximizes the benefit to get from the data stored in the client cache, and is thus expected to maximize the initial quality available from the client cache upon release of novel user request. In contrast, the second solution maximizes the increase of quality per unit of transmission, and thus minimizes the period of time needed to bridge the gap between the initial WoI quality and the Θ threshold. However, this second solution does not guarantee that the available memory space has been best allocated, i.e. that the packets stored upon filling of memory maximize the quality rendered in the WoI for the given memory size constraint. The simulations presented in Section 3.5.3 compare the performance of the memory- and channel-optimized schedulers in terms of browsing system responsiveness, and conclude that in most practical cases optimizing channel resources results in higher responsiveness than optimizing memory resources.

Second, we are interested in the removal of packets from the cache, and explain how it affects the practical implementation of the channel-optimized scheduler. Efficient memory usage requires removing the packets in increasing order of benefit per memory unit. Such removal strategy minimizes the loss of quality per freed memory units, independently of the scheme adopted to schedule packets at the server. As a consequence, when packets are scheduled based on transmission units, two distinct cost functions are followed to fill-in and fill-out the memory, which means that the addition and removal of packets to/from the memory targets potentially distinct cost/distortion convex-hull optimal truncation points. As a consequence, the precinct stored in memory at a given time might be convex-hull sub-optimal regarding one of the storage or transmission costs. Hence, the selection of the packets to add or remove to/from the memory has to cope

with a memory that is potentially convex-hull sub-optimal regarding the cost of interest. In the following, we describe how packets are transmitted, given an arbitrary memory state. Equivalent developments hold regarding memory packets removal.

To simplify notations, and without loss of generality, the precincts, originally defined by their (r, p) indices, are labeled by a single index j. We then define n_j to be the number of layers already stored in, or in transit to, the client buffer for the j^{th} precinct. For each precinct j, we select the layer indices $l_{n_j} > n_j$ that maximizes the distortion reduction per cost unit associated to the subset of packets labeled from $(n_j + 1)$ to l_{n_j}. The subset of packets to forward next is then defined to be the one that maximizes the distortion reduction per cost unit for all precincts j. The process goes on iteratively as long as transmission resources are available. It is globally optimal because, at each step, it selects the subset of packets that correspond to the largest slope of the set of convex-hulls sustained for every precinct j by the precinct's cost/distortion characteristics corresponding to n_j or more layers. In a practical and efficient implementation, the set of l_{n_j} associated to the j^{th} precinct can be computed a priori for all possible n_j values, independently of the browsing dynamics. This is because, in Equation (3.3), (3.7) and (3.8), the factor that weights the distortion reduction $d(r, p, q)$ of a packet to reflect the WoI dynamics is the same for all the packets of a precinct.

3.5 Simulation results

In this section, we rely on simulations to compare the oracle scheduler with the natural and anticipated pre-fetching methodologies. The simulations validate our proposed pre-fetching methodologies. They demonstrate that it is possible to improve the rhythm at which the user interacts with the image by anticipating the transmission of expected future information. As an additional outcome, our analysis provides generic guidelines for interactive image browsing system designers. That is, we identify the conditions that make the learning of application-specific navigation or reaction models relevant and helpful to improve browsing fluency.

As the goal of our work is not to propose a navigation model, but rather to study the benefit that one could expect from the knowledge of such a model, we fix *a priori* the navigation model and assume its knowledge by all schedulers. We then compare the performance of the 3 schedulers based

on a sequence of navigation commands generated by the chosen model. Although the sequence of navigation commands is fixed, the instants at which commands are triggered depend on the scheduler. Indeed, these instants are defined by the reaction model, in response to the quality displayed in the WoI, which in turn depends on the scheduling and cache management mechanisms. Hence, desirable scheduling and caching strategies end up in minimizing the time elapsed between consecutive commands. Without loss of generality, and to avoid the definition of an artificial reference channel bandwidth, our simulations express this time in terms of the number of bytes which could be transmitted by the channel between consecutive navigation commands.

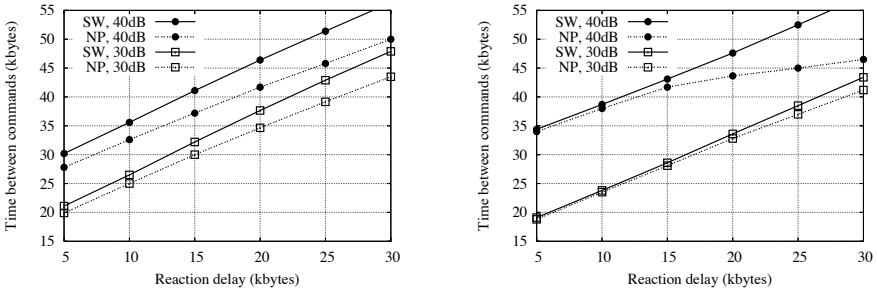
We now present some common parameters to all our simulations. All results are based on a 256×256 WoI. For each simulation, a sequence of 1000 commands has been generated based on the navigation model defined in Section 3.4.1. The picture used in our simulation is a 32768×32768 greyscale image defined based on a crop of downtown Toulouse provided by SPOT Image in the scope of IST-2000-28646 PRIAM project. JPEG 2000 coding parameters have been defined as follows. Tiles have a size of 1024×1024 pixels. Six resolution levels have been considered for each tile, and precincts of 128×128 have been defined at each resolution level[1]. Ten quality layers respectively target a quality of 26, 30, 32, 34, 36, 38, 40, 42, 44, and 46 dB in each tile. Due to the finite size of the client cache memory, once the cache gets full, packets have to be removed to leave space for incoming packets. As explained previously, to support optimal memory usage in all experiments, packets are removed in increasing order of loss in quality per storage cost, except in the experiments presented in Figure 3.10 where removal strategies based on storage and transmission costs are compared. In the case of static WOI scheduling (see next section), if some space is needed in the cache, all packets that do not belong to the current WoI are removed. This reflects the fact that expected future WoIs are not considered at all.

Sequences of 10 000 navigation commands have been used. To ensure the stability of the obtained results, some tests have been made with sequences of 50 000 commands leading to the same average values. Concerning the variance, several different sequences have been used for each δ-value, without leading to significant differences in the results.

[1] Obviously, smaller precincts are used for resolution 0 and 1, which contain less than 128×128 samples.

3.5.1 Natural pre-fetching vs. static WoI schedules

As a preamble to our analysis, Figure 3.2 compares the natural pre-fetching solution and the RD optimized scheduling strategy defined in [114]. The solution in [114] does not consider any a priori expectation about future navigation commands and, at any time, is optimal regarding the quality improvement obtained per transmitted bytes in the current WoI. We refer to it as the static WoI (SW) solution because it computes schedules only based on the current WoI knowledge, without taking the dynamic nature of the system into account. In practice, NP and SW follow the exact same scheduling strategy as long as there remains a current WoI packet to schedule. After that initial period, NP takes advantage of its knowledge of the navigation model to schedule data that are likely to be requested in the future, whilst SW goes idle[1].



(a) $\delta = 0.2$ (b) $\delta = 0.9$

Figure 3.2: *Comparison of natural pre-fetching (NP) and static WoI (SW) RD optimized schedules. Each curve plots the average time elapsed between two consecutive navigation commands by the two schedulers, as a function of the reaction delay parameter B_τ, and for two distinct Θ values (30 and 40 dB).*

Figure 3.2 compares both solutions in terms of the average time elapsed between consecutive navigation commands, measured by the number of bytes that could be transmitted by the channel during that period of time[2].

[1]Note here that idleness might be an advantage when the browsing cost is measured in terms of channel bytes rather than in terms of browsing time, as considered in our study.

[2]This does not necessarily reflect the actual number of bytes that are forwarded by

As expected, we observe that the reactivity improvement of NP over SW increases with the reaction delay B_τ and with the quality threshold Θ. Indeed, a large B_τ gives more time to download data after having reached Θ, while a high Θ means that less current WoI data has still to be downloaded during the reaction delay. As NP only performs pre-fetching once all current WoI packets have been scheduled, high B_τ and Θ increase the fraction of B_τ devoted to pre-fetching. Moreover, we observe that for such high B_τ and Θ, improvement brought by NP is much more significant when predictability is high. Indeed, a highly predictable navigation makes the pre-fetching more efficient.

We observe also that for small B_τ and/or small Θ, improvement of NP seems better for low predictability. We explain this by the fact that the cache removal strategies used in both cases have different impact whether δ is small or high. For SW, if space is needed, all packets that do not belong to the current WoI are removed from the cache. For NP, they are removed according to their expected interest in future WoIs. On one hand, when δ is low, the chosen navigation model tends to leave the user in the same spatial area. The SW removal strategy is not suited in this case as high resolution packets might be removed while they are still very likely to be needed in a near future. On the other hand, when δ is high, the navigation model will browse the image in the same direction for a while. In this case, both removal strategies are equivalent and NP outperforms SW only when it has enough time to pre-fetch data, i.e. for large B_τ and Θ.

From a computational point of view, a scheduling strategy that envisions to pre-fetch data is more complex than SW because (1) it selects the packet to forward within a larger set, including both the current WoI packets and the packets that lie outside the current WoI but are likely to be requested in the future, and (2) it computes the expected distortion based on the multiple terms defined in Equation (3.7), and not on the single term of Equation (3.3).

Accepting the computational overhead, we now provide a deeper analysis of the 3 scheduling strategies proposed to address the dynamic nature of the browsing system. Our purpose is to understand the impact of the navigation and reaction model parameters on the advantage taken from anticipated pre-fetching, compared to natural pre-fetching.

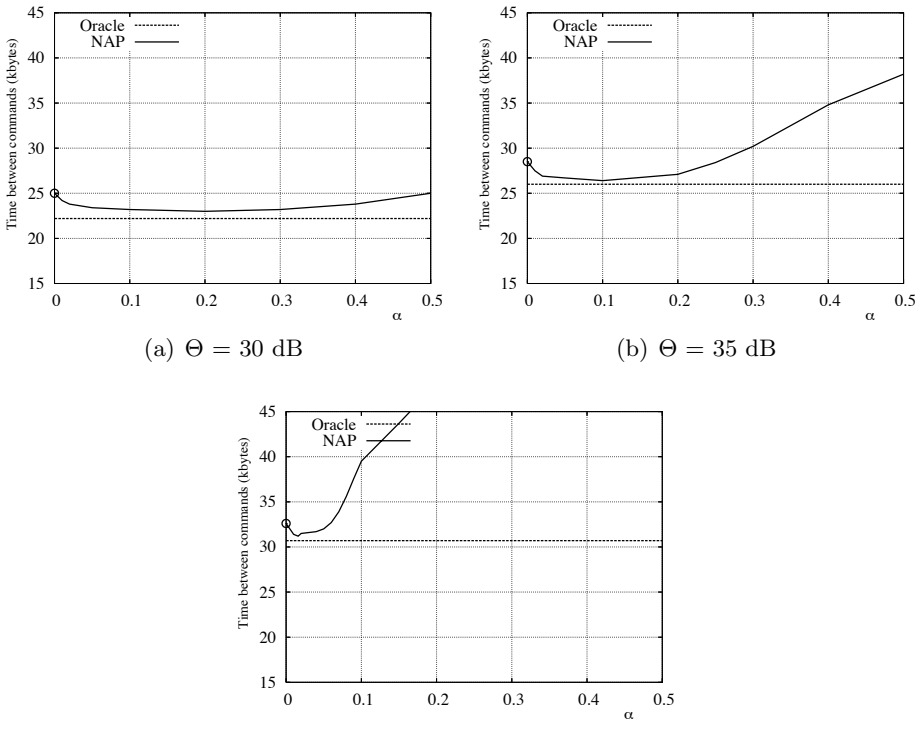
the scheduler, as SW goes idle once all the current WoI data have been sent out.

3.5.2 Anticipated vs. natural pre-fetching

This section compares the 3 proposed schedulers in terms of browsing session reactivity. All along the section, we use unitary transmission and memory units, so that the storage and transmission costs of a packet are identical.

Figures 3.3 and 3.4 plot the average time, expressed in number of transmitted bytes, between consecutive navigation commands for the 3 schedulers, as a function of the α pre-fetching parameter. The oracle scheduler does not depend on α and corresponds to a horizontal dashed line. The natural pre-fetching solution corresponds to $\alpha = 0$, and is depicted by a circle. The solid line curves plot the anticipated pre-fetching solution. Figure 3.3 considers a poorly predictable navigation model ($\delta = 0.2$), while Figure 3.4 corresponds to a highly predictable model ($\delta = 0.9$). In these figures, we note that the oracle scheduler performs significantly better than the natural pre-fetching solution (about 30% bit savings for $\delta = 0.9$). We also observe that for a well-chosen α parameter, the anticipated pre-fetching mechanism also improves the natural pre-fetching solution. Regarding the selection of the α parameter, we first observe that the α value that minimizes the number of bytes per command decreases as the quality threshold Θ increases. It makes sense because a small (large) α postpones (speeds up) the anticipation of future WoIs, and a large Θ means that a high quality is required in the current WoI before moving to the next user request. Moreover, we observe that the solid line goes up faster for increased α when δ is small. It means that the anticipated pre-fetching is much less sensitive to the α selection when the navigation model is predictable, i.e. when $\delta = 0.9$. In particular, for a reliable prediction of future WoIs, as long as α remains lower than 0.5, the anticipated pre-fetching outperforms the natural pre-fetching. We conclude that it is easier and less risky to implement anticipated pre-fetching when future WoIs are reliably predicted than in presence of a random navigation.

In Figure 3.4, we finally observe that anticipated pre-fetching (AP) is especially beneficial, i.e. comes close to the oracle, for small Θ values (typically below 35 dB). We explain that observation by the rate-distortion distribution of JPEG 2000 packets, i.e. by the fact that a typical JPEG 2000 code-stream contains few (a large number of) packets providing a large (small) gain in quality. As a consequence, for small Θ values, the packets that are required to reach the current WoI Θ threshold provide a significantly larger gain on the current WoI than the ones that are not needed to meet the Θ constraint. The required packets are thus easy to discriminate

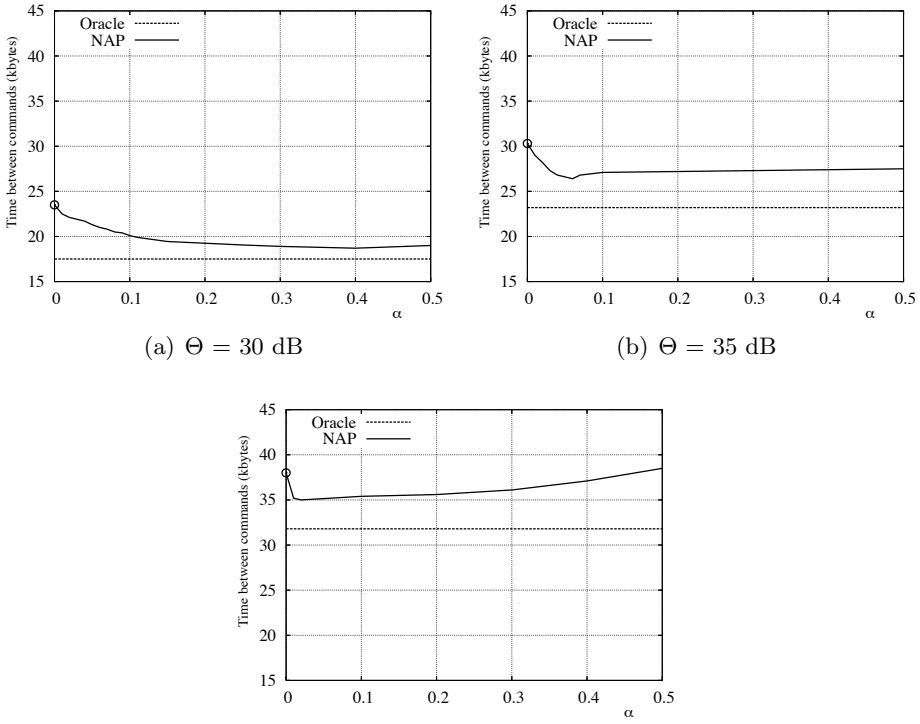


(a) $\Theta = 30$ dB

(b) $\Theta = 35$ dB

(c) $\Theta = 40$ dB

Figure 3.3: *For an unpredictable navigation model ($\delta = 0.2$), the graphs plot the average time -measured in number of transmitted bytes- between consecutive navigation commands, for the oracle scheduler (dashed line), the natural pre-fetching (circle) and the anticipated pre-fetching (solid line), as a function of the α pre-fetching parameter. NAP refers to natural and anticipated pre-fetching. Graphs are plotted for a user reaction model defined by a reaction delay $B_\tau = 10$ kbytes and by 3 different quality thresholds Θ = 30, 35 and 40 dB. The client buffer memory has a size equal to M = 150 kbytes.*



(a) $\Theta = 30$ dB

(b) $\Theta = 35$ dB

(c) $\Theta = 40$ dB

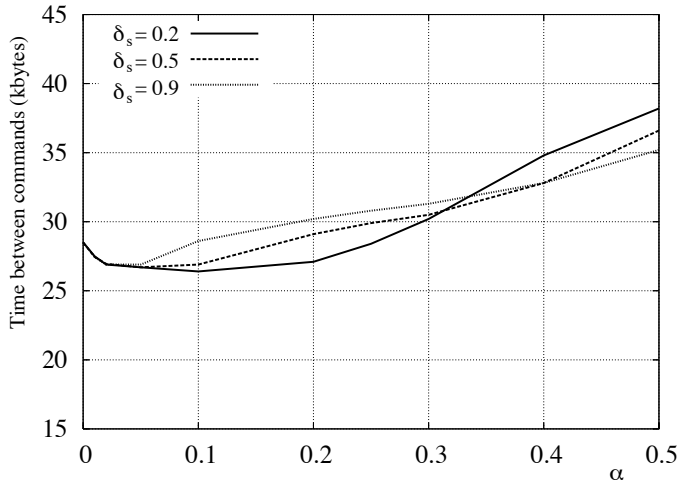
Figure 3.4: *For a predictable navigation model ($\delta = 0.9$), the graphs plot the average time elapsed between two consecutive navigation commands, for the oracle scheduler (dashed line), the natural pre-fetching (circle) and the anticipated pre-fetching (solid line), as a function of the α pre-fetching parameter. NAP refers to natural and anticipated pre-fetching. Graphs are plotted for a user reaction model defined by a reaction delay $B_\tau = 10$ kbytes and by 3 different quality thresholds $\Theta = 30$, 35 and 40 dB. The client buffer memory has a size equal to $M = 150$ kbytes.*

based on Equation (3.8), even for a large α parameter, which is an advantage regarding the ability to schedule future expected packets once the required current WoI packets have been forwarded. In contrast, for a larger quality threshold, it is not possible to find a weighting factor α so that AP mimics the oracle behavior, i.e. strictly forwards current WoI packets as long as the quality threshold is not achieved and focuses on future expected WoIs afterwards. Hence, in that case, it is recommended to acquire a better knowledge of the user reaction model so as to design a dedicated scheduler that performs closer to the oracle.

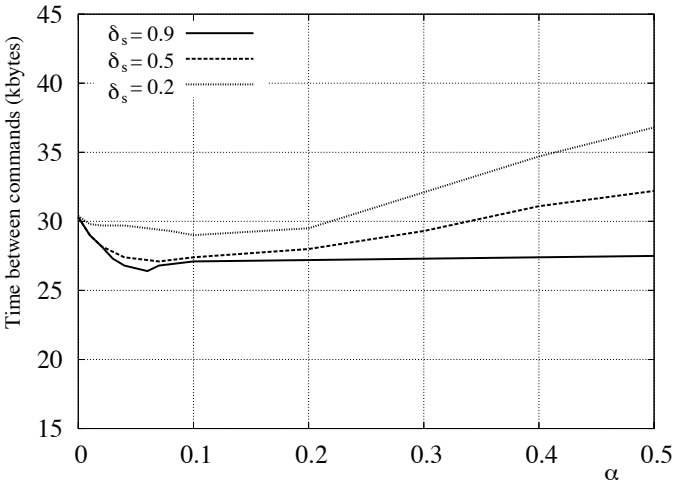
In Figure 3.5, we analyze the impact of an error while estimating the navigation model parameter. Specifically, we consider a mismatch between the actual δ parameter of the navigation model, and its estimate at the scheduler, denoted δ_s. In Figures 3.5 (a) and (b), we mainly observe that the minimum of the curves plotted as a function of α are not significantly affected by the δ parameter estimation error. Hence, we conclude that, upon appropriate α parameter selection, the performances of anticipated pre-fetching are preserved in presence of realistic errors of the navigation model parameter estimation.

Figure 3.6(a) depicts the relative browsing time overhead associated to natural and anticipated pre-fetching, compared to the oracle scheduler. The graph plots the overhead for a predictable ($\delta = 0.9$) and a random ($\delta = 0.2$) navigation model, as a function of the quality threshold Θ. For anticipated pre-fetching, the α value is chosen as the one that minimizes the overhead. We first observe that the overhead increases with the model predictability. It makes sense because the anticipation of future WoIs is expected to be more efficient when prediction is more reliable and, therefore, the performance of the oracle further increases. In that case of interest, for which the navigation is predictable ($\delta = 0.9$), we also observe that the relative overhead of natural pre-fetching (NP) over the oracle scheduler decreases as Θ increases, due to the increased fraction of B_τ devoted to pre-fetching, but also to the increase of the total number of bytes transmitted by the oracle. The fact that NP performs relatively better at large quality thresholds is an interesting observation as we have shown in Figure 3.4 that AP is less efficient at large than small Θ values.

Figure 3.6(b) presents the browsing time overhead computed for the natural and the anticipated pre-fetching, relative to the oracle scheduler. Graphs are plotted for two distinct navigation models ($\delta = 0.2$ and 0.9), as a function of the number of bytes B_τ transmitted during the reaction period τ. We observe that natural pre-fetching (NP) overhead rapidly increases

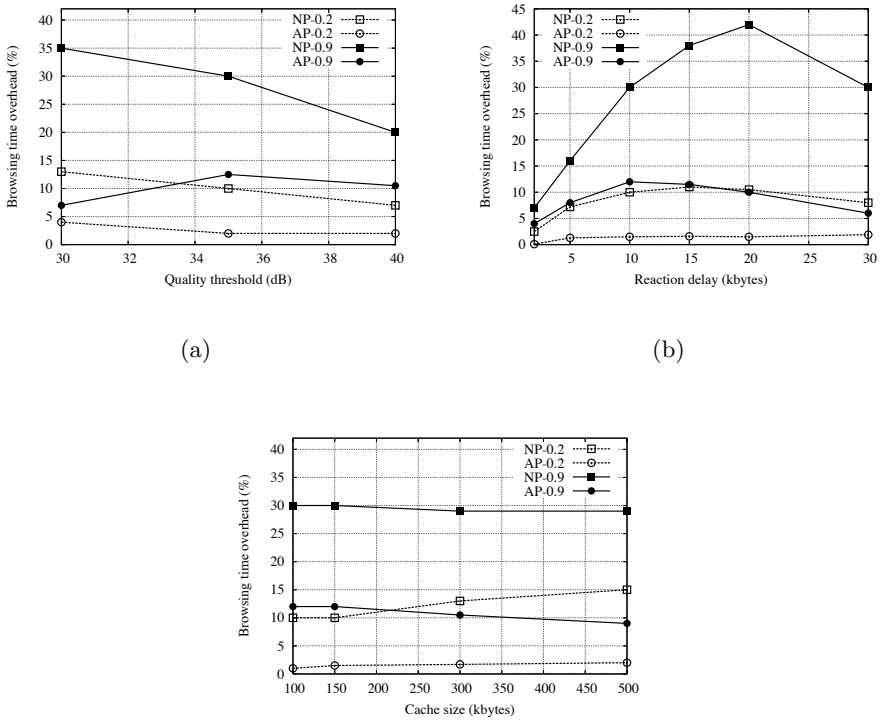


(a) $\delta = 0.2$



(b) $\delta = 0.9$

Figure 3.5: *Impact of an error performed while estimating the navigation model parameter. δ denotes the actual navigation model predictability. δ_s refers to the value of predictability estimated at the scheduler. Each curve plots the average number of bytes transmitted between two consecutive navigation commands by the anticipated pre-fetching scheduler, as a function of the α pre-fetching parameter.*



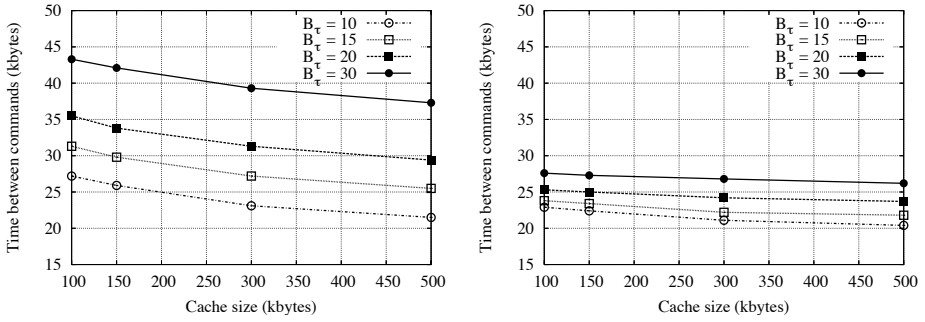
(a) (b)

(c)

Figure 3.6: *Overhead in average number of bytes between consecutive navigation commands for natural (NP) and anticipated (AP) pre-fetching, compared to the oracle scheduler. Graphs are plotted for a predictable ($\delta = 0.9$) and a random ($\delta = 0.2$) navigation models, as a function of (a) the quality threshold Θ, (b) the reaction delay B_τ and (c) the cache memory size. When not displayed in the x-axis, following parameters are used: $\Theta = 35$ dB, $B_\tau = 10$ kbytes, cache memory size $= 150$ kbytes.*

as B_τ increases. It reflects the incapacity of natural pre-fetching to take advantage of the reaction delay. That incapacity is even more penalizing when future WoIs are predicted accurately, i.e. when $\delta = 0.9$. In contrast, anticipated pre-fetching (AP) overhead remains small and about constant with B_τ. It demonstrates that anticipated pre-fetching performs close-to-optimal pre-fetching during the reaction delay, whatever the reaction delay duration or the model predictability. For completeness, note that for large B_τ values, NP gets the opportunity to pre-fetch future expected data (after all current WoI data have been scheduled). Hence, its performance gets progressively closer to the AP and oracle schedulers. In Figure 3.6(b), the converging trend is accentuated by the fact that we express the overhead in relative terms, compared to the oracle browsing time, which increases with B_τ.

Figure 3.6(c) depicts the browsing time overhead computed for the natural and the anticipated pre-fetching, relatively to the oracle scheduler. Graphs are plotted for two distinct navigation models ($\delta = 0.2$ and 0.9), as a function of the client cache memory size. We observe that the relative overhead does not significantly change with the cache size, which indicates that the three proposed scheduling algorithms get a similar benefit from an increased client cache size.



(a) $\delta = 0.2$ (b) $\delta = 0.9$

Figure 3.7: *Average number of bytes transmitted between two consecutive navigation commands, for the oracle scheduler and for a set of reaction delays B_τ, as a function of the client buffer memory size. Θ is set to 35 dB. The navigation predictability δ is respectively set to 0.2 and 0.9 in Figures (a) and (b).*

To figure out the actual impact of the cache size on the performance of

the browsing system, Figure 3.7 considers the oracle scheduler, and analyzes how the number of bytes transmitted between consecutive commands evolves with an increased cache memory, for two different predictability levels of the commands ($\delta = 0.2$ and 0.9), and for a set of reaction delays ($B_\tau \in [10, 30]$ kbytes). We first observe that the slope of the curves is larger in Figure 3.7 (a) than in Figure 3.7 (b), which indicates that the predictable navigation model (δ close to one) takes less advantage from an increased size of the cache memory. We explain that observation by the fact that a random navigation model ends up in browsing the image more or less uniformly and in filling the cache with data that are relatively spread on the image. Cached data have thus a good chance to benefit future requests of the user. In contrast, for our navigation model, the predictable behavior has a great chance to browse the image in the same direction for a while, so that it requests a majority of "fresh" image data at each novel navigation command. Beside the cache utility issue, we observe that the space between the curves plotted for distinct B_τ values is larger in Figure 3.7 (a) than in Figure 3.7 (b). It reflects the fact that a predictable navigation takes advantage of the reaction delay by transmitting data that are really useful to the next WoIs. In contrast, when the next WoI is poorly predicted, most of the transmission resources spend during the reaction period are wasted. They contribute to increase the overall bit-budget without really improving the initial quality for the next WoI. Pre-fetching is thus inefficient in that case.

We can summarize the results of our simulations as follows. As expected, the oracle scheduler always performs better than the two other schemes. We also observe that, compared to the oracle scheduler, the natural pre-fetching mechanism is further penalized as the user reaction delay increases, and as the navigation model becomes more predictable (δ gets close to one). These results reflect the fact that the oracle's ability to anticipate future WoIs provides a benefit that increases with the latency of the system and with the prediction accuracy of future WoIs. In addition, our simulations demonstrate that, for an appropriate α, anticipated pre-fetching always outperforms natural pre-fetching. Moreover, it gets quite close to the oracle scheduler for small quality thresholds Θ, and appears to be less sensitive to the α parameter when the navigation gets more predictable. Regarding the client cache memory, our simulations have revealed that all scheduling approaches get a similar benefit from an increased memory size. However, we observe that a large cache becomes more important as future navigation commands get less predictable (for small δ). For such

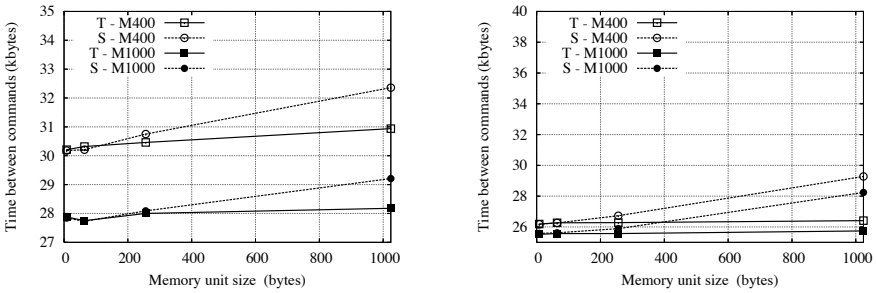
unpredictable navigation models, the use of anticipated pre-fetching, or even the accurate estimation of the user reaction model, is less beneficial than a large client cache memory.

Based on these observations, we infer the following guidelines regarding the design of an interactive browsing system. First, when the navigation is poorly predicted, the use of anticipated pre-fetching should be discouraged, and the search for an accurate definition of the reaction model, so as to get the best out of the oracle scheduler, is definitely not crucial. In that case, a simple conventional scheduling mechanism, that is natural pre-fetching, combined with a large cache memory, is expected to achieve satisfying performance. On the contrary, when the navigation commands get highly predictable, the anticipation of future WoI becomes crucial, and that is even more the case when the latency of the system (due to transmission, decoding or interpretation) increases. Based on the simulation results, we differentiate two distinct scenarios when the navigation is reliably predicted. First, when the quality needed by the user to take a decision about the next navigation command is rather small (typically 30 to 35 dB), the use of anticipated pre-fetching is strongly recommended because (i) it provides large gains in bit-budget (up to 30 %) compared to natural pre-fetching, (ii) it achieves close to optimal performance, and (iii) it is easy to implement and tune (small sensitiveness to the α parameter). In contrast, when the quality needed to decide about the next command is high, whilst remaining significantly beneficial compared to natural pre-fetching, the anticipated pre-fetching performs significantly worse than the oracle scheduler. Hence, in that particular case, it is recommended to attempt to get a better knowledge of the user reaction model, so as to better approximate the oracle behavior.

3.5.3 Memory- vs. channel- optimized scheduling

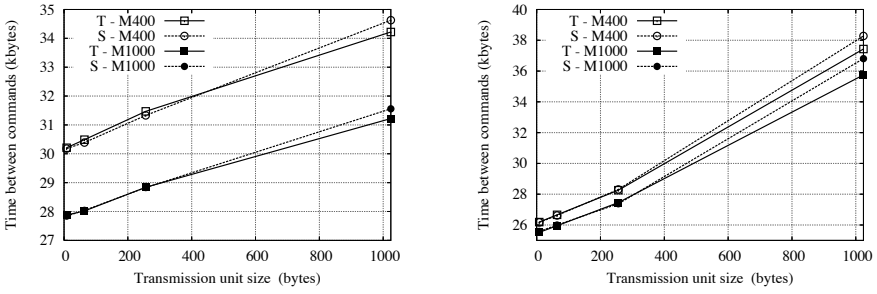
In Section 3.4.3, we have motivated the use of memory-optimized schedulers. We now provide some experimental results that compare memory- and channel- optimized schedulers, based on the responsiveness of the browsing session.

Figure 3.8 and 3.9 compare both oracle solutions in terms of the average time elapsed between consecutive navigation commands. As above, a period of time is measured by the number of bytes that the channel could transmit during that time. In Figure 3.8 the transmission unit size has been set to 8 bytes, and curves are drawn as a function of the memory unit size, for two navigation models ($\delta = 0.2$ and 0.9) and two memory sizes (400 and



(a) $\delta = 0.2$ (b) $\delta = 0.9$

Figure 3.8: *Amount transmission resources spared between consecutive commands as a function of the memory unit size, when transmission unit size is set to 8 bytes. Graphs are plotted for two navigation models, characterized by* $\delta = 0.2$ *and* 0.9*. S and T labels correspond to scheduling strategies respectively based on the storage and transmission cost. MX defines a total memory size of X kbytes.*



(a) $\delta = 0.2$ (b) $\delta = 0.9$

Figure 3.9: *Average transmission resources spared between consecutive commands as a function of the transmission unit size, when memory unit size is set to 8 bytes. Graphs are plotted for two navigation models, characterized by* $\delta = 0.2$ *and* 0.9*. S and T labels correspond to scheduling strategies respectively based on the storage and transmission cost. MX defines a total memory size of X kbytes.*

1000 kbytes). We observe that in nearly all cases, it is better to schedule packets as a function of their cost in transmission (T) rather than in storage (S). The only scenario for which S brings a (quite) small advantage over T corresponds to an unpredictable navigation model ($\delta = 0.2$), using a small memory size (400 kbytes), with a small memory unit size. This observation makes sense because (i) the cache memory has been shown to be especially useful when the navigation model is unpredictable (see Section 3.5), (ii) an efficient usage of memory is even more important when the memory size gets smaller, and (iii) a schedule based on the storage cost strongly penalizes the increments of quality per unit of transmission when there is a significant mismatch between the sizes of memory and transmission units.

Figure 3.9 confirms this conclusion. The memory unit size has now been set to 8 bytes while the transmission unit size ranges from 8 to 1024 bytes. As above, we observe that scheduling packets in terms of their transmission cost generally provides the highest responsiveness. Again, we observe that S only improves over T for small memory size and when $\delta = 0.2$. In particular, we note that the improvement occurs for a moderate mismatch between transmission and memory units, typically around a transmission unit size of 256 bytes. We explain this last observation as follows. When the transmission unit size increases and packets are scheduled based on their transmission cost, some small packets are likely to be discarded from the transmission schedule despite their significant quality contribution, because they do not fill an entire transmission packet and thus waste transmission resources. As a consequence, these packets never reach the client cache, which penalizes the initial WoI quality provided by the cache when a novel navigation command is released. This phenomenon explains why the S curve goes below (and thus outperforms) the T curve in Figure 3.9, for a small transmission unit size. However, in most other cases, the T strategy outperforms the S one, mainly because it maximizes the growth of quality achieved per unit of transmitted data while bridging the gap between the initial cached quality and the required threshold Θ. Finally, based on these observations, we recommend to schedule packets based on their transmission cost.

For completeness, Figure 3.10 compares the browsing responsiveness for two distinct cache freeing strategies, when packets are scheduled based on their transmission costs. The two envisioned strategies respectively discard the packets from the cache based on their transmission and storage cost. As expected, we observe that removing the packets based on the storage cost maximizes responsiveness, because it minimizes the quality

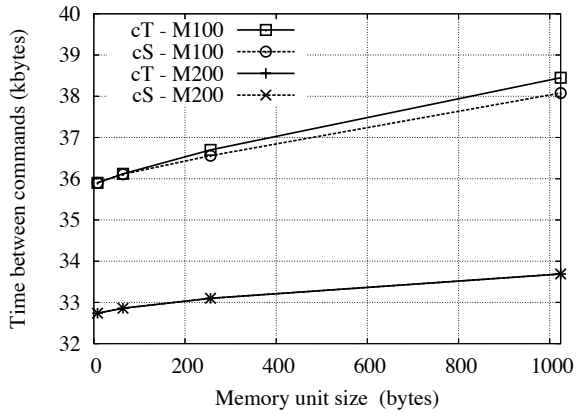


Figure 3.10: *Average time between consecutive commands for two distinct cache packets removal strategies. The cT and cS labels respectively mean that packets are discarded from the cache based on their transmission and storage cost. The graph is plotted as a function of the memory unit size, when transmission unit size is set to 8 bytes. MX defines a total memory size of X kbytes. For all curves, incoming packets are scheduled by the oracle scheduler, based on their transmission cost. The navigation model is characterized by* $\delta = 0.2$*, and* B_τ *has been set to 20 kbytes.*

loss per freed memory units. However, a difference between both packet removal strategies only appears for small memories. It makes sense because a large memory is expected to contain packets that do not bring any benefit to the current (and future) WoI and that are removed whatever the cost function is. In practice, this observation also means that, in most practical cases, packets could be equivalently removed based on their transmission cost, which preserves rate/distortion convex-hull optimality for each precinct contained in the cache, and thus slightly simplifies the scheduler implementation (see Section 3.4.3).

3.6 Conclusions

This chapter considers the remote interactive browsing of JPEG 2000 images. Our study focuses on the dynamic nature of the system, i.e. on the fact that the sequence of windows of interest (WoIs) is defined interactively

by the client in response to the received data. A major contribution of our work has been to propose and evaluate pre-fetching solutions that anticipate future navigation commands by scheduling JPEG 2000 packets that are expected to become relevant in future WoIs. Our results demonstrate that, by speeding up the increase of quality in future WoIs, pre-fetching has the capacity to accelerate the rhythm of user requests and in turn, to increase the responsiveness of the browsing system. The fundamental reason that makes the anticipation of future WoIs helpful is the existence of a delay between the transmission of information by the server, and its translation into a novel WoI request by the user. This *reaction delay* is due to the transmission and rendering latency, but also to the time needed by a user to analyze and interpret the displayed information.

We have compared three schedulers based on extensive simulations, and have identified the browsing scenarios for which pre-fetching and caching strategies impact the browsing system reactivity. As an outcome of our study, we draw the following guidelines to design an interactive remote image browsing system.

- First, natural pre-fetching, which only pre-fetches future expected WoI data after all current WoI data have been sent out, provides a simple and satisfying scheduling solution when future navigation commands are hardly predictable. Moreover, in that case, a large client cache memory appears to be beneficial.

- In contrast, when future WoIs can be predicted reliably, the client cache size becomes less crucial, but it becomes important to derive scheduling policies that are able to anticipate future WoIs. This statement becomes even more relevant as the reaction delay of the browsing system increases. In practice, to anticipate future WoIs, a simple scheduling strategy that transmits packets in decreasing order of the weighted benefit provided on the current and future WoIs achieves close to optimal performance, nearly independently of the weighting factor, when the level of quality required by the user to trigger subsequent commands is reasonable. Hence, it appears that sophisticated scheduling mechanisms, based on the exploitation of a deep and formal knowledge of the user reaction model, are only recommended when future commands are highly predictable and when a high quality is needed by the user to take a decision about the next WoI. This result is interesting because, in practice, the reaction model is expected to be difficult to estimate and formalize, especially because

the reactivity of a user in front of the information displayed in a WoI depends on complex and subjective interpretation abilities.

Finally, we have also demonstrated that scheduling data as a function of their storage cost rather than transmission load achieves optimal allocation of the memory space, but only rarely improves the browsing system reactivity.

Compressed Image Retrieval

4

Coarse-to-fine image classification in the JPEG 2000 compressed domain for fast processing of large datasets.

In this chapter, we address the issues of analyzing and classifying JPEG 2000 code-streams. The presented techniques could be useful in any application that involves (semi-)automatic image categorization or retrieval tasks, such as audiovisual archives retrieval, computer-assisted medical diagnosis, remote sensing on satellite images, etc. We first study the kind of features that can be extracted from a JPEG 2000 code-stream and at what cost. An original representation, called integral volume*, is proposed to store the features values. It can be progressively built and allows an easy computation of these values on any spatial image area, regardless of code-block borders. Then, a JPEG 2000 classifier is presented, that uses integral volumes to learn an ensemble of randomized trees. Several classification tasks are performed on various JPEG 2000 image databases and results are close or even better than the ones obtained in the literature with non-compressed version of these databases. Finally, a cascade of such classifiers is introduced, in order to specifically address the image retrieval issue, i.e. bi-class problems characterized by a highly skewed distribution and by a large amount of test samples compared to learn samples. An efficient way to learn and then optimize such cascade is proposed. We show that staying in a JPEG 2000 framework, initially seen as a constraint to avoid heavy decoding operations, is actually an advantage as it can benefit from the multi-resolution and multi-layer paradigms inherently present in the compression standard. The work presented in this chapter has lead to a conference paper [22], and a journal paper is being finalized and will be submitted soon.*

4.1 Introduction

As already stated in chapter 1, today's imaging applications have to deal with an ever-growing amount of digital data [39]. In order for this huge amount of information to be manageable and useful, it requires on one hand an efficient and flexible representation for the images. On the other hand, application-specific techniques exploiting this flexibility are needed to guide the user in the data space and assist him in his task. In this thesis, we focus on one flexible representation, namely JPEG 2000, and on two kinds of tools helping the user when he interacts with an image database, namely *browsing* and *retrieval* tools. Browsing was investigated in the previous chapter: we proposed techniques and gave advices to enable an efficient and smooth remote navigation across large JPEG 2000 image data spaces. Image retrieval consists in techniques enabling the user to efficiently and automatically find (portions of) images that are relevant for his current task. Such techniques will be investigated in the present chapter.

Following the red wire of this thesis, we focus in this chapter on image retrieval in the JPEG 2000 domain. In the future, it is very likely that the vast majority of stored and transmitted digital data will be compressed rather than raw. Consequently, being able to search for relevant images directly within compressed bit-streams (or through a partial decompression) has an obvious advantage in terms of computational load. Moreover, we show in this chapter that JPEG 2000 is well suited for image retrieval tasks. Indeed, its various scalability levels -already very useful for remote browsing- allow for true and efficient coarse-to-fine searches among the bit-streams. Furthermore, the JPEG 2000 wavelet transformation, designed for compression purpose, has also a strong discriminant power and is suited to analyze image characteristics like edges and textures. The reader unfamiliar with JPEG 2000 is therefore invited to refer to Chapter 2 if some concepts used here and related to JPEG 2000 are misunderstood.

Among the broad range of imaging applications involving classification or retrieval tasks, we will focus on those where areas of interest are characterized by an ensemble of different textures. Texture has to be understood here in the broad sense of a "set of local neighbourhood properties of the gray levels of an image region" [63]. "Grays levels" is even too restrictive in our case as color information (at least in the low frequencies) will also be used in the retrieval process. While this choice might appear to exceedingly narrow the set of potential applications, we will show that many objects can actually be characterized by an ensemble of textures. This is even truer if the relative spatial position of the textures composing an ob-

ject is taken into account, as it will be the case in the proposed system. Concerning potential applications, we could cite at least three of them, for which professionals from related fields have explicitly shown an interest in our work:

1. *Remote sensing.* Example: while remotely browsing a satellite image, an agronomist wants to automatically retrieve all the image areas corresponding to a specific kind of cultivation (corn, wheat, ...) or wood (leafy trees, conifers, ...).

2. *Medical applications.* Example: receiving an unknown radiography, a doctor wants to retrieve in his database similar images that are already diagnosed, in order to help him make his mind on the unknown one.

3. *Audio-visual archives.* Example: a user wants to retrieve in an audio-visual database all the scenes shooted in a specific kind of location (a desert for example), or all the scenes from a specific movie with at least two people appearing on the screen.

Focusing on texture-based retrieval techniques, we will not consider shape matching algorithms, like the ones presented in [4] or, in a JPEG 2000 framework, in [50]. However, these algorithms could efficiently complement the proposed retrieval system and are, as such, an interesting future research direction, as explained in the conclusion of this chapter.

Eventually, we give us an additional working assumption: experiments carried out in this chapter always use a learning set (ensemble of labeled images) to train classifiers. Performances achieved by these classifiers are then assessed with a test set (ensemble of images with labels that the retrieving system is not aware of). This is a first step towards a user-centered approach with relevance feedback and query-by-example method: in such framework, the user would progressively build the learning set by giving positive and negative examples and labeling examples selected by the system, while the system would progressively build the classifiers based on the user feedback. Results obtained in this chapter remain valid for this kind of system and the presented JPEG 2000 coarse-to-fine approach is even very well suited for it.

4.1.1 Contributions overview

Three main contributions are presented in this chapter. We will introduce each of these ideas briefly and then describe them in detail in subsequent sections.

The first contribution is a **method to extract, process and store a hierarchy of features from a JPEG 2000 code-stream**. Both header-based [107] and wavelet-based [50] features are used. For a given image area, it extracts global and normalized data that are expected to be representative of this area. In the proposed method, these global features are computed on local areas inside an image sample. In this way, relative spatial position of several groups of pixels -each characterized by a certain global feature value- can be taken into account and used for shape description. Moreover, as all extracted values are normalized relatively to the size of the considered area, samples of different sizes can be compared together. To store all the extracted JPEG 2000 features values, we extend the concept of integral image first introduced in [14] and more recently made famous by Viola and Jones in [119]. We propose to represent a JPEG 2000 code-stream with a chain of *integral volumes*. This representation draws benefit from the JPEG 2000 scalability levels and allows for a very fast features computation on any image area.

The second contribution is the combination of integral volumes mentioned above with ensemble of random decision trees [41] to build an **efficient and robust JPEG 2000 image classifier**. Based on the work by Marée *et al.* [71], we take advantage of the integral volume structure to generate random samples in the images. We show that chosen features lead to satisfying classification results on several different image databases.

The last contribution is the tight integration of the JPEG 2000 scalability levels for designing a **cascade of JPEG 2000 image classifiers**. Cascade of classifiers are well suited for image retrieval tasks and object detection. When one has to mine into a compressed image database, the challenge is to retrieve relevant areas while minimizing the amount of decompressed data. In this work, starting from the coarse classification that can easily be obtained from header-based features, we use the resolution and bit-depth scalability of JPEG 2000 to progressively extract more information and refine the classification with wavelet-based techniques. For a given image area, the amount of data that will need to be entropy-decoded is therefore directly related to the relevance of this area in the retrieval process. This idea is directly inspired from the work of Geman and Fleuret [35] and more particularly from the one of Viola and Jones [119]. Concurrently

to [5], we proposed in [22] to improve the Viola and Jones cascade by taking into account the extraction cost of a feature. This leads us to an interesting "complexity[1] *vs* performances" trade-off when designing a cascade of classifiers.

4.1.2 Chapter organization

The remainder of the chapter is organized as follows. Section 4.2 details which features are used and how they are extracted from JPEG 2000 code-streams. In particular, the scalability of the extraction process is underlined. In Section 4.3, we introduce the concept of integral volume to store these features. In Section 4.4, the proposed JPEG 2000 image classifier is presented and applied to several image classification problems (n-class problems with $n > 2$). Various freely available databases are used and results are compared to related existing systems. A cascade of those classifiers is then introduced in Section 4.5 and used to address a texture retrieval problem (2-class problem) in a computationally efficient way. Finally, Section 4.6 summarizes the chapter and gives some future research directions.

4.2 JPEG 2000 features extraction

Numerous papers have been written on the use of JPEG 2000 code-streams for retrieval tasks. Each of them proposes a set of features that can be more or less easily extracted from the code-stream and that is used in a classification process (in most cases this process is a similarity measure based on a certain distance metric). As we shall see, the proposed approach uses features with similar discriminant power than the ones proposed in the related literature. However, in order to address more realistic classification problems, we propose an efficient way to process and store these features, called *integral volumes*. Among other advantages, this representation allows to easily take into account spatial relative position of groups of coefficients.

4.2.1 Feature types

Two kind of features can basically be extracted from a JPEG 2000 code-stream: the ones from the packet headers that do not involve any partial

[1] This complexity being computed as the total amount of data that has to be decompressed in the retrieval process.

decompression, and the ones based on the wavelet coefficients that require an entropy-decoding step.

Header-based features

As detailed in Chapter 2, a JPEG 2000 code-stream is made of a succession of packets. Each of them is uniquely defined by a set of 4 indices: p, c, r, l. These values mean that the entropy-coded data enclosed in the packet relates to l-th quality layer of the code-blocks belonging to a certain spatial area p ("p" for *precinct*), component c, and resolution r. Each packet has a header that contains relevant information about its content. Let us consider the b-th code-block of precinct p, component c and resolution r. Following relevant information can then be extracted from related packet headers:

- $N^{mq}_{b,p,c,r}$, the number of bytes used to entropically encode the code-block. This number shows the efficiency with which the MQ-coder did compress the wavelet coefficients belonging to this code-block. As such, it can be seen as a measure of the source entropy and gives therefore a good indication on the informational content of the code-block. In [107], this value is used as a texture classification tool and an event detection system in video, while in [81] they use it to select the most important parts of an image in cropping/scaling applications. As shown in Fig. 4.1, if L quality layers have been specified in the encoding options, $N^{mq}_{b,p,c,r}$ aggregates information from L packet headers:
$$N^{mq}_{b,p,c,r} = \sum_{l=1}^{L} N^{mq}_{b,p,c,r,l}.$$

- $N^{bp}_{b,p,c,r}$, the maximum number of significant bit-planes in the code-block. We can easily show that
$$N^{bp}_{b,p,c,r} = \lfloor \log_2(\|\mathbf{x}\|_\infty) \rfloor + 1$$
where $\mathbf{x}$ is the vector of wavelet coefficients from code-block (b, p, c, r) [68]. In [67], Mallat showed that the maximum modulus of wavelet coefficients (and, hence, also the $\log_2$ of this modulus) describes well the kind of singularity present in the image. This information can be found in the first packet header where (a part of) code-block b is included[1] (see Fig. 4.1).

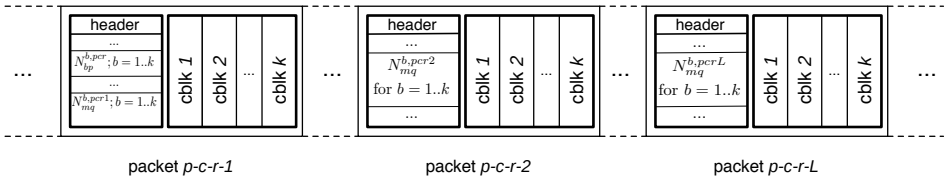


Figure 4.1: *Portion of a JPEG 2000 code-stream. When L quality layers are specified, the number of bytes used to encode a given code-block $N^{mq}_{b,p,c,r}$ must be aggregated from L packet headers. The number of significant bit-planes $N^{bp}_{b,p,c,r}$ can be computed from data in the first packet header where code-block b is included.*

$N^{mq}_{b,p,c,r}$ and $N^{bp}_{b,p,c,r}$ are both global values on a code-block. To evaluate N^{mq} and N^{bp} on a ensemble $\mathcal{B}$ of code-blocks, we simply aggregate values from each code-block:

$$N^{mq}_{\mathcal{B}} = \sum_{b \in \mathcal{B}} N^{mq}_{b} \tag{4.1}$$

$$N^{bp}_{\mathcal{B}} = \max_{b \in \mathcal{B}} N^{bp}_{b}. \tag{4.2}$$

In Fig. 4.2, we illustrate the discriminant power of these header-based features with two different textures from the Ponce texture database introduced in [52]. For a given subband, the feature values are aggregated over all code-blocks from this subband. As we see, N^{mq} and N^{bp} can be used to differentiate the two textures, as the one on the right contains significantly more high frequencies than the one on the left, leading to higher values in the high resolution levels.

In terms of extraction complexity, these features are of great interest as they do not imply any (partial) decompression of the JPEG 2000 code-stream. Headers from all relevant packets have only to be reached and processed by the retrieval system to collect the information. In comparison with a decompression operation, the computational complexity of this header processing step is negligible. In the following, we will consider only the number of entropy-coded bytes N^{mq}, as experiments have shown little improvement when using both kind of features.

[1]The value actually included in the header is the number of zero bit-planes to skip before reaching the first bit-plane with at least one significant coefficient. $N^{bp}_{b,p,c,r}$ is easily computed by subtracting this value to the global image bit-depth.

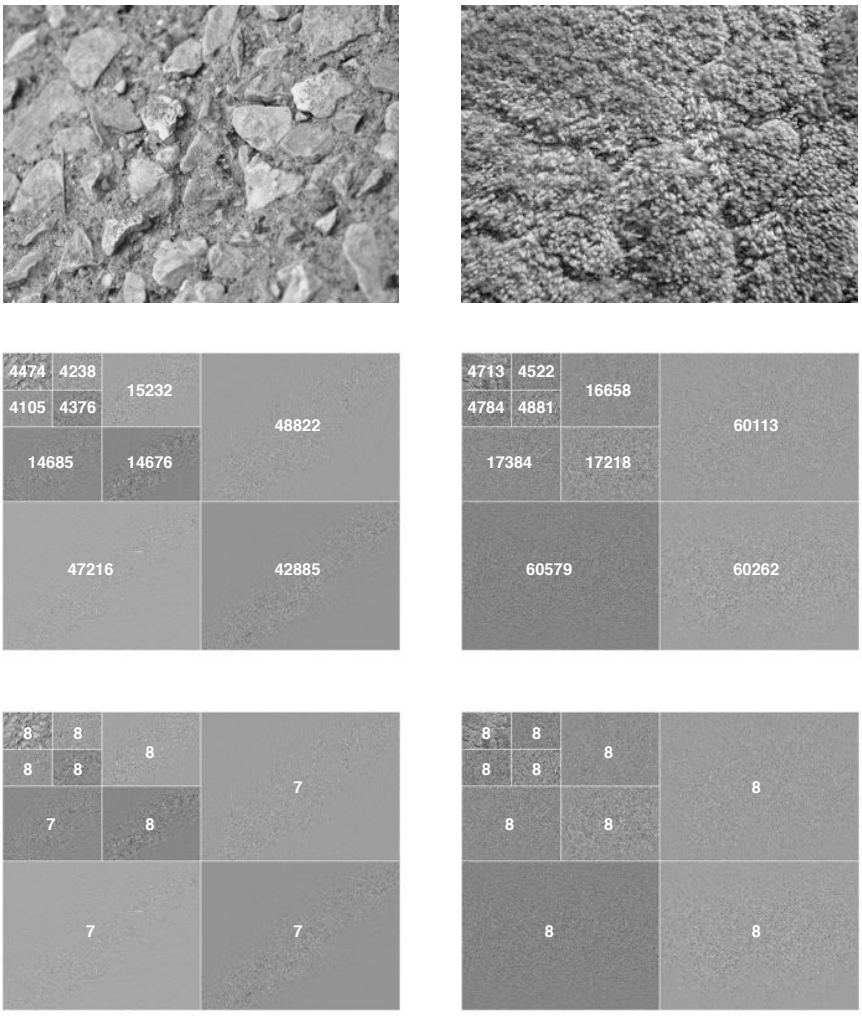


Figure 4.2: *Discriminant power of header-based features on two textures from the Ponce texture database introduced in [52]. Original images are on the first line. The second line shows the number of bytes used to encode each subband (N^{mq}). The third line shows the maximum number of bit-planes in each subband (N^{bp}).*

Wavelet-based features

Research on wavelet-based retrieval techniques is ongoing for many years now, and it started long before the JPEG 2000 standardization effort. Wavelets were indeed already made famous in the mid-eighties and quickly became a very useful tool in almost all signal processing applications [67]. As explained in Chapter 2, a wavelet decomposition of an image allows to separate high frequencies from low ones while keeping the spatial relationship between coefficients. Hence, this is a well suited representation for compression purpose (as image information is usually mainly concentrated in low frequencies) but also for discriminant analysis. The spatial-frequency representation provided by the wavelet transform gives indeed important information on the kind of local singularities present in the image [66]. In particular, textures end edges are well discriminated. Among the papers using wavelets in image retrieval systems, we identify two categories of features, that are subsequently used in a classification process:

- *the significance map of wavelet coefficients.* In [3, 50, 58, 68, 108], authors propose to use the significance status of each wavelet coefficient at a certain bit-plane as the discriminating feature. A coefficient x is said significant at a given bit-plane k if $x \geq 2^{k-1}$ for $k = 1, .., N^{bp}$. This significance map, processed with morphological operators and/or distance transform, can then be used in shape-matching applications [50].

- *statistical properties of the wavelet coefficients distribution.* In this case, feature vectors are built using different properties of the wavelet coefficients distribution in each subband. These global properties are particularly suited to discriminate different textures, as shown in Fig. 4.3: histograms of each subband are depicted for two different images. As we see, probability distributions are closely related to the kind of texture represented (in this case, histograms of high frequencies subbands from the left texture have a much higher peakedness than on the right). [69] and [30] propose to model this distribution with a Generalized Gaussian Density (GGD) and hence, to characterize an image with the vector aggregating the 2 parameters of a GGD in each subband. [115] follows the same approach using Gaussian Mixture Models instead of GGD. In [124], authors use variance of each subband to characterize an image while in [3] they use 2nd and 3rd moments of the significance map at a certain bit-plane.

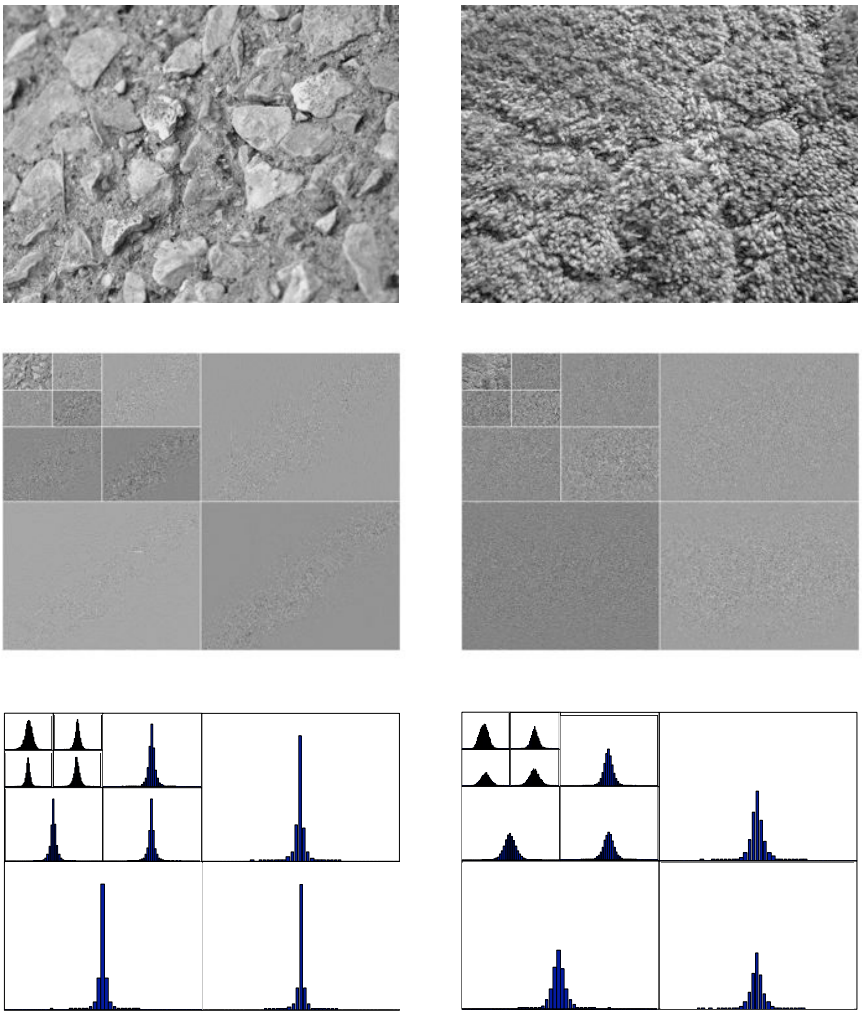


Figure 4.3: *Probability distributions for each subband of 2 different textures. To allow a fair comparison, histograms of every pair of corresponding subbands (one from each texture) have been scaled together to have the same y-axis. High frequencies subbands from the left texture contain less information than in the right texture, which explains the higher peakedness of corresponding histograms.*

In this work, we will focus on this second kind of wavelet-based features, namely statistical properties. Like in [30] and [115], we propose to characterize the probability distribution of the wavelet coefficients by a set of parameters. Starting from a JPEG 2000 code-stream, the extraction process of these parameters obviously imply an entropy-decoding step, as we have to get back to the wavelet domain. In order to avoid a computational overhead in addition to the already heavy decoding step, the feature computation itself has to be as simple as possible. For instance, this is not the case in [30] or [115] as several complex parameters for the gaussian distribution model have yet to be computed once all the coefficients have been decoded. In this work, a very simple distribution approximation is used: it does not require any subsequent heavy computation and, more importantly, it can be *progressively refined* during the decoding step. Let's consider a code-block with N^{bp} significant bit-planes and N coefficients. As a bit-plane from this code-block is being decoded, new significant coefficients are counted. For a given bit-plane k ($k = 1, ..., N^{bp}$), the number N_k^{sc} of new significant coefficients corresponds to the amount of wavelet coefficients x in the code-block such that

$$2^{k-1} \leq |x| < 2^k$$

It should be noted that we could easily differentiate positive and negative new significant coefficients. This will actually be done when processing code-blocks from the LL-subband. For all other (high frequency) subbands, the sign will not be considered. Indeed, given the shape of the wavelet function (the one usually denoted as ψ), the sign of the wavelet coefficients mainly discriminates signals based on their phase, which is not relevant in the considered classification tasks. It could also be possible to distinguish significant coefficients decoded by the significance pass from those coming from the clean-up pass. Theoretically, this should be pertinent as it gives valuable information on the significant status of the neighbors of the processed coefficient. However, experiments have shown no improvement when making this distinction. We therefore only consider the significance status of a coefficient, whatever its sign and encoding pass type. Once N_k^{sc} has been computed for $k = 1, ..., N^{bp}$, let's define a partition of the coefficients into $N^{bp} + 1$ disjoint intervals $\{S_0, ..., S_{N^{bp}}\}$ such that

$$S_k = \{x : 2^{k-1} \leq |x| < 2^k\} \text{ for } k = 1, .., N^{bp} - 1 \quad (4.3)$$

$$S_0 = \{x : |x| = 0\} \quad (4.4)$$

$$S_{N^{bp}} = \{x : 2^{N^{bp}-1} \leq |x|\} \quad (4.5)$$

The probability distribution of |x| in the processed code-block might then be approximated by the following histogram:

$$\begin{aligned} p(|x|) &= n_k^{sc} \text{ for } x \in S_k,\ k = 0, ..., N^{bp} \\ \text{where} \quad n_k^{sc} &= \frac{N_k^{sc}}{N} \text{ for } k = 1, ..., N^{bp} \\ n_0^{sc} &= \frac{N - \sum_{k=0}^{N^{bp}} N_k^{sc}}{N} \end{aligned} \tag{4.6}$$

which verifies $n_k^{sc} \geq 0$ $(k = 1, ..., N^{bp})$ and $\sum_{k=0}^{N^{bp}} n_k^{sc} = 1$. Note that this is not a classic histogram as bins have a variable width. Indeed, each bin covers exactly the range of values that coefficients might take if they become significant in the corresponding bit-plane. This allows the histogram to be progressively refined: once the first most significant bit-planes have been decoded, relevant information on the histogram shape is already available. All the remaining non-significant coefficients are left in the last bin (S_0) until a new decoded bit-plane splits this last bin in two new and smaller non-overlapping bins, further refining the distribution approximation. When all bit-planes are decoded, the distribution is modeled by a set of N^{bp} parameters. In the following, these bin values will be considered as independent features, each one giving some relevant information on the image.

4.2.2 Feature properties

The header and wavelet-based features described above are *global* values: any feature we use corresponds to an average value on a given image area. In comparison with shape descriptors, this kind of feature is usually much easier to process. Moreover, in a JPEG 2000 framework, they are less influenced by aliasing effects that might occur due to the decimation operation during wavelet decompositions. On the other hand, their major drawback is obviously that they are not able to account for any geometrical structure present in the image. We will propose in Section 4.3 a method to solve this problem.

Another desired property of our extracted features is that they should not be dependent on the sample size. Indeed, we want to be able to compare (portions of) images that do not necessarily have the same amount of pixels. To do so, a *normalization* of the size-dependent features takes place. Concerning header-based features, if we consider an ensemble $\mathcal{B}$ of

code-blocks, we define

$$n_{\mathcal{B}}^{mq} = \frac{\sum_{b \in \mathcal{B}} N_b^{mq}}{\sum_{b \in \mathcal{B}} N_b^{bytes}}$$

where N_b^{bytes} is the number of bytes used by the decoded wavelet coefficients in code-block C[1]. Concerning wavelet-based features, the bin values defined in previous section are already independent of the sample size, as long as we use n_k^{sc} defined in Eq. 4.6, rather than N_k^{sc}. Thanks to this normalization, the subsequent classification process will use global and normalized features (n^{mq} and n_k^{sc} for $k = 1, .., N^{bp}$) whose values are all comprised between 0 and 1, whatever the size of the image area on which they are computed.

Depending on the subsequent classification task, one can take advantage of an additional property for the feature values, namely *rotation invariance*. As detailed in Chapter 2, subbands HL, LH and HH from a given resolution highlight high frequencies in the vertical, horizontal and diagonal directions, respectively. In some applications, these directions are not relevant to differentiate samples from different classes. For instance, Fig. 4.4 presents various texture classes that will be used in a classification process detailed in the next section. Samples from each of these classes have to lead to the same range of feature values, although different main directions are present in each class (this is particularly visible on the "corduroy" class, last row of Fig. 4.4). To enable this rotation invariance, feature values from the 3 subbands are merged together. In the rest of the chapter, this merging operation is simply done by averaging the n^{mq} and n_k^{sc} values over the different subbands of each resolution level.

4.2.3 Scalable extraction

The feature extraction process can take advantage of the scalability inherently present in JPEG 2000, namely the spatial, resolution and SNR[2] scalability levels. It is indeed possible to make a trade-off between the desired characterization accuracy and the feature extraction cost it will imply. "Extraction cost" has to be understood here as the amount of data from the JPEG 2000 code-stream that will have to be processed and/or decoded.

Let's consider a given spatial area in an image. Thanks to the spatial scalability of JPEG 2000, we can easily select the packets related to this

[1] Note that in most cases, this number N_b^{bytes} is equal to the number of coefficients in the code-block, as coefficients from a given component are usually coded with 1 byte.

[2] SNR scalability denotes the bit-plane progressive decoding ability.

area without having to process the remaining code-stream. Once relevant packets have been identified, a coarse characterization of the area content can be obtained by extracting the header-based features. Information from the lowest resolution headers is first collected, before being progressively refined with higher levels headers (using the resolution scalability). If a more accurate description of the image content is required, we can then complement the header information with wavelet-based features. This is done at the price of an higher computation load, as entropy-decoding operations are required. Again, we can start with the coarse data present in the low resolution levels and then get more information from higher levels as it is needed. Moreover, in this case, the SNR scalability is also useful. As described in previous section, for a given subband, the coefficient distribution can be more accurately approximated as more bit-planes are decoded.

To illustrate the benefit that can be drawn from such scalable extraction, a texture classification process is presented. Let's consider 4 different classes of textures. Each class contains 40 items, each one being a 640×480 grayscale image. Several random samples from each class are presented in Fig. 4.4. All images are JPEG 2000-compressed using 3 decomposition levels, which means 4 resolution levels in total, resolution 1 corresponding to the lowest resolution (the LL-subband). Feature values are all normalized[1] and information extracted from different subbands of the same resolution level is merged, as described in the previous section.

There are three successive steps in this example of classification process, each using a bit more information than the previous one.

1. In the first step of the classification process, only very cheap features are used, namely header-based features. Fig 4.5 shows how the 40 samples of each class are distributed in the 2D-space formed by two features chosen among all the header-based extracted information. These two features are the number of entropy-coded bytes that have been generated to encode the 2nd and the 4th resolution, respectively. As we see, this information, very easily extracted from the code-stream, is already sufficient to discriminate the "water" class from the three other ones. In contrast, classes "glass1", "glass2", and "corduroy" require more information to be clearly differentiated.

2. In addition to the headers already processed, we now assume that the 2 most significant bit-planes (7th and 8th) of resolution 1 and 2 have

[1] Note however that as samples have all the same size, the normalization is not strictly required.

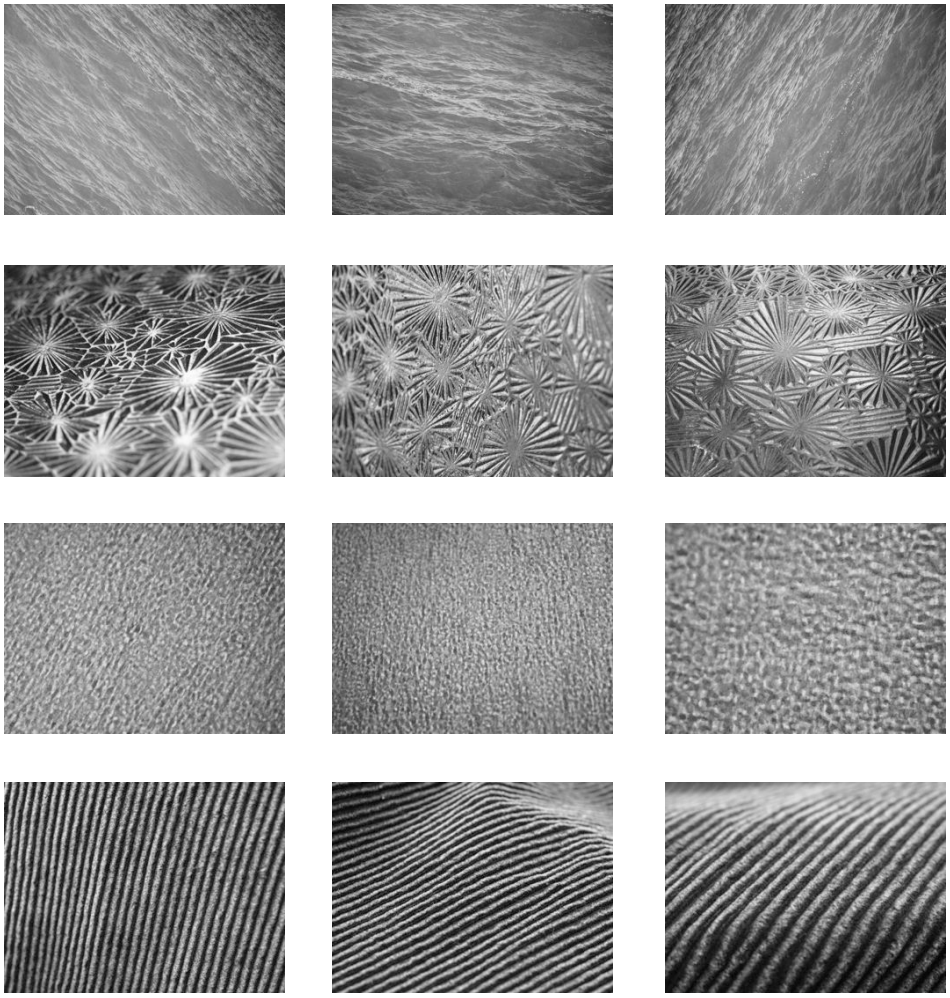


Figure 4.4: *Texture samples from 4 different classes, one in each row: water, glass1, glass2 and corduroy. They correspond respectively to class 7, 16, 17 and 24 from the Ponce texture database [52].*

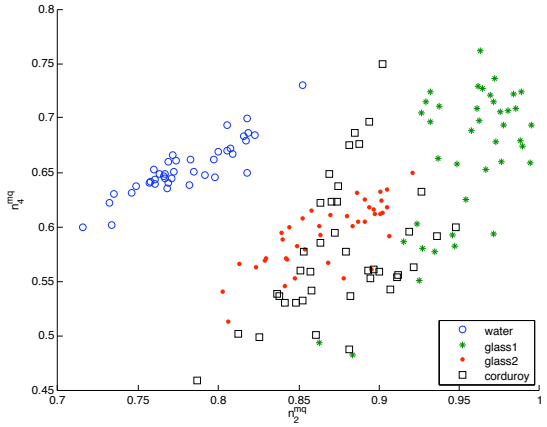


Figure 4.5: *First step of a scalable classification process, based only on header information.* n_i^{mq} *= normalized number of entropy-coded bytes for resolution* i*. These header-based features are sufficient to clearly discriminate the "water" class from the three other ones.*

been decoded. Valuable information about the number of significant coefficients from these bit-planes and resolutions are thus from now on available. As shown in Fig. 4.6, this new information allows to clearly separate the "glass2" samples from the two remaining classes, although it was not possible to do so with header-based features only. Even classes "glass1" and "corduroy" are almost separated but deeper feature extraction should enable to better distinguish them.

3. In this last step, the third most significant bit-plane (the 6th) from all resolutions is supposed to be decoded. Fig 4.7 shows how the two last classes can be separated using the number of new significant coefficients in the 6th bit-plane of resolution 1 and 4. While this separation is not as good as for the other classes, it is nevertheless slightly better than the one obtained for these two classes in Fig 4.6. Moreover, for illustration purpose we did only consider two features but the classification results are expected to be much better when considering more features. We invite the reader to refer to Section 4.5 where a complete classification process is envisioned.

As we see, this scalable extraction perfectly fits with a classification process made of several cascaded classifiers, like the one proposed by Viola

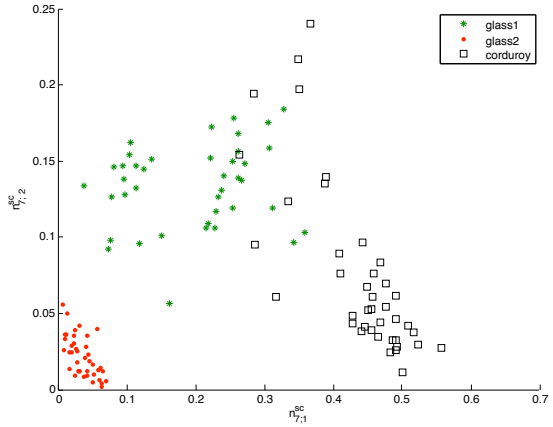


Figure 4.6: *Second step of a scalable classification process. The two most significant bit-planes from resolution 1 and 2 have been decoded.* $n^{sc}_{j;i}$ *= fraction of new significant coefficients in bit-plane* j*, for resolution* i*. It provides valuable information allowing to separate textures from class "glass2". Classes "glass1" and "corduroy" are also almost separated.*

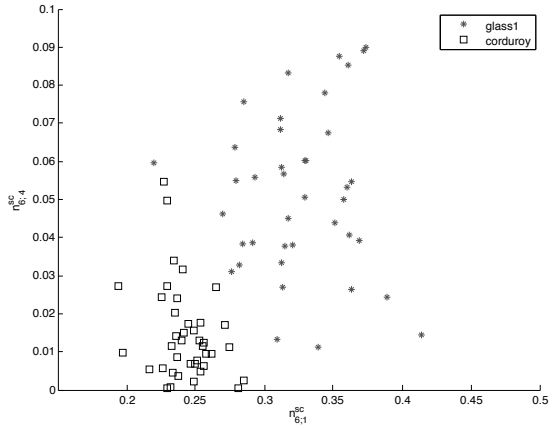


Figure 4.7: *Third and final step of a scalable classification process. The third most significant bit-plane from all resolutions has been decoded. This allows to more clearly discriminate the two remaining classes, "glass1" and "corduroy".*

and Jones in [119]. Such cascade will be detailed in Section 4.5 and used in an image retrieval context.

4.3 Integral volume representation

Let's summarize what we do have up to this point. Considering a code-block b of a JPEG 2000 compressed image, we showed that relevant information could be extracted from the code-stream about it:

- n_b^{mq}: normalized number of bytes used to entropically encode the code-block.
- $n_{k,b}^{sc}$: fraction of coefficients, among all coefficients of the code-block, that become significant in the bit-plane k ($k = 1, .., N_b^{bp}$).

Each of these extracted values gives a global information on the code-block and is comprised between 0 and 1. If the area of interest is made of several code-blocks, these values are very simply aggregated. This could happen if this area covers a bigger spatial zone than the one covered by the code-block, and/or if the rotation invariance is required. In this latter case, spatially corresponding code-blocks from the different subbands are aggregated. Eventually, we showed that these features were suited to discriminate textures and that the JPEG 2000 scalability levels could efficiently be used to set up a scalable classification process.

In many imaging applications involving content analysis, the user wants to be able to extract information about a specific image area, rather than about the whole image. Such area selection process might indeed be used to characterize the entire image by a set of feature values extracted from different spatial areas of the image. It can also be useful in an image retrieval task: each image of a database is scanned at various scales with sliding subwindows of different sizes. For each position of a subwindow, the corresponding image area is analyzed and classified as a positive or negative area.

Unfortunately, one of the major drawbacks of the features described in previous sections is that we have to stick to code-blocks borders when selecting an area of interest. This can reveal itself to be very constraining, in particular at low resolution levels. To realize this, let's remind that all code-blocks in an image have the same size: it is specified in the encoding options and is typically around 32x32 or 64x64 pixels. The effective image area covered by a given code-block depends on the resolution level to which this

code-block belongs. Indeed, let's consider an image compressed with N_L decomposition levels. There are $N_L + 1$ distinct resolution levels, denoted $r = 0, 1, ..., N_L$. The lowest resolution level, $r = 0$, is represented by the N_L-LL subband. The size of the original image area covered by a given code-block (b, p, c, r) is then

$$\begin{aligned} S_{b,p,c,r} &= (wb.2^{N_L}).(hb.2^{N_L}) = wb.hb.2^{2N_L} && \text{if } r = 0 \\ &= (wb.2^{N_L-r+1}).(hb.2^{N_L-r+1}) = wb.hb.2^{2(N_L-r+1)} && \text{if } r > 0 \end{aligned}$$

where wb and hb are the width and height of the code-blocks in the image, respectively. This is illustrated in Fig. 4.8 and 4.9. A 512x512 image of Lena has been compressed with 4 resolution levels ($N_L = 3$) and the size of code-blocks has been set to 32x32. Fig. 4.8 shows how subbands are each divided into a set of code-blocks of the same size. This subdivision in code-blocks allows to analyze independently several spatial areas in the image. The size and location of these areas depends on the resolution level and are depicted in Fig. 4.9 for each of them. Imagine now that for some

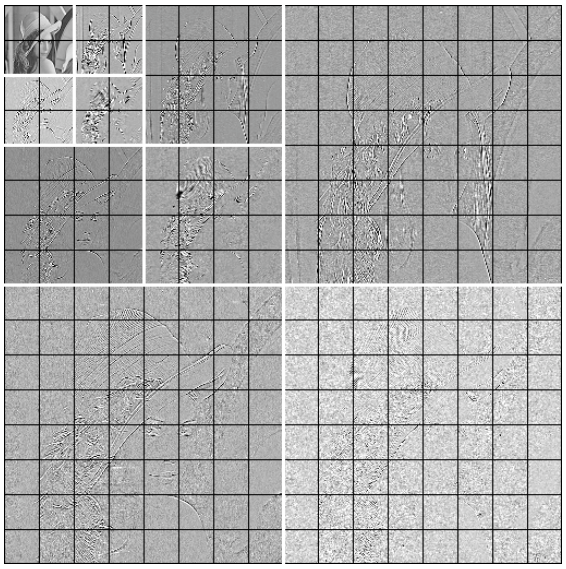


Figure 4.8: *Subdivision of resolution levels into code-blocks. All code-blocks have the same pixel size but cover different spatial areas in the image depending on the resolution level to which they belong.*

good reason we want to specifically analyze and characterize the texture

corresponding to Lena's shoulder smooth skin. To do so, we would like to extract relevant information about this area from each resolution level. In the highest resolution level ($r = 3$), there are fortunately two code-blocks[1] precisely covering the area of interest: features n^{mq} and n_k^{sc} from corresponding high resolution code-blocks will give valuable information about the texture. However, in the lower level ($r = 2$), the shoulder has already to "share" a code-block with some background information in the upper-right corner. Global features extracted from this code-block will average shoulder and background contributions and will therefore not purely characterize the skin texture. And the situation is even worse in the 2 lowest levels ($r = 0, 1$). This example shows that being constrained by code-block borders is problematic when information about a specific spatial area has to be extracted from each resolution level. The integral volume representation proposed in this section intends to solve this problem.

4.3.1 Principle

Let's consider first the wavelet-based features, namely the n_k^{sc} values. These features can be computed very rapidly, at any resolution level and on any spatial area, independently from code-blocks borders, using an intermediate representation of the image, called *integral volume*. This representation is a direct extension of the integral image introduced in [14] and more recently made famous in [119]. In this case, such representation has the noticeable advantage of being easily built *during* the decoding process. Once computed, features values are easily computed on any given spatial area.

While decoding the k-th bit-plane of a code-block b, a table T_k^{sc} is progressively built such that each element $T_k^{sc}(i, j)$ at column i and row j is the number of significant coefficients in the code-block at bit-plane k, among all coefficients above and to the left of element (i, j), inclusive:

$$T_k^{sc}(i,j) = \sum_{i' \leq i, j' \leq j} s_k(i', j') \text{ , for } i = 1, ..., wb \text{ and } j = 1, ..., hb$$

where wb and hb are respectively the width and height of code-blocks and $s_k(i, j)$ refers to the significance map of bit-plane k and verifies

$$\begin{aligned} s_k(i,j) &= 1 \quad \text{if } b(i,j) \geq 2^{k-1} \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

[1] There are actually 6 code-blocks in total as there are 3 subbands and 2 code-blocks per subband. However, for display convenience, we did represent the reconstructed level and not the subbands.

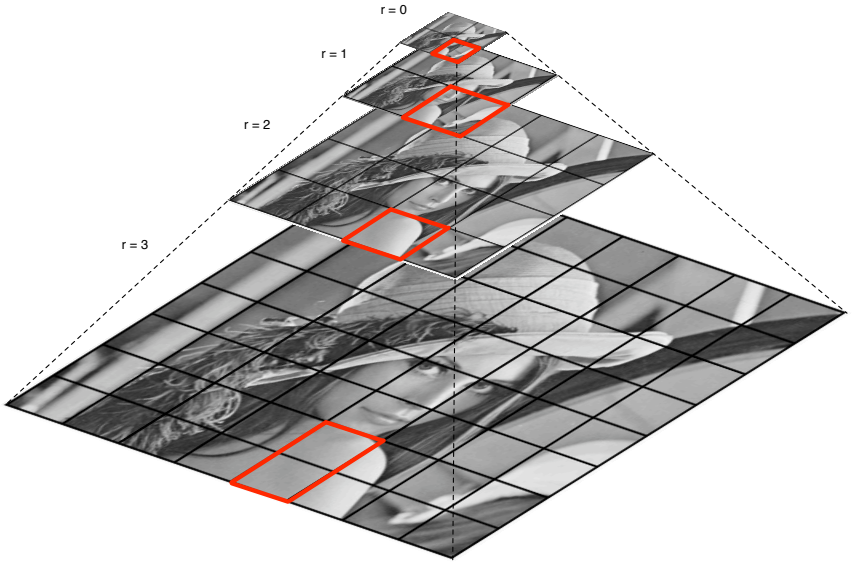


Figure 4.9: *Spatial areas covered by code-blocks in each resolution level. In the lower levels, the subdivision is very coarse, preventing to finely select a desired spatial area.*

This concept is illustrated in Fig. 4.10: the significance map s_k of a random bit-plane k is represented, together with the corresponding integral table T_k^{sc}.

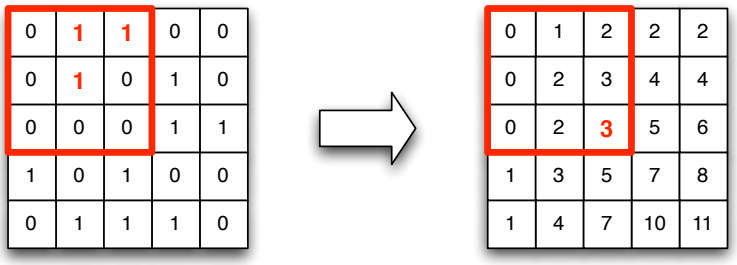


Figure 4.10: *significance map s_k of some 5x5 code-block (a) and corresponding integral table T_k^{sc} (b). Table $T_k^{sc}(i,j)$ is the sum of significant coefficients among all coefficients above and to the left of element (i,j).*

As mentioned above, computation of table T_k^{sc} perfectly fits with the entropy-decoding procedure of bit-plane k. Indeed, the significance status of each coefficient of a code-block is a state variable of the MQ-decoder. As such, it is constantly maintained and updated during the decoding procedure. Hence, after the MQ's clean-up pass has processed element (i,j) of the bit-plane, $s_k(i,j)$ is directly available and $T_k^{sc}(i,j)$ is obtained by the following recursive formula

$$T_k^{sc}(i,j) = T_k^{sc}(i-1,j) + T_k^{sc}(i,j-1) - T_k^{sc}(i-1,j-1) + s_k(i,j)$$

where initial conditions are

$$\begin{aligned} T_k^{sc}(i,-1) &= 0 \\ T_k^{sc}(-1,j) &= 0 \\ T_k^{sc}(-1,-1) &= 0. \end{aligned}$$

Moreover, changing these initial conditions enables to easily take into account previously decoded adjacent code-blocks, as shown in Fig. 4.11. If we want to account for code-blocks on the left or above the one being currently decoded, we just have to know the closest line from integral tables of these

adjacent code-blocks and use them as initial conditions:

$$\begin{aligned} T_k^{sc}(i,-1) &= T_k^{sc,up}(i,hb) \\ T_k^{sc}(-1,j) &= T_k^{sc,left}(wb,j) \\ T_k^{sc}(-1,-1) &= T_k^{sc,upleft}(wb,hb). \end{aligned}$$

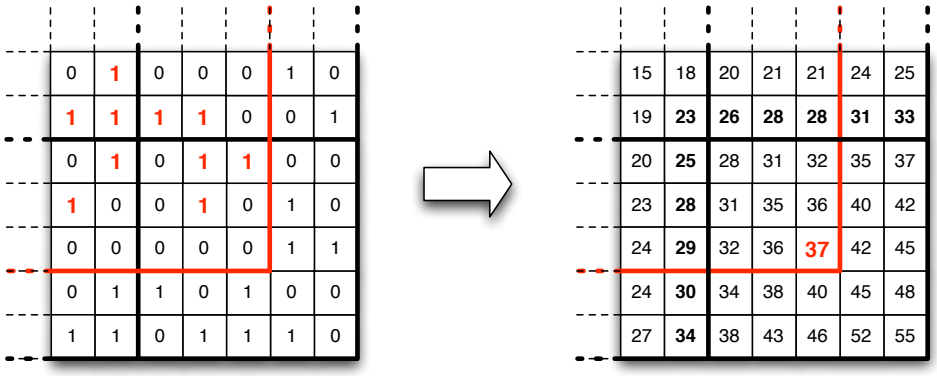


Figure 4.11: *Integral table T_k^{sc} with adjacent code-blocks taken into account. Initial conditions are set to the border lines values of the code-block currently decoded (black bold numbers in the figure).*

If we compute this integral table T_k^{sc} for all bit-planes of all code-blocks from a given subband sb ($sb \in \{LL, HL, LH, HH\}$) of a resolution level r, we obtain what we call an *integral volume* of this subband. Each element of such volume is simply given by

$$IV_{r,sb}(i,j,k) = T_k^{sc}(i,j) \quad for \quad \begin{aligned} i &= 1,...,w_{r,sb} \\ j &= 1,...,h_{r,sb} \\ k &= 1,...,N_{r,sb}^{bp} \end{aligned} \tag{4.7}$$

where $w_{r,sb}$ and $h_{r,sb}$ are respectively the width and height of subband sb in resolution r. $N_{r,sb}^{bp}$ is the maximum number of bit-planes in the subband. Once computed, this integral volume allows to very easily obtain the number N_k^{sc} of new significant coefficients in bit-plane k on any spatial area inside the subband. Indeed, let's consider a subband represented by

its integral volume IV, and a spatial area A in this subband, defined by its upper left corner (i_0, j_0) and its lower right corner (i_1, j_1). The amount of new significant coefficients in A, comprised in a bit-planes subset K ($K = [k_0; k_1]$, $k_0 \leq k_1$) is simply given by

$$\begin{aligned} N^{sc}_{A,K} &= \sum_{k=k_0}^{k_1} N^{sc}_{A,k} \\ &= IV(i_0, j_0, k_0) + IV(i_1, j_1, k_0) \\ &\quad -IV(i_0, j_1, k_0) - IV(i_1, j_0, k_0) \\ &\quad -IV(i_0, j_0, k_1) - IV(i_1, j_1, k_1) \\ &\quad +IV(i_0, j_1, k_1) + IV(i_1, j_0, k_1). \end{aligned} \tag{4.8}$$

Eight elements of integral volume IV and seven arithmetic operations are

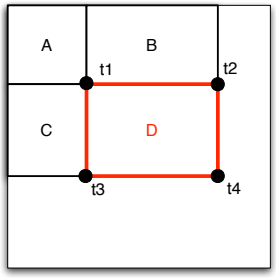
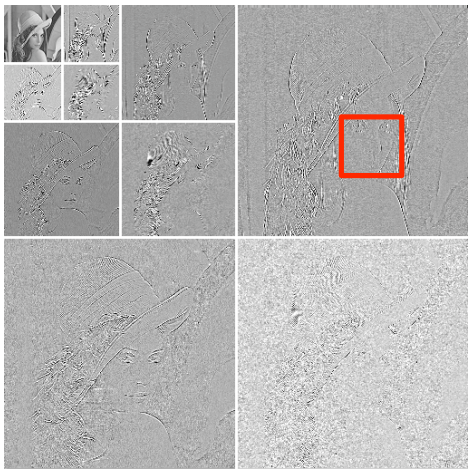


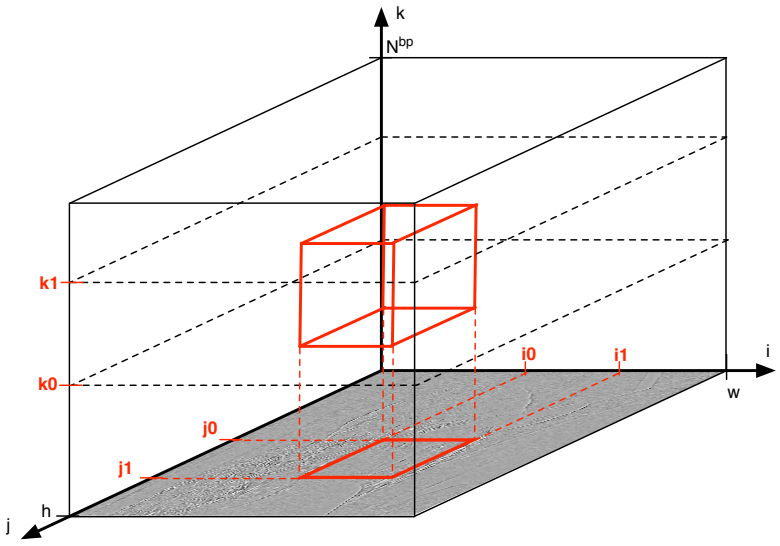
Figure 4.12: *Integral table illustration. The sum of all coefficients in area D is obtained with four elements of the corresponding integral table. The value $t1$ is the sum of all elements in area A, $t2$ corresponds to $A + B$, $t3$ to $A + C$ and $t4$ to $A + B + C + D$. The sum within D is then equal to $t1 + t4 - t2 - t3$. (Figure retrieved from [119]).*

thus sufficient to compute $N^{sc}_{A,K}$ on any area A. This is a 3D-extension of the concept explained for the 2D in [119] and reproduced in Fig. 4.12 for convenience. The integral volume of the subband is illustrated in Fig. 4.13.

In order to have a single representation from which it would be possible to extract both wavelet and header-based features, we use the same concept of integral table to store the N^{mq} values. For this kind of feature, one single value is available for each code-block: the total number of entropy-coded bytes for this code-block. Consequently, an approximation is made so as to infer a value for every coefficient of the code-blocks and fill all the elements



(a) The red square area is the zone in a given subband from which we want to extract the number of new significant coefficients between two bit-planes k_0 and k_1.



(b) Integral volume of the corresponding subband is represented. The k-axis is the bit-plane dimension. The number of new significant coefficients in the red volume is given by Eq. 4.8.

Figure 4.13: *Integral volume illustration.*

of the integral table. We simply assume that a coefficient x at location (i, j) in a code-block b is entropy-coded using

$$N_{x,b}^{mq}(i,j) = \frac{N_b^{mq}}{N_b^{coeff}}$$

bytes, where N_b^{coeff} is the number of coefficients in code-block b. It is then possible to define the integral table T^{mq} like it was done for T_k^{sc}:

$$T^{mq}(i,j) = \sum_{i' \leq i, j' \leq j} N_{x,b}^{mq}(i',j') \text{ , for } i = 1, ..., wb \text{ and } j = 1, ..., hb$$

By adding table T^{mq} on top of all tables T_k^{sc} of the integral volume $IV_{r,sb}$ defined in Eq. 4.7

$$IV_{r,sb}(i,j,N^{bp}+1) = T^{mq}(i,j) \quad for \quad \begin{matrix} i = 1, ..., w_{r,sb} \\ j = 1, ..., h_{r,sb} \end{matrix}$$

we obtain a single representation of the subband sb of resolution r from which any feature N^{mq} or N^{sc} can be easily computed on any spatial area. As information included in IV for $k = 1, ..., N^{bp}$ is not of the same kind as for $k = N^{bp} + 1$, attention will of course be given to not select subvolumes in IV that would mix these two zones. They are actually put together only to provide an unique representation of the subband.

A normalization is then applied to this integral volume in order to be able to compare areas of different sizes. To do so, we simply divide all elements of the integral volume $IV_{r,sb}$ by its total number of elements $N^{IV_{r,sb}}$:

$$iv_{r,sb}(i,j,k) = \frac{IV_{r,sb}(i,j,k)}{N^{IV_{r,sb}}}$$

Then, for any spatial area A defined by its upper left and lower right corner (i_0, j_0) and (i_1, j_1) and for any bit-planes interval $K = [k_0; k_1]$, the normalized values n_A^{mq} and $n_{A,K}^{sc}$ are given by ($iv_{r,sb}$ is simply noted iv for convenience)

$$\begin{aligned} n_{A,K}^{sc} &= (iv(i_0,j_0,k_0) + iv(i_1,j_1,k_0) \\ &\quad -iv(i_0,j_1,k_0) - iv(i_1,j_0,k_0) \\ &\quad -iv(i_0,j_0,k_1) - iv(i_1,j_1,k_1) \\ &\quad +iv(i_0,j_1,k_1) + iv(i_1,j_0,k_1))/a \end{aligned} \tag{4.9}$$

$$\begin{aligned} n_A^{mq} &= \Big(iv(i_0,j_0,N^{bp}+1) + iv(i_1,j_1,N^{bp}+1) \\ &\quad -iv(i_0,j_1,N^{bp}+1) - iv(i_1,j_0,N^{bp}+1)\Big)/a. \end{aligned} \tag{4.10}$$

where

$$1 \leq i_0 < i_1 \leq w_{r,sb}$$
$$1 \leq j_0 < j_1 \leq h_{r,sb}$$
$$1 \leq k_0 < k_1 \leq N^{bp}_{r,sb}$$

and

$$a = \frac{A}{A_{total}} = \frac{(i_1 - i_0).(j_1 - j_0)}{w_{r,sb}.h_{r,sb}} = \frac{(i_1 - i_0).(j_1 - j_0)}{N^{IV_{r,sb}}}.$$

The normalized area a actually corresponds to the fraction of the total area of the integral volume covered by the area of interest. Moreover, to get really comparable volumes, the N^{bp} value is set to a fixed value (8 in our work), whatever the maximum number of bit-planes in the considered subband. If they are more bit-planes, they are all accumulated in the most significant one. If they are less, the most significant bit-planes are simply set to zero. Eventually, to provide rotation invariance property, integral volumes from the 3 different subbands of a same resolution level are merged by a simple averaging operation:

$$iv_r(i,j,k) = \frac{\sum_{sb} iv_{r,sb}(i,j,k)}{3}$$

In conclusion, starting from any JPEG 2000 image, compressed with N_L decomposition levels, an intermediate representation can be generated. The building of this representation suits well with the image decoding process and can be done progressively, as more data is decompressed. It consists in several different volumes of size w_rxh_rx$(N^{bp}+1)$ $(r = 0, .., N_L)$. With these volumes available, two kind of relevant information can be obtained about the content of any spatial area inside the image, independently from the code-block borders. These two kind of information are, for a given resolution level, (i) the normalized number of entropy-coded bytes used to encode this area and (ii) the number of wavelet coefficients in this area that are comprised between two given values of power of 2. For a given image, there are 9 parameters needed to specify the exact feature value we want to extract from the volumes:

- r: the resolution level,
- c: the component ($c > 1$ in case of color or multi-modal images),

- sb: the subband number ($sb > 1$ if $r > 0$ and subband averaging has not been done),
- x_0, y_0: the upper left corner of the area of interest,
- x_1, y_1: the lower right corner of the area of interest,
- k_0, k_1: the interval of bit-planes that has to be taken into account.

The number of volumes representing an image is thus given by (r×c×sb). Note that (x_0, y_0, x_1, y_1) are normalized coordinates. They are comprised between 0 and 1 and are related to coordinates in each volume in the following way

$$\begin{aligned} i_{r,c} &= x_c.w_r \\ j_{r,c} &= y_c.h_r \end{aligned}$$

for $c = 0, 1$. Note also that if the number of entropy-coded bytes is desired on the area, k_0 and k_1 are both taken equal to $(N^{bp} + 1)$. Using the r, c, and sb parameters to select the right volume and then using the other parameters to resolve Eq. 4.9 or 4.10, the desired feature value is obtained.

4.3.2 Spatially localized image characterization

Global features computed on a whole image are well suited to discriminate homogenous images, i.e. images representing a single kind of texture, like the ones presented in Fig. 4.4. More generally, we can say that global features are more relevant if they are extracted from homogeneous zones. However, in most classification or retrieval applications, relevant image content is actually characterized by an ensemble of such homogeneous zones. This is illustrated with a very simple example in Fig. 4.14. The depicted landscape is obviously made of 3 main zones, each characterized by its own set of global feature values.

In this work, we make the assumption that any image of interest can be approximated by a "patchwork" of different textures. We assume also that the spatial localization of the different zones in the image is relevant and should be taken into account in the image description. This idea is inspired from the work of Amit and Geman [2] where the authors used the relative spatial position of tags to characterize written letters. Our image description is thus made of an ensemble of *global* feature values computed on several *local* areas, each area being located at a precise place in the image. This is illustrated in Fig. 4.15(a): the image is characterized with

Figure 4.14: *An image is in most cases made of several different textures. In this case, there are obviously 3 main zones, each characterized by its own set of global feature values.*

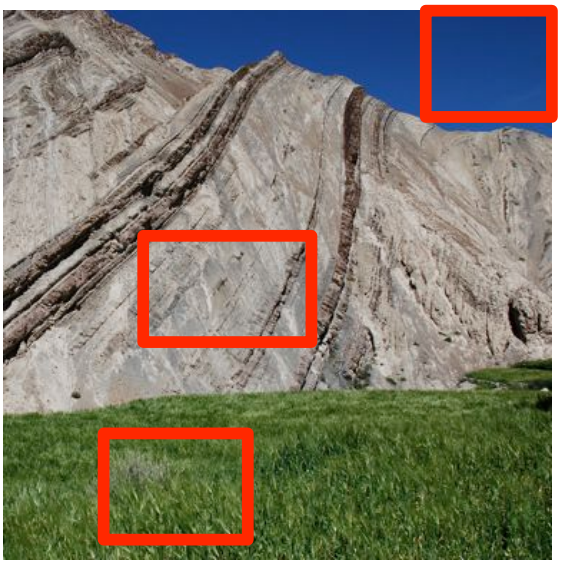
three different areas, each one from a different zone in the image. The image description will include the exact position of each area, together with the kind of global feature value extracted from each of them (i.e., the normalized number of bytes used to encode the area, or the number of significant coefficients in a given bit-plane, ...). Thanks to the integral volumes described above, it becomes very easy to generate such description for a given image. Indeed, we just have to provide a set of nx9 parameters, n being the number of local areas where a feature value is extracted. It should be noted that color information can also be used by extracting information from color components.

As suggested in Fig. 4.15(b), this description can subsequently be used to retrieve similar images. To do this, all candidate images are evaluated in the specific areas mentioned in the image description and the same features as in the image-query are extracted. Closest candidate images in terms of feature values are then considered as similar. The example shown in Fig. 4.15(b) is of course a very simple one and is more a proof-of-concept than a "real-world" application. In more realistic scenarios, a set of several classes of images is used as a learning set. Based on these images, the learning algorithm will try to find which areas and which features in those areas best separate the image classes. Once the classifier has learned enough to correctly discriminate the learning set, it can be used to classify unlabeled images. This kind of classification scheme is presented in Section 4.4.

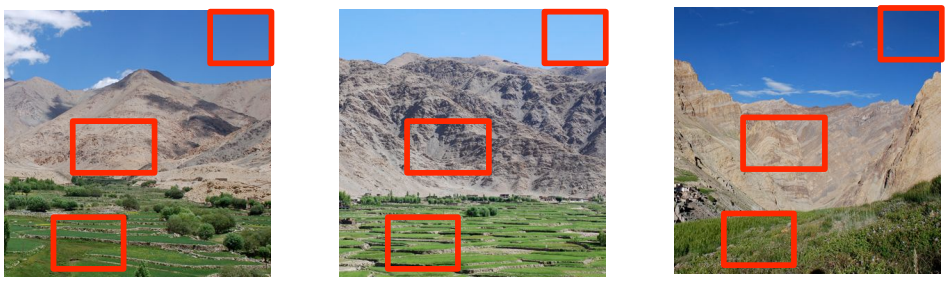
4.3.3 Samples and features

Integral volumes enable to evaluate different features on any spatial area inside an image. This has been used to generate more accurate description of such image by taking into account the spatial localization of the different areas considered. The same mechanism can also be used to generate independent samples from a given image. This might be useful in at least two (non-exclusive) cases.

1. In a classification task, images from a same class often differ in many aspects. From one image to another, there might be partial occlusions, cluttered background, illumination or orientation changes, etc. These differences can drastically reduce the classification performances if each image is only considered globally. If we randomly extract from each image a certain amount of square samples of different size and if we consider them as independent items belonging to the initial image class, classification results will be less sensitive to the



(a) Different areas are selected to characterize an image. The description of the image includes the exact position of each area, together with the kind of global feature value extracted from each of them



(b) The image content description can subsequently be used to retrieve similar images.

Figure 4.15: *Image content characterization.*

above-mentioned problems. This idea has been introduced in [70] and further explained in [71]. As detailed in these papers, subsampling an image has several advantages:

- The correct classification of all samples is not required to correctly classify one image. This can make the classification process less sensitive to partial occlusions.
- Spatial image transformation do not alter all samples to the same extent. Many of them will be left more or less intact, resulting in increased robustness to viewpoint and orientation changes.
- When only few images are available in the learning set, subsampling enables to increase the number of items in each class and feed the learning algorithm with more examples.

Subsampling an image might appear to pursue goals that are opposed to the feature extraction process described above which specifically took into account the spatial position of various areas. However, the two processes are actually complementary. Samples are generated to increase the classification robustness. Then, each sample is described with a set of features computed on different areas in the sample.

2. In a retrieval task, big images have to be progressively scanned by a sliding window of interest. For each position and scale of this window, the related image content has to be evaluated and classified as a positive or negative item. Being able to easily analyze independent samples from a given image is obviously useful in such task.

Practically, once the integral volumes of an image are available, a sample can simply be described by four parameters, namely the two pairs of normalized coordinates corresponding to the upper left and lower right corners of the sample. These normalized coordinates allow to access the targeted area at any resolution and in any component of the image. Features can then be extracted from the sample by combining the sample coordinates with the feature coordinates. Let's consider a grayscale image of size $w \times h$, compressed with N_L decomposition levels. The integral volume representation of this image consist in $(N_L + 1)$ volumes (we assume that subband merging has been done) of size $w_r \times h_r$ given by

$$w_r = \frac{w}{2^{N_L - r + 1}}$$
$$h_r = \frac{h}{2^{N_L - r + 1}}$$

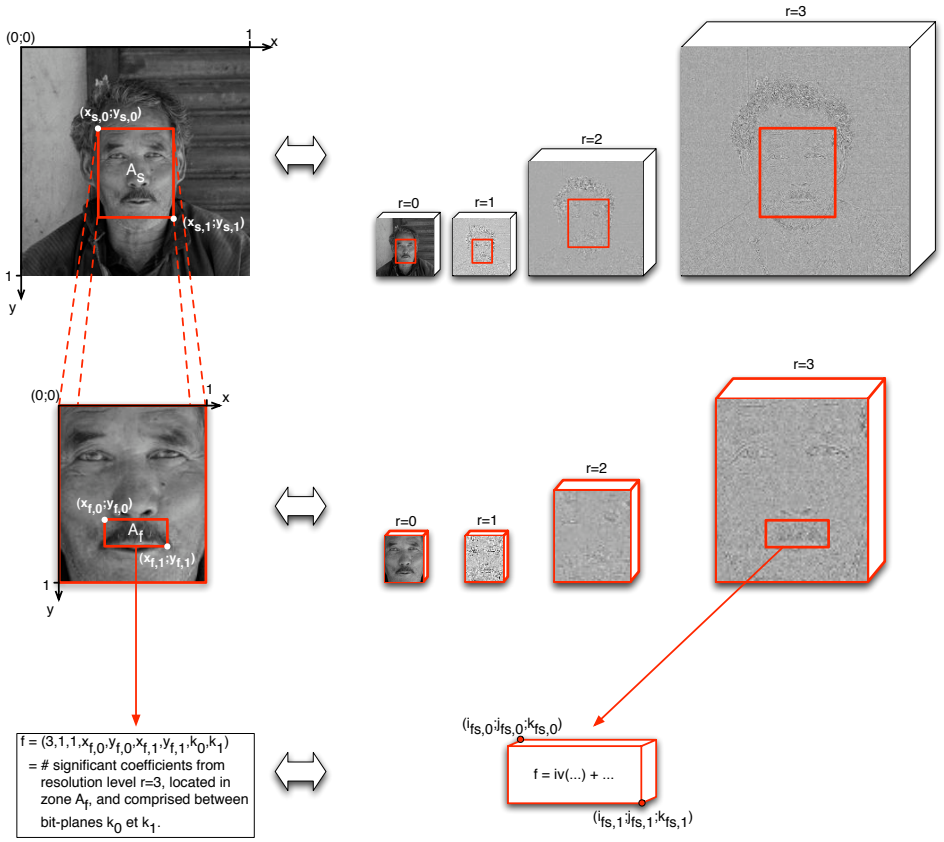


Figure 4.16: *Combined sampling and feature extraction in integral volumes. A sample is first defined with the normalized coordinates of two of its corners. This allows to select the corresponding portions of integral volumes in each resolution level. Then, the desired feature* f *can be extracted from the sample.*

for $r = 1, ..., N_L$, and

$$w_0 = \frac{w}{2^{N_L}}$$
$$h_0 = \frac{h}{2^{N_L}}.$$

If we want to extract a feature f defined by the 9 parameters $(r_f, c_f, sb_f, x_{f,0}, y_{f,0}, x_{f,1}, y_{f,1}, k_{f,0}, k_{f,1})$ from a sample of this image defined by the 4 parameters $(x_{s,0}, y_{s,0}, x_{s,1}, y_{s,1})$, the integral volume iv_{r_f} corresponding to resolution level r_f is first selected (c_f and sb_f parameters are useless in this case). Then, a subvolume is extracted. It can be defined by two of its corners $(i_{fs,0}, j_{fs,0}, k_{fs,0})$ and $(i_{fs,1}, j_{fs,1}, k_{fs,1})$ such that

$$i_{fs,c} = ((x_{s,1} - x_{s,0}) .x_{f,c} + x_{s,0}) .w_{r_f} \quad (4.11)$$
$$j_{fs,c} = ((y_{s,1} - y_{s,0}) .y_{f,c} + y_{s,0}) .h_{r_f} \quad (4.12)$$
$$k_{fs,c} = k_{f,c} \quad (4.13)$$

for $c = 0, 1$. Eventually the desired feature value is computed with Eq. 4.9. This process is illustrated in Fig. 4.16.

Thanks to this sampling procedure, information from all resolution levels and all components about any given spatial area in an image can be very easily isolated and analyzed.

4.4 A JPEG 2000 image classifier

In this section, we introduce a JPEG 2000 image classifier. It uses the integral volume representation described above in a robust and efficient classification algorithm. This classifier allows us to assess the discriminant power of the features extracted from a JPEG 2000 code-stream, together with the efficiency of the integral volume representation used to store them. Moreover, it paves the way towards an image retrieval system that will combine several such classifiers in a scalable cascade architecture (see Section 4.5).

Like many image classification methods, the proposed classifier assumes the existence of two sets of images, each image belonging to one of N_c distinct classes ($N_c > 2$)[1]. The first set, whose labels are known by the classifier, is called the learning set and is used to train the classifier so

[1]The case ($N_c = 2$) is assimilated to an image retrieval task and will be envisioned in Section 4.5.

that it best separates images from different classes. The second set, called the test set, is used to evaluate the performances achieved by the trained classifier.

In the remainder of the Section, the classification method is first detailed (4.4.1). Experiments on freely available image databases are then carried on (4.4.2) to assess the performances obtained with such classifier. Eventually, a discussion on the results is proposed (4.4.3).

4.4.1 Method

The classification algorithm used in combination with integral volumes is an ensemble of binary decision trees.

Tree-based methods consist in a succession of answers to (binary) questions that progressively divide (and classify) an initial set of items. Starting from a root node, the tree is grown by finding splits of data that make the representation more "pure" [31]. The entropy of the data set is often used to evaluate this reduction of "impurity". Among the advantages of these methods, we should cite the ability to produce complex and accurate classifiers through a succession of simple recursive operations. This gives such methods a great flexibility. Moreover, as they can process all kind of data (numerical or non-numerical), they are used in a wide range of application. Historically, they were developed in the 80's (see [9] for example), in particular with the work of Quinlan who introduced 2 famous tree-based classifiers: ID3 [90] and C4.5 [91].

However, these methods usually suffer from high variance: small changes in the training set often produce large changes in resulting decision tree [27, 38]. This instability is a major drawback that has limited the interest for decision trees until the development of ensemble methods [28]. An ensemble method is a "meta-algorithm" that consists in combining in some way (typically by weighted or unweighted voting) a set of individual classifiers. Under certain conditions, such meta-classifier performs much better than any individual classifier composing it. As explained in [28], "*a sufficient and necessary condition for an ensemble of classifiers to perform better than any of its individual members is if these members are accurate and diverse. An accurate classifier is one that has an error rate better than random guessing on new x values. Two classifiers are diverse if they make different errors on new data points*". In the context of decision trees, ensemble methods can drastically reduce the variance problem and even benefit from this instability by averaging the decisions from a set of trees.

Among ensemble methods, three of them have raised particular inter-

est [29], namely boosting, bagging and randomization. In each case, the same base learning algorithm is invoked many times with different parameters or with a different training set. In boosting [37] (and its famous algorithm AdaBoost), a weight is maintained for every item in the training set. This set of weights is adapted from one individual classifier to the next one by increasing the importance of misclassified items. In this way, a succession of single classifiers is built, that focus progressively on hard examples. Bagging [8] builds a set of totally independent classifiers, each one being trained on a bootstrap replicate of the original training set. With randomized methods [27], the training set is usually the same from one classifier to the other but internal decisions of the base learning algorithm are randomized. For example, each split is randomly chosen among 20 good pre-computed candidate splits. In [29], a comparison of the three methods shows that boosting usually outperforms bagging or randomization. However, these better performances come at the cost of a much bigger complexity in terms of learning time [72].

More recently, Geurts *et al.* further increased the randomness of the learning algorithm by introducing their Extra-Trees (for "Extremely Randomized Trees") [41]. In their work, every split in a tree is chosen completely at random. This basically means that both the tested feature and the cut point selected for thresholding, are chosen randomly. Surprisingly, this pretty simple method proves to be quite efficient, and obviously benefits also from a very low complexity.

In our work, we propose to combine an ensemble of random decision trees with our JPEG 2000 integral volume representation. The proposed scheme is based on the work by Marée *et al.* [71, 72, 73]. Authors of these papers combined the Extra-Trees mentioned above with random subwindows extracted from the sets of images. In the following, we summarize their method and detail the few changes we had to make in order to fit in our JPEG 2000 framework.

Sampling

As explained in Section 4.3, subsampling images have several advantages when dealing with classification tasks, the two main ones being that

- it increases the robustness of the classifier: all samples from an image do not need to be correctly classified to still correctly classify the image,
- it decreases the influence of the *statistical problem* described by Diet-

terich in [28]: when the amount of training data is too small compared to the size of the hypothesis space, the hypothesis selected by the classifier might not be accurate at all when applied to the test set. In particular, when building decision trees, a great amount of samples need to be available: it allows to build trees deep enough to test a sufficient number of relevant features.

In [71], authors extract a large number of possibly overlapping, square subwindows of random sizes and at random positions from training images. These subwindows are then rescaled to a size of 16x16. A sample is then defined as the vector of 256 values (768 in case of color images) corresponding to the pixel values of each rescaled subwindow. Note that for color images, resized subwindows are transformed to a HSV color space which is more robust to illumination changes compared to RGB. These samples are generated offline and once for all, before training the decision trees.

In our work, the approach is similar but differs in two main points:

- There is no "fixed-size feature vector" characterizing each sample. Actually, there is virtually an unlimited number of features that can be extracted from a given sample. A feature is uniquely defined by the 9 parameters described in Section 4.3, defining the resolution, component, subband, bit-planes and spatial zone where the feature should be evaluated. Thanks to the integral volume representation, this evaluation can be done at a very low cost.

- Integral volumes are computed before the training process but samples are extracted from these volumes *during* the training phase. Samples are indeed defined by two pairs of coordinates and feature evaluation on a sample is simply made by combining these coordinates with the feature parameters (see Eq. 4.11 to 4.13). One advantage of this approach is that this very cheap sample generation procedure allows us to renew or add samples whenever it is useful. In particular, samples are easily renewed from one tree to the next one, which is similar to a bagging process [8], without any additional cost.

Learning

During the learning phase, samples extracted from the training images are associated with the class of their parent image and are then provided to a machine learning algorithm that will build a sample classification model. Any algorithm could be used here (Support Vector Machines, Neural Networks, ...) but as already mentioned, we choose to follow the approach of

Marée *et al.* [71] and decide to use an ensemble of extremely randomized trees [41] (Extra-Trees). This choice is motivated by the excellent classification results obtained with such trees in consideration of the low complexity of their building process.

An Extra-Tree differs from other decision tree by the way each node of the tree is split. A split is defined by two variables:

- the feature f which will be evaluated in each sample in order to know if this sample will go in the left or right child node,
- the threshold th defining the cut-point in the range of all possible values for feature f. For a given sample s, if $f(s) \leq th$, then s goes to the left child node, otherwise it goes to the right one.

In an Extra-Tree, these two variables are chosen completely at random, i.e. f is randomly chosen among all available features and th is randomly picked between the minimum and maximum values reached by feature f among all samples in the node to be split. Then, the best split is chosen among K random splits, according to a score measure based on the entropy reduction brought by each split. Note that if $K = 1$, the tree is called *totally randomized tree* as it is not based anymore on any score measure. Once there are less than n_{min} samples in a node (or if all samples belong to the same class), the node is not further split and is then considered as a leaf. For a complete study of Extra-Trees (including a bias/variance analysis), we invite the reader to refer to [41].

An ensemble of Extra-Trees is thus defined by three parameters:

- T: the number of trees
- K: the number of random splits among which the best one is chosen when splitting a node.
- n_{min}: the minimum number of samples required in a node to initiate a split.

In our work, we add a fourth parameter:

- Sc_{min}: the score minimum to be reached by a split. In our implementation, splits are randomly picked until one of the two following conditions is true:
  1. The score Sc_{min} has been reached by a split.
  2. The maximum number of splits K has been tried.

Testing

During the test step, N_{test} samples are extracted from each image and are propagated in the T trees. Each sample reaches a specific leaf in each tree. A leaf l from a tree t is characterized by a probability distribution $p_t(i|l)$ $(i = 1, ..., n_c)$, the probability that a sample s reaching leaf l belongs to the ith class among n_c classes. This distribution is simply obtained by counting, for each class, the training samples reaching this leaf. Each sample s from a test image k reaching a leaf l from tree t is then associated with the corresponding leaf distribution:

$$p_t(i|s_k) = p_t(i|l) \text{ for } i = 1, ..., n_c$$

Once all N_{test} samples from the test image k have been propagated in the T trees, we obtain a set $\Gamma_k = \{s_k\}$ of observations on image k. The probability $p(i|\Gamma_k)$ that image k belongs to class i is given by

$$p(i|\Gamma_k) = \frac{1}{N_{test}} \sum_{s_k \in \Gamma_k} \frac{\sum_{t=1}^{T} p_t(i|s_k)}{T}$$

and the class c_k^* predicted for test image k is simply

$$c_k^* = \arg \max_{c_k \in [1; n_c]} p(c_k|\Gamma_k)$$

If $n_{min} = 2$, the trees have been fully grown and each leaf is populated with training samples from one and only one class. In this case, the aggregation procedure is equal to a majority vote rule.

4.4.2 Experiments set-up

We propose in this section to assess the performances that can be reached using the classification scheme described above.

Databases and protocols

Four databases have been used in these experiments. Except for the PONCE dataset, the databases and experimental protocols used are the same as in [71]. Datasets and protocols description is reproduced here for completeness.

1. **PONCE**

   - **Description**: the Ponce Texture Database[1], introduced in [52], is a dataset of 640×480 grayscale images of 25 different textures with 40 images per class (1000 images in total). An item from each class is presented in Fig. 4.17. Significant viewpoint changes and scale differences are present within each class, and illumination conditions are uncontrolled. This database has been used in our experiments to highlight the efficiency of wavelet-based features when dealing with textures. It will again be used in Section 4.5 to evaluate the performances of a cascade of JPEG 2000 classifiers.
   - **Protocol**: the learning set is made of 39 images from each of the 25 classes. The remaining 25 images compose the test set. The final error rate is the average of 40 error rates, the ith rate being obtained by using image i from each class for the test set.

2. **ZUBUD**

   - **Description**: the ZuBud dataset[2] [103] contains color images of 201 different buildings in Zürich (see a few examples in Fig. 4.18). Each building in the training set is represented with five 640×480 images (1005 images in total). The test set is made of 115 images of size 320×240, covering a subset of the 201 different classes. The two sets were taken by two different cameras, in different seasons and under different weather conditions, leading to various illumination conditions. Partial occlusions and cluttered backgrounds, as well as scale and orientation changes, are often present in the images. Moreover, the test set is available as JPEG images while the learning set uses the PNG format. As high frequencies are not processed in the same way by JPEG and PNG, initial classification tests with this database were awful (about 80% error rate). Features extracted from high-frequencies resolution levels in the training set were indeed not representative at all of the corresponding features in the test set. To circumvent this problem and enable a better classification, images from the training set have been compressed beforehand using the JPEG algorithm with the same targeted quality level than the one used in the test

[1] http://www-cvr.ai.uiuc.edu/ponce_grp/data/
[2] http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html

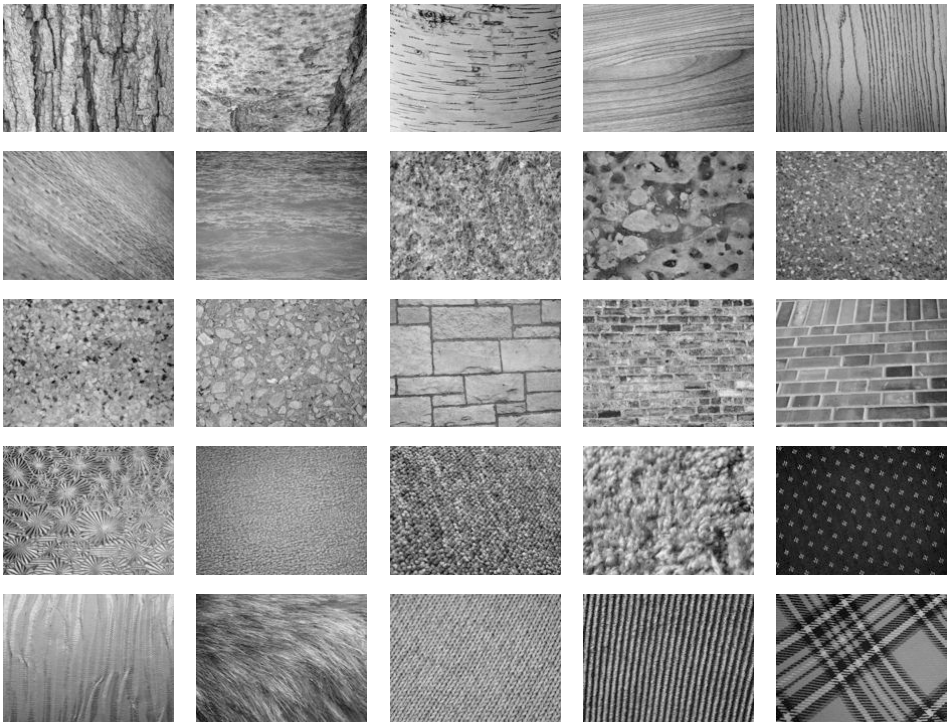


Figure 4.17: *One texture sample from each of the 25 classes of the PONCE database.*

set. This example showed us that as features used in our classification scheme depends on the way high frequencies are processed, the initial format of the images should carefully be taken into account when implementing a classification or retrieval application.

- **Protocol**: as learning and test images are provided as separate sets, there is no testing protocol to define.

Figure 4.18: *3 views from 3 different buildings in Zürich. The ZuBud database contains 5 such views from 201 different buildings.*

3. **ETH-80**

- **Description**: the cogvis ETH-80 dataset[1] is targeted specifically to the task of object categorization. It is made of 3280 color images (256x256) divided in 8 distinct categories (apples, pears, tomatoes, cows, dogs, horses, cups, cars). For each category, 10 different objects are provided. Each object is repre-

[1] http://www.mis.informatik.tu-darmstadt.de/Research/Projects/categorization/eth80-db.html

sented with 41 different images from viewpoints equally spaced over the upper viewing hemisphere (Fig. 4.19).

- **Protocol**: the protocol used is the same as the one used by Leibe and Schiele in [53], i.e. a leave-one-object-out cross-validation. The learning set is made of all views of 79 objects (which means a set of 3239 images) and the test is operated on all 41 views of the remaining object. Results are averaged over all 80 possible test objects.

4. **COIL-100**

   - **Description**: the COIL-100 dataset [79] from the Columbia Object Image Library[1] is made of 7200 color 128x128 images. There are 100 classes, each one representing a different 3D object (Fig. 4.20). Each object is represented by 72 images at pose intervals of 5 degrees.
   - **Protocol**: two different protocols were used, like in [74]. *Protocol 1* takes the pose at 0° of each object as the learning set (100 images in total) and the remaining images as the test set (7100 images). *Protocol 2* takes 18 images of each object (two images of the same object being separated by 20°) as the learning set (1800 images) and the remaining ones as the test set (5400 images).

JPEG 2000 parameters

All images from the databases are first encoded with the JPEG 2000 algorithm using the OpenJPEG library[2]. This algorithm has a lot of different options that can be specified when compressing an image. These options influence the compression efficiency, the robustness to errors, the encoder speed, etc. For our experiments, we decided to avoid using too "fancy" options values as we wanted our system to be able too work with a basic JPEG 2000 codec. Most important options used for the compression are:

- *Number of resolution levels*: 4. This means 3 successive decompositions. For the COIL-100 database however, we specified only 2 de-

[1] http://www1.cs.columbia.edu/CAVE/databases/

[2] OpenJPEG is an open-source JPEG 2000 library developed and maintained in the Communications Laboratory at UCL: http://www.openjpeg.org.

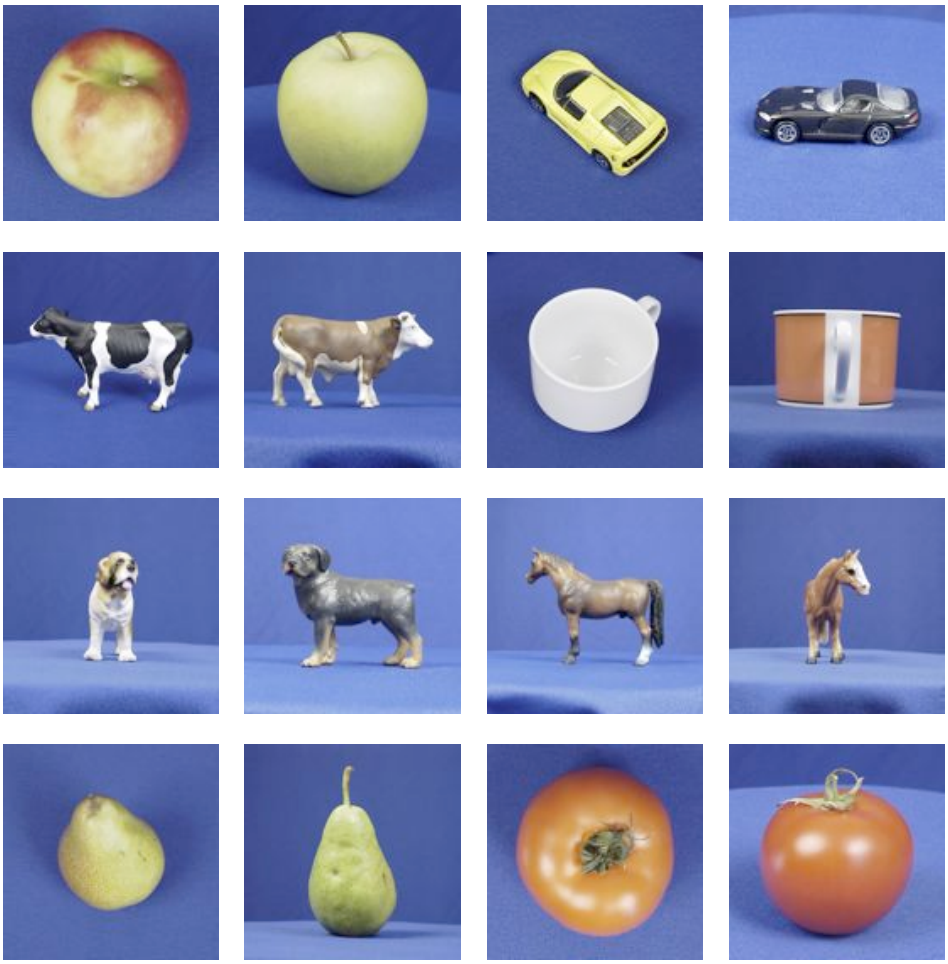


Figure 4.19: *Two objects from each of the 8 categories in the ETH-80 database: apple, car, cow, cup, dog, horse, pear and tomatoe.*

Figure 4.20: *One sample from each of the 100 classes in the COIL-100 database. The remaining samples of a given class are other views of the same object.*

compositions (3 levels) because the images were already quite small (128x128).

- *DWT filter*: 5-3. This is the lossless filter bank provided in JPEG 2000 part 1. Actually, all experiments described below have been conducted with both 5-3 and 9-7 filter banks. However, no significant difference could be highlighted concerning the achieved classification performances. We therefore conclude that this setting has no significant impact and should be set according to other aspects, like the need for a lossless compression (5-3) or a better compression efficiency (9-7).

- *Code-blocks size*: 32x32. This is a common size for JPEG 2000 code-blocks. In particular, this is the standard size required when compressing images for the Digital Cinema industry.

- *No quality layers*. No target bit-rates (either intermediate or final) were specified for the compression. This means that all data (i.e., all bit-planes) present in the original images were included in the compressed code-streams. This choice has been made in order to assess the influence of every bit-plane in a classification task. As a result of this study, we will show that the last bit-planes could actually be dropped as they have a smaller discriminant power.

Integral volume parameters

Once the images have been compressed, integral volumes are extracted from the compressed code-streams and stored separately. We consider a maximum number of bit-planes equal to 8. This means that if a wavelet coefficient becomes significant in the i-th bit-plane ($i > 8$), it will be included in the set of coefficients becoming significant in the 8-th bit-plane. This exactly corresponds to Eq. 4.5 where N^{bp} is always taken equal to 8. As mentioned in Sec. 4.2.1, positive and negative coefficients are differentiated only in the LL-subband (which corresponds to the lowest resolution level). In all other kind of subbands (LH, HL, or HH), absolute values of coefficients are considered. Moreover, integral volumes extracted from these 3 kinds of subbands and belonging to the same resolution level are merged together by an averaging operation. It increases the rotation invariance and reduces the amount of space needed to store the integral volumes.

Classifier parameters

Eventually, integral volumes or pre-computed list of samples are provided to a program that builds an ensemble of random decision trees. As our goal was not to do an extensive study to find the ultimate set of parameters for each dataset, we tried several different settings and finally adopted the following ones, as they prove to give sufficiently good results on all datasets (see Section 4.4.1 page 110 for the meaning of these parameters):

- $T = 10$,
- $K = 30$,
- $n_{min} = 2$, which means that the trees are fully grown,
- $Sc_{min} = 0.25$.

During the learning phase, a total of 100 000 samples are extracted from the learning images while during the test phase, 100 samples are computed in each test image to decide to which class this image belongs. We invite the reader to refer to [41] for a deeper analysis of the influence of each of these parameters.

4.4.3 Results and discussion

We present in this section the results of our experiments. Global error-rates are first compared to other existing results. Then, as all features used in our system do not contribute to the same extent to the classification results, we propose an analysis of their discriminant power.

Comparison to other systems

To assess the accuracy that can be achieved with the proposed JPEG 2000 classifier, we compare our performances to other results. In particular, we focus on the system proposed by Marée *et al.* in [71]. Our system is indeed directly inspired from their work and mainly differs in the fact that features are extracted from a compressed code-stream and not from the pixel-domain. Consequently, the comparison to such system should give us a good indication on the analysis capacity that JPEG 2000 can offer in a classification task.

There are 5 different experiments: 4 databases including 2 different protocols for the COIL-100 dataset. For each of these 5 experiments, we compare 3 systems:

1. *Raw-RSw-ExTr*: this is the classification scheme described in [71]. Random subwindows ("RSw") are extracted from the uncompressed images ("Raw" images), rescaled to a size of 16x16, and used to build an ensemble of Extremely Randomized Trees ("Ex.Tr."). In case of color images, subwindows are also converted from RGB to another color space. In [71], authors chose the HSV color space as it proved to be more robust to illumination changes than RGB. In addition to HSV, we also consider the YCbCr color space in our work. This color space is the one to which color components are converted during a JPEG 2000 compression, just before applying the Discrete Wavelet Transform. By using this color space, this system is closer to our scheme and the comparison between both systems is more relevant. Indeed, we are able to truly compare the efficiency of two different kinds of features set: the features corresponding to average values in the pixel domain (like in [71]) and the ones corresponding to histogram bin values in the wavelet domain (like in our system). The classifier parameters are the same as those described above and random subwindows are extracted in the same way samples from integral volumes are.

2. *J2K-IVol-ExTr*: this is the proposed classification scheme. It starts from the JPEG 2000 code-streams ("J2K") from which it extracts Integral Volumes ("IVol") before building Extra-Trees. Settings used are the ones presented in the previous Section.

3. *Others*: this refers to the range of other results found in the literature that follow the same evaluation protocols. In each case, the paper corresponding to the lowest bound of this range is indicated. Note that there is no such other result for the PONCE database.

Table 4.1 shows the error rates achieved by each of these systems, according to the protocols described in previous section.

Several interesting aspects of these results can be pointed out.

- The performances achieved by our system are always close from the best result found in the literature. This shows that a JPEG 2000 framework, beside its compression purpose, is well suited to perform classification operations.

- Except for the first protocol of the COIL-100 dataset, the results obtained with the proposed classifier are always equal or better than in [71]. This tends to prove the higher discriminant power of wavelet

Table 4.1: *Classification error rates achieved by the system described in [71] (Raw-RSw-ExTr) and by our system (J2K-IVol-ExTr). The range of other results found in the literature is indicated in the last column. The citation refers to the best other result.*

	Raw-RSw-ExTr		J2K	Others
	HSV	YCbCr	IVol-ExTr	
PONCE	45.20%		11.20%	-
ZUBUD	5.22%	4.35%	4.35%	59%-0% [74]
ETH-80	29.09%	27.90%	19.02%	35.2%-13.6% [53]
COIL-100 (1)	14.55%	15.01%	22.51%	50.1%-24% [74]
COIL-100 (2)	0.0033%	0.0024%	0.0017%	12.5%-0.001% [74]

histogram bins compared to averaged pixel values. In particular, the error-rates obtained for the PONCE dataset underline the much higher ability of the proposed JPEG 2000 classifier to discriminate textures.

- The result obtained for the first protocol of the COIL-100 dataset illustrates the lack of robustness of the proposed classifier in presence of viewpoint changes, compared to [71]. Fig 4.21 shows the error rates achieved depending on the viewpoint from which the picture of the object is taken (let us remind that in this protocol, learning has been done on one image per class, corresponding to the viewpoint located at 0°). As we see, wavelet histogram bins are more sensitive to a viewpoint change than average pixel values. This should be further studied, for example by investigating a more efficient way to combine integral volumes from different subbands.

- Among the results for the "*Raw-RSw-ExTr*" system, no significant trend could be observed that would urge us to choose between HSV or YCbCr transformation. YCbCr is slightly better for ZUBUD and ETH-80 while HSV performs better with the COIL-100 dataset. Further investigation would be needed to better understand the influence of the color transformation. However, we can at least conclude that both transformations give good and similar results.

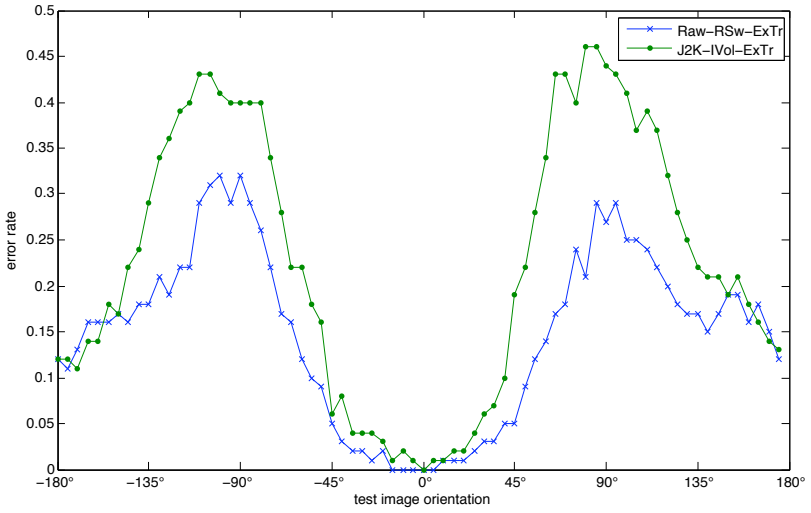


Figure 4.21: *Error rates depending on the viewpoint for the COIL-100 dataset. Learning has been done on the view at 0° of each object. The proposed JPEG 2000 classifier is more sensitive to a viewpoint change than the classifier based on rescaled random subwindows in the pixel domain.*

Feature discriminant power comparison

All features used in our system do not contribute to the same extent to the image classification task. Starting from this observation, we want to analyze how the discriminant power of a feature depends on the resolution, component or bit-plane involved in the feature computation and see if the observed trends do depend on the kind of images processed or not.

To do so, we build an ensemble of totally randomized trees for each dataset. This means that each split is picked totally at random (K=1). As the feature and threshold chosen in each node is totally random, we assume that the average score reached by the nodes using a certain kind of feature is representative of the discriminant power of this kind of feature.

Three global trends have been observed, whatever the dataset. The two first ones are presented in Figure 4.22. In this figure, the nodes have been partitioned in 9 subsets. The first one corresponds to all nodes for which the number of entropy-coded bytes is used as the feature ("header"). The ith one ($i = 2, ..., 9$) corresponds to all nodes whose feature is the number of new significant coefficients in bit-plane i. The figure compares the normalized average score reached by each of these subsets. "Normalized" means that for each dataset, the highest score has received the value of 1 and the score of the other subsets have been scaled accordingly. The results have then been averaged over all datasets but it should be noted that the same behavior was observed for each of them. Without surprise, we observe that

1. header information is less discriminant than features based on wavelet coefficients histogram bins,

2. the more significant the bit-plane, the more efficient features extracted from it will be.

The third trend concerns the color components. Figure 4.23 shows the average score reached by Cb and Cr components for each resolution level. In this graph, $r = 1$ corresponds to the lowest level (LL-subband). All (color) datasets have the same behavior, that has been averaged and normalized in the figure. As we see, high frequencies from these components are less discriminant than the corresponding low frequencies. Assuming that classes are differentiated through human visual criteria, this observation is consistent with the experiments done on the Human Visual System (HVS). The HVS has a differential sensitivity to spatial frequencies ([112], page 628), this sensitivity being much lower for high frequencies from Cb and Cr components.

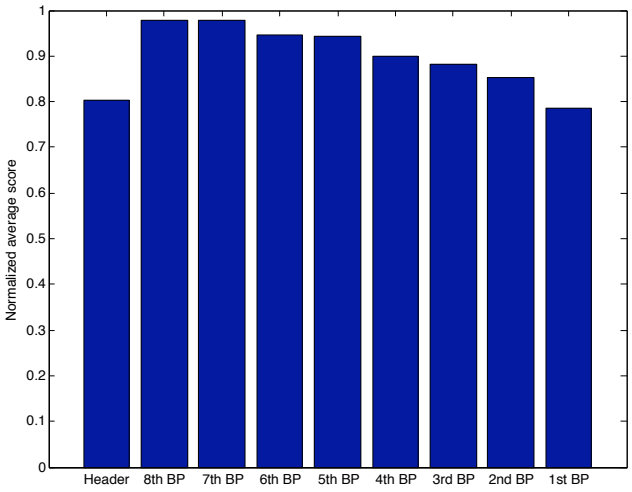


Figure 4.22: *Comparison of normalized average scores for the header information and successive bit-planes. Scores have been computed on totally randomized trees.*

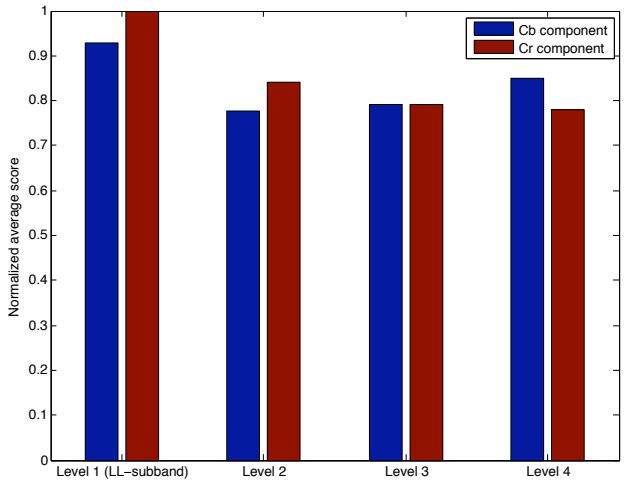


Figure 4.23: *Comparison of normalized average scores reached by color components in successive resolution levels. High frequencies (HL, LH, HH subbands are less discriminant than low frequencies (LL subband).*

Concerning the Y component, the LL-subband is generally also a bit more discriminant but differences are smaller and this is not the case of all datasets (ETH-80, for example, has on average a luminance component that is more discriminant in the high frequencies). It should also be noted that among LL-subbands of color datasets, the Cb and Cr components reach always an higher score than the luminance. This proves, for the considered classification tasks, the importance of the color information to discriminate classes. However, it cannot be seen as a global trend as we could easily imagine a classification task of color images that does not take into account this color information.

An important remark to be done is that the higher average score reached by a feature f_1 compared to a feature f_2 does *not* mean that feature f_2 is useless in the classification task or that we could reach the same overall performance by considering only feature f_1. The different kinds of feature are complementary, as they discriminate different aspects of the image. This has been shown in Section 4.2.3: the classification accuracy increases as more resolution levels and bit-planes are considered.

However, such higher score indicates which features will statistically give the best splits in a tree. And the results of this analysis confirm us that JPEG 2000, while designed for compression purpose, is also well suited for a classification task. The least discriminant bit-planes have indeed the highest extraction cost. Concerning color components, the lower discriminance of their high frequencies could be taken into account when designing a cascade of classifiers, by keeping these features for a latter stage, when the amount of potential zones of interest is smaller.

4.4.4 Conclusion

We introduced in this Section an image classifier that uses the JPEG 2000 integral volumes presented in Section 4.3. This classification is not only based on textures, but also on the color information. Moreover, it exploits the spatial localization of textures or colors inside each sample.

The results obtained with this classifier for several different datasets are in the same range or even better than other systems found in the literature. In particular, compared to [71] whose classification mechanism has been reused in our work, it performs far better for grayscale texture recognition. This underlines the suitability of JPEG 2000 features for such images. On the other hand, a lack of robustness to orientation changes has been identified: further research will investigate how to solve this problem.

Initially, we gave us the constraint to stay in a JPEG 2000 framework in

order to avoid heavy transcoding operations, speed up image handling, and exploit the algorithm scalability. Based on the results obtained that demonstrate the JPEG 2000 analysis ability, this constraint actually appears to be also an advantage in terms of classification performance.

The goal of this Section was thus to show the potential efficiency of features extracted from JPEG 2000 code-streams. However, this kind of classifier is not optimal at all, as we considered that all these features were directly available (i.e. features from all bit-planes and resolution levels). It does not exploit the inherent JPEG 2000 scalability and therefore implies a partial decompression of all images (until getting complete wavelet coefficients).

In the next Section, we propose an image retrieval system based on the above-described classifier but using also this scalability, so as to limit as much as possible the amount of information that has to be partially decompressed.

4.5 A cascade of image classifiers

In this Section, we -finally- address the problem of image retrieval. The basic formulation of this problem is always the same: based on a query submitted by a user, we want to retrieve in an image database all the samples that match the criteria specified in the query. In this work, we focus on example-based image retrieval: the query is made of one or more zones of interest selected by the user and the goal is to retrieve similar zones.

This example-based image retrieval problem can be seen as a particular case of the classification task envisioned in the previous section. Three additional hypotheses are set:

1. There are only two different classes, respectively containing the negative and positive samples. The latter ones should be returned by the system and presented to the user.

2. There are much more negative samples than positive ones: such image retrieval process is therefore equivalent to a rare event detection. As we shall see, by cascading classifiers, we precisely take into account this class asymmetry.

3. The amount of positive learning data is small, as it is usually limited to the samples provided by the user. On the contrary, once the “image

retriever" has been built based on the learning data, the amount of data to mine is huge: it corresponds to the remainder of the images in the database. Consequently, techniques able to speed up this retrieval step are of great interest in such application.

This last hypothesis shows one of the main challenges of an image retrieval system: maximize the results accuracy while minimizing the computational load. Assuming that most images are (or will be) stored in a compressed format, meeting this challenge implies to avoid heavy decompression procedures while digging inside image content. Investigating *compressed* image retrieval techniques makes therefore a lot of sense.

Beside an excellent compression efficiency, JPEG 2000 proves to be also well suited for such compressed image retrieval. Indeed:

- as shown in previous section, features that one can extract from a JPEG 2000 code-stream are discriminant in a variety of classification tasks,
- the various scalability levels inherently present in the algorithm allow to control and limit the amount of information that has to be processed.

In the remainder of the Section, we detail the cascade of JPEG 2000 classifiers that we propose as image retrieval system (4.5.1). Then, experiments are set up (4.5.2): the PONCE texture database is used to validate our approach. Finally, results are presented and discussed (4.5.3).

4.5.1 Method

Overview

As mentioned above, in an image retrieval task, there is a lot of data to mine and most of this data is negative. Rather than trying to find directly the few positive events in this ocean of negative items, a natural idea that comes up is to first try to reject the most obvious negative samples. Indeed, given the unbalanced repartition of samples, this first step is usually much easier than trying to find positive samples. After this first step, we are left with our positive samples disseminated in a much smaller amount of negative ones. Of course, this subset of samples is harder to discriminate but as it is much smaller, we can afford to spend more computational load on each of its samples.

This simple idea has lead to the *cascade of classifiers* concept. As shown in Fig. 4.24, a cascade of classifiers is made of a succession of stages, each of them being fed by the samples identified as positive by the previous stage. The first stage is a simple classifier able to rapidly scan the whole set of test samples and decide if each of them is surely negative or might be positive. The positive candidates are provided to a second stage during which a more accurate classifier will do the same job as in the first stage, and so on until the last stage is reached. Samples having reached this last stage and labeled as positive by the last classifier are considered as the final positive samples the user is looking for.

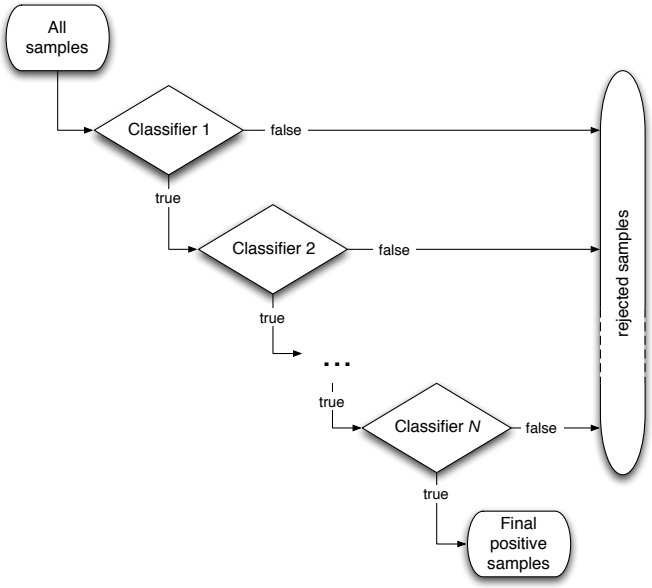


Figure 4.24: *A cascade of classifiers. Each stage tries to reject as many negative samples as possible.*

The goal of an image retrieval system is to maximize the detection rate (i.e. maximize the number of positive samples that are identified as such by the system) and minimize the false positive rate (the number of negative samples identified as positive by the system). A cascade framework allows to relax one of the objectives on each stage: each classifier has still to achieve the highest possible detection rate (if a positive sample is rejected, it is lost), but it can afford a moderate false positive rate (if a negative sample is accepted, next stages still have the opportunity to reject it). The

succession of stages will then guarantee a small global false positive rate (by multiplying the succession of moderate intermediate false positive rate). Given the highly unbalanced distribution of negative and positive samples, the stage objectives described above are much easier to achieve: a cascade is therefore well suited for such problem as it takes into account the class asymmetry.

The proposed system follows the general concept detailed above and integrates it in a JPEG 2000 framework. As explained in Chapter 2, the resolution and SNR scalability levels present in JPEG 2000 allow for a progressive decoding of a compressed image. This characteristic can be adequately exploited in a cascade of classifiers. Successive stages of our JPEG 2000 cascade will be based on an increasing amount of data extracted from the code-streams, beginning with the most easily accessible information. The first stages will perform a coarse classification, based on the header information extracted from one or several resolution levels. This first step, while not very accurate, is very cheap in terms of computation load as no entropy-decoding operation is required. Then, the next stages will focus on the remaining positive candidates and extract their most significant bit-planes (from one or several resolution levels). This additional information provides a coarse approximation of wavelet coefficients and allows to further discriminate the remaining samples using wavelet-based features. This second step is more accurate than the first one but is obviously also a bit heavier as a few bit-planes have to be entropy-decoded. Finally, the last stages will focus on the very best positive candidates and will try to reject those negative samples that are the most similar to positive ones. To do so, more (or even all) bit-planes of these positive candidates are decoded so as to obtain a very fine approximation of their wavelet coefficients and achieve an accurate classification. As we see, the proposed system uses a true **coarse-to-fine approach**: for a given sample, the amount of data that will need to be extracted (and entropy-decoded) is directly related to the relevance of this sample in the retrieval process.

An important point to note about the proposed JPEG 2000 cascade is the increasing discriminant power of features used in successive stages. Indeed, most other cascade systems use the same set of features to generate the succession of classifiers (see "Related Work" below). In this latter case, the higher accuracy of classifiers comes only from a harder training set. In our proposal, we not only provide harder samples to a classifier but also better tools to discriminate them. Figure 4.25 illustrates this increasing discriminant power. It shows the average number of splits required by a

totally randomized tree to perfectly classify a given set of samples. For our experiment, we used 14 000 samples from the PONCE database, 4000 from the *brick* class (considered as the positive class) and 10 000 from the remaining classes. The four results given correspond to four different sets of trees, each of them built with a different feature set. The first one uses only header information from the lowest resolution level (LL-subband). The second one uses header information from all levels. The third one is based on header information and the 3 most significant bit-planes. The last one uses all the extractable information: headers and all bit-planes. As we see, for a given set of samples, a richer feature set manages to perform the classification quicker. It should be noted that this analysis is not in contradiction with the one based on Fig. 4.22. Successive bit-planes bring a decreasing amount of valuable information but nevertheless improve the classification performance.

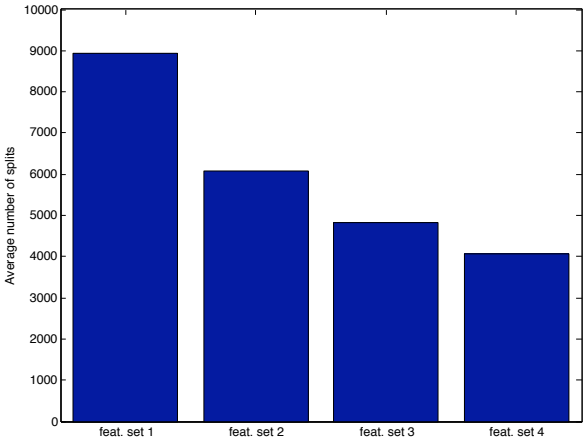


Figure 4.25: *Average number of splits to perform a perfect classification, depending on the feature set used. (1) Headers from LL-subband ; (2) Headers from all resolution levels ; (3) Headers and 3 most significant bit-planes ; (4) Headers and all bit-planes.*

Of course, using richer features implies an higher extraction cost. To take this into account, and to achieve the best performance while minimizing the computational load, we propose to include the cost in the assessment of the cascade performance. This evaluation occurs during the optimization step that tries to find the best threshold for each stage. By

doing so, we are able to tune our system to target a specific trade-off between accuracy and complexity.

Practically, our image retrieval system is obtained in two successive steps: building and optimization. The initial learning set is divided in two subsets. The first one is used during the first step to learn several ensembles of random decision trees (one ensemble for each stage of the cascade). Once stages have been built, the second subset is used to find the best set of thresholds to achieve a given trade-off between performance and computation cost. The principle of this post-learning **cost-performance optimization** might be compared to the rate-distortion optimization that occurs after the entropy-encoding step in the JPEG 2000 algorithm. Once the cascade has been built and optimized, it can eventually be used to scan a test set.

In the rest of the section, after having reviewed the related work found in the literature, the 3 phases of our system are detailed: building, optimization and scanning.

Related work

As a cascaded detection process is essentially that of a degenerate decision tree, it is obviously related to the literature on decision trees, that has been briefly introduced in Section 4.4.1. In particular, the work of Amit and Geman [2] proposed to use randomized trees that focus on areas characterized by the co-occurence of certain given features. By doing so, it avoids having to run the full detection process at all image locations and scales. At the same period, Rowley *et al.* [97] applied a notion similar to a cascade of detectors to the face detection application. They used one simple neural network to select the best candidates to feed a second, more complex, neural network, which is equivalent to a 2-stages cascade. Fleuret and Geman proposed in [35] a chain of tests based on disjunctions of fine scale edges to retrieve faces at a particular scale and location. Shortly after, Viola and Jones published their seminal work for robust real-time object detection [118, 119]. It consists in a cascade of boosted classifiers based on simple rectangle features. These features are extracted from integral images, that inspired the integral volumes proposed in the present work. Each stage is a weighted sum of single-feature-thresholding classifiers, trained through an AdaBoost process.

Since their work, a lot of improvements or variants have been proposed. Many of them explored alternative boosting algorithms that increase the performances of node training. This includes FloatBoost [56], Gentle-

Boost [59], Kullback-Leibler Boost [61], or AsymBoost [117]. In addition to this, [121] proposed a forward feature selection process that drastically reduces the amount of time required for the learning phase (weeks of training were needed in the initial system proposed by Viola and Jones). In a more recent work [10], Brubaker *et al.* questioned the relevance of single-feature-thresholding classifier and showed that the overall performance could benefit from stronger weak classifiers like CART (Classification And Regression Trees). In our work, we push this idea one step further by using an ensemble of random decision trees. Such random decision tree is also a stronger weak classifier than thresholding on a single feature but is also much easier and faster to build than a classic CART. Moreover, in our implementation, we did not consider any heavy boosting algorithm, which further reduces the complexity of the learning phase.

Concerning the JPEG 2000 framework, the principle of jointly considering a cascade detection process and a progressive JPEG 2000 content extraction has been introduced in our preliminary work [22]. Beside this contribution, to the best of our knowledge, only [51] suggests to exploit the JPEG 2000 scalability levels in a recognition task but authors did not envision a true coarse-to-fine approach made of several successive stages.

Schneiderman suggested in [101] that successive stages should use increasingly discriminant feature sets, at the cost of an increasing extraction complexity. This trade-off between complexity and performance has been formalized in a recent past. Hence, simultaneously to the work in [5], we have proposed in [22] to incorporate the feature extraction cost in the global cascade evaluation.

Finally, several recent papers have been written about the global cascade optimization [10, 32, 64]. In their initial paper, Viola and Jones did not deeply investigate this aspect. They suggested, for a given global detection rate D and a global false positive rate F, to train each of the n stages with individual local objectives equal to the n-th root of D and F. However, ideally, the best cascade will be obtained through a global trade-off between the number of stages, the thresholds used, and the kind and number of features involved. Such trade-off is obviously very difficult to obtain. In [10] and [32], the proposed optimization algorithms are tightly coupled with the training phase. This allows to get very close from a global optimal solution as training and optimization are mutually updating each other. However, it comes at a price of a high computational cost as training operations are included in optimizing loops and must be repeated many times. In [64] and in the system proposed in this work, the best thresh-

olds are found in a separate process, after the cascade design. As the final adopted thresholds are not necessarily the ones used for training, it obviously implies a suboptimality, but not sufficiently important to justify a huge increase in computational load.

Building the cascade of classifiers

During the building phase, a cascade of classifiers is generated, based on a given learning set. In our approach, we fix *a priori* the number K of stages, together with the part of the JPEG 2000 code-streams that will be available in each stage.

For each stage, an ensemble of random decision trees is learned. This process is the same as the one described in Section 4.4, with the noticeable restriction that the number of classes is always 2. The learning set used to generate each stage is thus made of positive and negative samples. The set of n_+ positive samples is the same in each stage while the n_- negative samples change from one stage to the other. They are extracted from a pool of negative images, a different pool being used for each stage. Moreover, samples extracted from the negative pool dedicated to stage k are used in this stage only if they successfully passed stages 1 to $k-1$. In this way, successive stages are trained with increasingly difficult learning sets.

This negative samples selection process implies to choose a set $\mathbf{T}_{learn}$ of learning thresholds. The testing procedure is done in the following way. Each tree of a stage has a certain amount of leafs, each of them being characterized by a class distribution[1]. For a given sample s that we want to test in stage k, the distribution of the leaf reached by the sample in each tree is aggregated over all trees, leading to a global confidence measure $C_{s,k}$ for this sample:

$$C_{s,k} = Pr\,(s \text{ positive, according to stage } k)$$

Each stage k is associated with a threshold T_k and if $C_{s,k} > T_k$, sample s is forwarded to stage $k+1$. Otherwise it is rejected.

Optimization

Once the cascade has been built, the optimization phase aims at finding the set $\mathbf{T}^*_{opt}$ of K thresholds that will lead to the best computation-performance trade-offs. To do so, we use an *optimization set*, different from

[1] In case of fully grown trees, the two probability values of this class distribution are 1 and 0.

the samples used to build the cascade (the *learning set*), and of course also different from the samples that will be used to assess the final cascade (the *test set*). However, in the perspective of an online retrieval system, the amount of available labeled samples is likely to be small as they are manually provided by the user. If they are further divided in a learning and a optimization sets, performances could significantly decrease. A solution to this could be to use the two distinct sets to compute optimized thresholds, and then to learn the final cascade with the whole set of labeled samples.

For a given set of thresholds $\mathbf{T}_{opt}$, the performances of a cascade are evaluated according to its classification error and to its cost in terms of computational load.

The classification error is measured by two values, namely the *precision* and the *recall*. Compared to the more common *detection rate* and *false positive rate* leading to the classical ROC curve, precision and recall are defined in the following way:

$$\begin{aligned} \text{Recall} &= \frac{TP}{TP+FN}, \\ \text{Precision} &= \frac{TP}{TP+FP}, \\ \text{Detection rate} &= \frac{TP}{TP+FN}, \\ \text{False positive rate} &= \frac{FP}{TN+FP}. \end{aligned}$$

where TP, FP, TN and FN are respectively the number of True Positive, False Positive, True Negative and False Negative samples. Precision and recall are preferred to the classical ROC curve because these measures are more suited for highly skewed datasets, as explained in [15]. For a given set of thresholds $\mathbf{T}_{opt}$ leading to Precision P and Recall R, the global classification error E is then given by

$$E(\mathbf{T}_{opt}) = 1 - [\lambda_{prec}.P(\mathbf{T}_{opt}) + (1 - \lambda_{prec}).R(\mathbf{T}_{opt})] \tag{4.14}$$

where λ_{prec} is fixed *a priori* and represents the relative importance of the precision compared to the recall ($0 \leq \lambda_{prec} \leq 1$). $E(\mathbf{T}_{opt})$ is a value comprised between 0 and 1, a better cascade leading to a value closer to 0.

The computational cost measures the fraction of bits that need to be entropically decoded to make a decision on all samples. A cost $C(\mathbf{T}_{opt}) = 1$ means that all bit-planes from all resolutions levels of all samples have to be decoded. For the sake of simplicity, the cost of header information extraction is considered equal to the cost of decoding one bit-plane. Practically,

the cost computation is done in the following way: for each stage k, a cost factor c_k is computed. This coefficient represents the portion of a sample that will have to be decoded if it reaches stage k. It takes into account the number of resolution levels (and their respective size), subbands, components and bit-planes that are involved in stage k. As these numbers are fixed *a priori* when designing a cascade, the c_k coefficients are constant values for a given cascade. For example, a stage k that uses header information and the 2 most significant bit-planes out of 8, from all resolution levels, subbands and components will be given a cost factor $c_k = \frac{3}{9} = 0.333$. The global cost is then given by

$$C(\mathbf{T}_{opt}) = \sum_{k \in [1,K], T_{opt,k} > 0} n_k.(c_k - c_{k'}) \tag{4.15}$$

where n_k is the number of samples that enter stage k ($k = 1, ..., K$) and k' refers to the closest previous stage with a threshold > 0 ($c_{k'} = 0$ if there is no such stage). The dependency on the set of thresholds $\mathbf{T}_{opt}$ comes obviously from the fact that a threshold change will change the n_k's. It should be noted that Eq. 4.15 assumes that each stage will at least re-use the data already decoded in previous stages. Moreover, for a given stage k, if $T_{opt,k}$ is taken equal to 0, this stage is considered to be skipped by the classification process and no cost is counted for this step. The number of stages defined in the building step is therefore only a maximum value as the optimization step could realize that better performances can be obtained by skipping some of these stages.

Based on Eq. 4.14 and 4.15, the global cascade inefficiency for a given set of thresholds $\mathbf{T}_{opt}$ is measured by the following expression

$$\begin{aligned} I(\mathbf{T}_{opt}) &= (1 - \lambda_{cost}).E(\mathbf{T}_{opt}) + \lambda_{cost}.C(\mathbf{T}_{opt}) \\ &= (1 - \lambda_{cost}).\left(1 - (\lambda_{prec}.P(\mathbf{T}_{opt}) + (1 - \lambda_{prec}).R(\mathbf{T}_{opt}))\right) \\ &\quad + \lambda_{cost}.C(\mathbf{T}_{opt}) \end{aligned} \tag{4.16}$$

where λ_{cost} is fixed *a priori* and represents the relative importance of the computational cost compared to the classification error ($0 \leq \lambda_{cost} \leq 1$). Consequently, the cascade optimization process consists in finding a set of thresholds $\mathbf{T}^*_{opt}$ such that

$$\mathbf{T}^*_{opt} = \underset{\mathbf{T}_{opt} \in \mathbb{R}^K_{[0,1]}}{\arg\min}\ I(\mathbf{T}_{opt}) \tag{4.17}$$

Equation 4.16 shows that the optimization process is influenced by two parameters: λ_{prec} and λ_{cost}, that balance the relative importance of cost,

precision and recall in Eq. 4.16. Although we did consider in our work that those two parameters are set *a priori*, it should be noted that the adopted formulation could be used in a lagrange-multiplier approach to solve a constrained optimization problem such as "Maximize the detection rate under the constraints of a limited amount of false positive samples and a bounded cost".

The optimization problem 4.17 can be solved with an iterative algorithm. As shown in Algorithm 4.1, each iteration is made of K steps: during step k, the algorithm seeks the threshold for stage k that will give the lowest inefficiency I, while all other thresholds are fixed to their "up-to-this-point best" value. The number of iteration is limited by an upper bound *maxiter* but can be smaller if the global performance gain between two successive iterations is smaller than a given Δ_{stop}. The initial set of thresholds is taken equal to the set used for learning:

$$\mathbf{T}_{opt}^{init} = \mathbf{T}_{learn}.$$

The procedure COMPUTE_INEFFICIENCY computes Equation 4.16 based on given values of λ_{prec} and λ_{cost}, and on the set of thresholds $\mathbf{T}_{opt}$ to be tested. It should be noted that this procedure can be executed very fast. Indeed, as the cascade has already been designed in the learning phase, we begin the optimization phase by testing all samples in all stages. By doing so, we obtain, for each sample and a given stage k, the confidence measure that this sample is positive, according to stage k. Then, the COMPUTE_INEFFICIENCY procedure can be executed very fast as it consists only in thresholding operations on pre-computed values.

Scanning

Once the cascade has been designed (*building* step) and the best set of thresholds $\mathbf{T}_{opt}^{*}$ has been found (*optimization* step), the image retrieval system can be tested on a test set to assess its real performance. The test set is made of unlabeled positive and negative samples. The performances achieved by the cascade (i.e. the precision, recall, and cost achieved) are simply obtained through a procedure similar to the COMPUTE_INEFFICIENCY procedure mentioned in Algorithm 4.1.

The generation of these test samples can be done in different ways. As explained below, in our experiments, we randomly generated samples from a pool of negative and positive test images. However, in more realistic conditions, these samples are progressively generated during a scanning

Algorithm 4.1: FIND_BEST_THRESHOLDS

Require: $0 \leq \lambda_{prec} \leq 1$, $0 \leq \lambda_{cost} \leq 1$
  $\mathbf{T}_{opt} \Leftarrow \mathbf{T}_{learn}$
  $I \Leftarrow$COMPUTE_INEFFICIENCY$(\mathbf{T}_{opt}, \lambda_{prec}, \lambda_{cost})$
  $I^* \Leftarrow I$
  $\mathbf{T}^*_{opt} \Leftarrow \mathbf{T}_{opt}$
  $\Delta_I \Leftarrow 1$
  $n \Leftarrow 1$
  while $\Delta_I > \Delta_{stop}$ & $n < maxiter$ **do**
    $LastI \Leftarrow I^*$
    for $k = 1$ to K **do**
      for $t = 0$ to 1 with step Δ_T **do**
        $T_{opt}(k) \Leftarrow t$
        $I \Leftarrow$COMPUTE_INEFFICIENCY$(\mathbf{T}_{opt}, \lambda_{prec}, \lambda_{cost})$
        if $I < I^*$ **then**
          $I^* \Leftarrow I$
          $\mathbf{T}^*_{opt} \Leftarrow \mathbf{T}_{opt}$
        end if
      end for
      $\mathbf{T}_{opt} \Leftarrow \mathbf{T}^*_{opt}$
    end for
    $\Delta_I \Leftarrow I^* - LastI$
    $n \Leftarrow n + 1$
  end while
  return $\mathbf{T}^*_{opt}$

procedure that analyses all images from a test database, at different locations and scales. The implementation of this scanning procedure is part of our future work but we would like to underline the adequacy of such procedure with the integral volume representation introduced earlier in this chapter. As already explained, integral volumes can be generated progressively, as the JPEG 2000 image is decoded. Moreover, samples of any size can be very easily extracted from such volume at any location. This is particularly useful in a cascaded scanning procedure:

- For a given stage k, integral volume of each image will be progressively generated until features used in stage k are available. It will of course re-use the integral volume used in stage $k-1$.
- Whatever the scanning strategy, samples are easily extracted from the integral volume and no rescaling operation is needed as feature values are normalized.
- When going from stage k to stage $k+1$, a certain amount of images areas are rejected and the integral volume has to be updated only for those areas containing samples that were labeled as positive in stage k. This can be done in at least two ways:
  1. We keep managing a single integral volume for the image, and the additional data that would have to be decoded for the rejected areas is simply replaced by zeros.
  2. As positive samples are likely to overlap each other, clusters of positive samples can be formed. The bounding box of each of these clusters can be considered as a new independent integral volume and new samples can be extracted from it in stage $k+1$.

4.5.2 Experiments set-up

To assess the performances of the image retrieval system described above, we have conducted some experiments. The database used is the PONCE texture database. The main objective is to design a cascade of classifiers able to retrieve samples belonging to a given class c^* among all samples belonging to other classes. Based on the 25 different classes from the original database, we therefore define two image categories:

$$c_+ = \{images \in c^*\}$$
$$c_- = \{images \notin c^*\}$$

By doing so, we fulfill the two first hypotheses of our problem (p. 126): two classes with a highly unbalanced distribution between positive and negative samples. Different choices of positive class c^* have been tested but we only present here results obtained for class $c^* = 25$ ("Fabric") because achieved performances are representative of the average performance reached with other classes. JPEG 2000 parameters and integral volume parameters are the same as the ones used in Section 4.4.

We arbitrarily fixed the number of stages in the cascade to 3, each having access to an increasing amount of data compared to the previous one:

1. The first stage uses only header information, from all resolution levels.
2. In addition to header information, the second stage has access to the 3 most significant bit-planes of each resolution level.
3. Eventually, the last stage computes features based on all information: header information and complete decoded wavelet coefficients.

As explained above, each stage is an ensemble of random decision trees. Following settings were used:

- $T = 50$,
- $K = 30$,
- $n_{min} = 2$, which means that the trees are fully grown,
- $Sc_{min} = 0.25$.

During optimization, we used following values for the parameters:

- $\Delta_{stop} = 0$
- $maxiter = 100$
- $\Delta_T = 0.02$

Concerning the samples sets, the database has first been divided in 3 parts: on the 40 images of each of the 25 classes, the first 13 were put in the learning set, the following 13 in the optimization set and the last 14 in the test set.

For the learning phase, 4000 positive samples and 10 000 negative ones are used to train each stage. The 4000 positive samples are extracted

from the 13 positive images and the same set is used to train all stages. Concerning negative samples, the 13×24 = 312 negative images are divided in 3 non-equal sets, the ith set being approximately 2 times bigger than the $(i-1)$th one. From the first set, 10 000 samples are extracted to train the first stage. From the second set, samples are extracted until 10 000 have passed the first stage and can be used to train the second one. And the same process is made with the third set except that samples have to pass the two first stages.

For the optimization and test sets, 4000 positive samples and 10 000 negatives ones are simply extracted from the corresponding images and used to optimize and then validate the cascade. It should be noted that unlike the JPEG 2000 image classification experiments presented in Section 4.4, samples are considered independently and are not aggregated in a majority vote rule to make a decision on the image they belong to.

4.5.3 Results and discussion

Once the cascade has been built and the thresholds optimized for a given pair of $(\lambda_{prec}, \lambda_{cost})$ (or, equivalently, for a given set of constraints on cost and precision), the test set is used to analyze the performances. Figure 4.26 compares various Precision-Recall curves. Each curve corresponds to a certain upper bound on the global cascade cost, as indicated in the legend. Points belonging to the same given curve are operating points, each fulfilling the cost constraint and corresponding to a certain trade-off between precision and recall (i.e. a certain λ_{prec} value). This trade-off will be set according to the considered application, depending on the relative importance of false-positive error compared to false-negative error.

The last curve indicated in the legend of Figure 4.26 is the one corresponding to a single JPEG 2000 classifier (no cascade). In this case, the classifier is trained on the whole learning set and with all features directly available. As it can directly use features from any bit-plane, the global cost is *de facto* equal to 1. Optimization therefore only consists in finding the best trade-off between precision and recall, for a given λ_{prec}. Each such trade-off corresponds to an operating point on the curve.

Although not mentioned on the graph, it should be noted that curves corresponding to cost-targets greater than 0.6 superimpose with the "$cost \leq 0.6$" one. Non-convexity observed at some places on the curves is due to the fact that thresholds for these operating points have been computed on the optimization set while results are given here for the test set.

Based on the curves from Fig. 4.26, several observations can be made

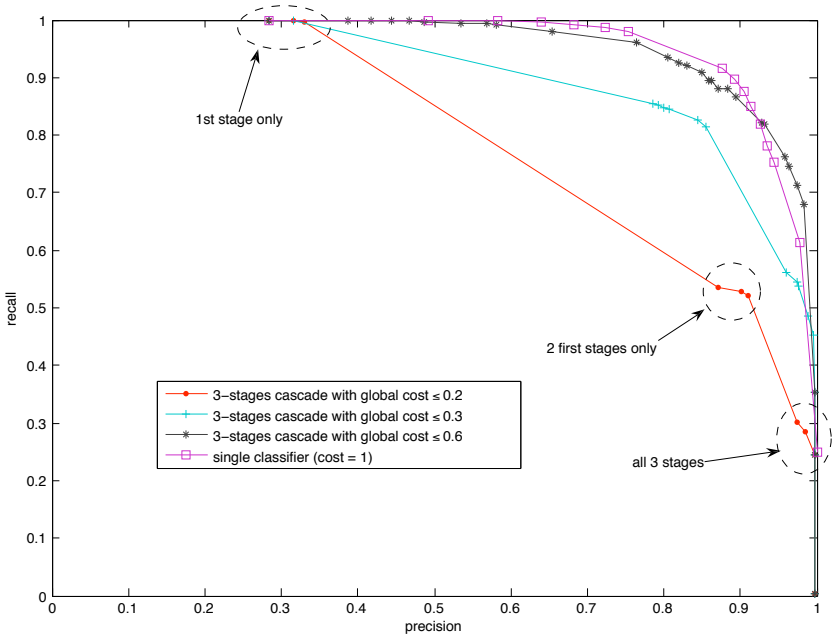


Figure 4.26: *Precision-recall curves, depending on a targeted cost. "Precision" is the ratio of true positives among all samples declared as positive. "Recall" is the ratio of true positives among all positive samples initially in the set. Database used is the PONCE texture database, positive class is the T25 class (Fabric), all other classes belong to the negative class. Comparison is made with a classification process without cascade (1 stage, cost=1).*

about the performances obtained with our cascade, compared to a conventional single classifier approach.

Cost benefit

First of all, from a cost point of view, the benefit of the cascade is clear: the curve targeting a cost ≤ 0.6 is already very close from the one obtained without any cascade and for which the cost is 1. This is made possible thanks to the first stages that avoid many negative samples from being further decoded. To illustrate this, we show in Table 4.2 the contribution of each stage to the classification process for a given operating point on the "*cost* ≤ 0.6" curve. As we see, after the first stage, 25% of negative samples have been rejected and after the second stage, 87% have been discarded. Only a few negatve samples will therefore be entirely decoded.

Table 4.2: *Effect of each stage in a classification process. The analyzed operating point is the one given by the optimization step with* $\lambda_{prec} = 0.56$ *and a maximum global cost equal to 0.6. The total number of samples is* 14 084*, with* 4 004 *positive samples and* 10 080 *negative ones.*

	Stage 1	Stage 2	Stage 3	**Cascade**
Entering samples	14 084	11 615	5 167	**14 084**
True positives	3 985	3 884	3 523	**3 523**
False positives	7 630	1 323	463	**463**
True negatives	2 450	6 307	860	**9 617**
False negatives	19	141	321	**481**
True pos. rate (%)	99.53	96.46	91.65	**87.99**
False pos. rate (%)	75.69	17.34	35.00	**4.59**
Decoded bits ratio (%)	11.11	27.49	20.38	**58.98**

Stage skipping triggered by the optimization step

Another observation is that on each curve (and particularly on "*cost* $\leq$ 0.2" and "*cost* $\leq$ 0.3"), we observe several "clusters" of operating points. Such cluster corresponds to some λ_{prec} values for which the optimization algorithm has decided to use only some of the available stages, as indicated on the figure. This is further illustrated in Fig. 4.27: for a given cost constraint, the global precision-recall curve is the convex-hull of 3 different

curves, each corresponding to the use of a given subset of stages in the cascade. As we see, depending on the cost constraint and the importance of precision compared to recall (i.e. the value of λ_{prec}), it might be more efficient to use only 1 or 2 out of the 3 stages.

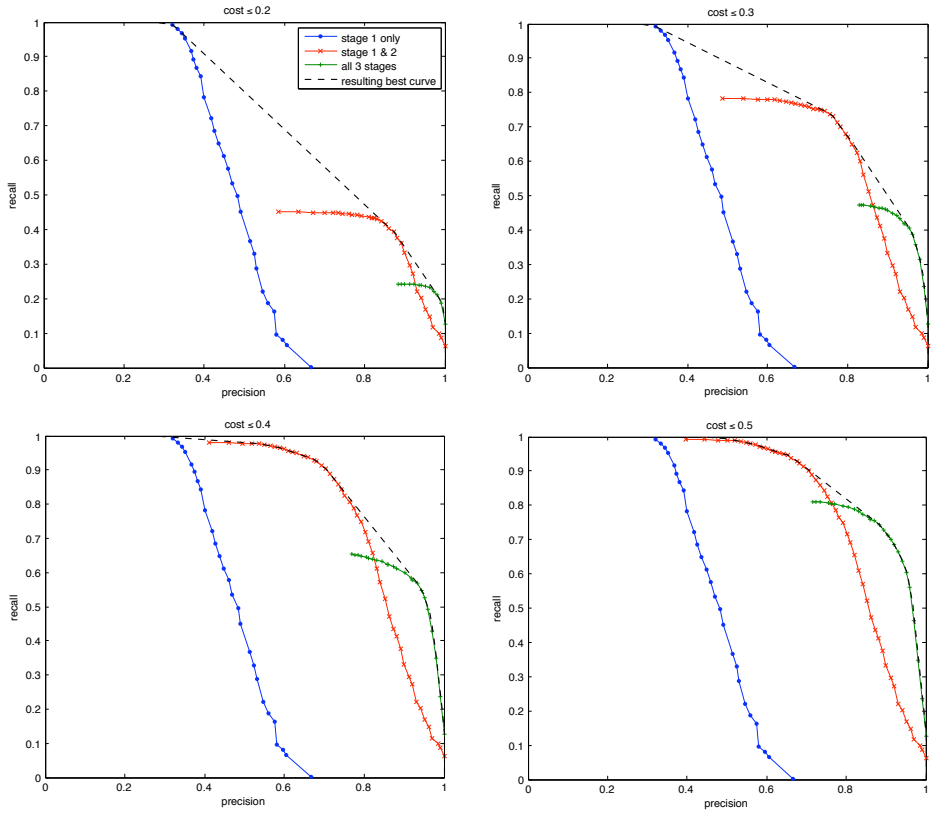


Figure 4.27: *Global precision-recall curve as the convex-hull of several different ones, each based on a fixed number of stages.*

Better precision at low or moderate recall values

An interesting point to observe on Figure 4.26 is the better performances obtained by the cascade for moderate recall values (i.e. detection rates): when the cost-target is not too low (0.6 for instance), the cascade leads to a better precision than the single classifier, for recall values up to 0.83. This

can be explained by the following arguments. When using a cascade, the whole classification problem is split in several successive smaller problems. This succession of sub-problems will focus on increasingly difficult samples. Although not identical, this operation is conceptually similar to a boosting process [37] in which a classifier will be progressively forced to give more importance to ambiguous samples. It leads to stronger classifiers and explains that, as long as targeted detection rates are not too high, the cascade outperforms a single classifier that has to directly classify the whole set of samples with a single ensemble of independent random decision trees[1].

When targeted detection rates increase, each stage of the cascade will have to guarantee a very small amount of false negative errors. To explain the cascade behavior in such context, let us remind that unlike the cascade proposed by Viola and Jones in [119], the set of features used in our case is increasingly discriminant along the stages. The first stages, that use less discriminant features than the last one, will therefore tend to choose smaller thresholds and let a greater number of samples reach the last stage. Indeed, this will increase the number of false positive errors but will guarantee a higher detection rate. This trend will progressively make the cascade more similar to a single classifier. We can therefore expect a decrease of the "boosting" effect described above and the cascade performances should get closer from the single classifier results.

Sub-optimality at high recall values

Based on the explanation from the last paragraph, we should expect that if the cost-constraint is completely relaxed, the obtained cascade will outperform the single classifier at moderate detection rates and then get closer of (or even superimpose) the single classifier curve for higher rates. Indeed, at high detection rates and as the cost is no more constraining, the cascade should become similar to a single stage that uses the whole set of features and therefore give the same results than a single classifier. However, we observe that the curve crosses the single classifier curve and remains below it when the detection rate increases[2]. This is due to the fact that the last stage has not been trained to classify all entering samples but only those having successfully passed the two first stages set up with

[1] In our future work, it should be therefore interesting to verify that this better result vanishes if we consider a single classifier based on a boosted ensemble of random decision trees.

[2] As stated above, the cost-unconstrained curve superimposes with the "$cost \leq 0.6$" one.

"normal" thresholds. If the cascade is subsequently degenerated to its last stage by the optimization step, it will remain sub-optimal in comparison to a single classifier whose training phase has been done on all samples. This sub-optimality should disappear if we replace our post-learning optimization step with an iterative process that include the training phase in the loop. However, as explained page 132, such strategy drastically increases the computational load required to build a cascade, as the training step has to be repeated several times.

4.6 Conclusion

In this chapter, we have investigated techniques able to realize classification and retrieval tasks based on JPEG 2000 code-streams. We have first studied how to extract relevant information from such code-streams in order to generate an image characterization that could subsequently be used in a classification process. We showed that, thanks to the various scalability levels inherently present in JPEG 2000, the quality of this characterization is directly related to the amount of data processed (and partially decoded) by the classifier. In particular, a new way to store the extracted feature values, called *integral volumes*, has been introduced and allows to very easily obtain relevant information on any area in the image. This structure is progressively refined, as each image is being further decoded. We then assessed the performances that can be expected when using JPEG 2000 features in various classification tasks based on random decision trees. The results obtained were similar to the best results available in the literature on raw images classification and categorization. This proves that staying in a JPEG 2000 framework not only avoid heavy transcoding operations and speed up image handling but allows also to get very good classification performances. Eventually, we combined several JPEG 2000 classifiers in a cascade suited for image retrieval operations. This cascade has been designed so as to draw benefit from the JPEG 2000 scalability: in particular, the amount of data that has to be extracted from a sample is directly related to the relevance of this sample. Optimization of this cascade is made in a process separated from the learning phase and takes into account the feature extraction cost: this allows to make a trade-off between complexity and performances, like in a rate-distortion optimization process. Such trade-off is of high interest in the perspective of an online retrieval system.

A lot of interesting future research directions have already been identified on several aspects of the presented system. Among them, one could cite

the study of a better way to combine information from various subbands in order to increase the robustness to orientation changes. Concerning the JPEG 2000 classifier that combines random decision trees and integral volumes representation, improvements could be obtained by investigating boosting solutions and/or techniques to more adequately select relevant samples for the learning phase [7]. The cascade also could be improved by considering for example an asymmetric score measure in each stage, that takes into account the lower importance of a false positive error compared to a false negative one [121, 122]. In parallel, other application frameworks have to be explored, like the famous face detection application, in order to be more easily comparable to other systems.

More globally, this work has to be seen as a step towards a user-centered online image retrieval system. While experiments in this chapter used offline computed samples sets, the goal is to be eventually able to interact with the user and to progressively learn what he is looking for. As described in the conclusion of this thesis, such online image retrieval system is complementary to a navigation tool such as the one proposed in Chapter 3 and could highly benefit from the use of hardware components such as the ones mentioned in Chapter 2. Most of the choices in the presented system have been made in the perspective of such user-centered system: the ability to generate a lot of random samples from a few learning images, the progressive generation of integral volumes, the possibility to return intermediate results in the cascade, the trade-off between performances and complexity, etc. The potential improvements raised in the previous paragraph will be implemented in the same perspective.

5 Conclusion

In this thesis, we have considered the very general issue of managing the growing amount of digital information that surrounds us. Within such a wide topic, we have focused on digital images, and more specifically on three complementary aspects that could be part of a generic remote image server:

- the representation used to store the images,
- the browsing ability that lets a user access and navigate inside these images,
- the retrieval ability that allows a user to find areas of interest among the images.

These three research directions have lead to several results, summarized in Section 5.1. Various perspectives have also been opened by this work, they are presented in Section 5.2.

5.1 Contributions

Scalable image representation

Among the three topics enumerated above, the contributions of this thesis have been made on the browsing and retrieval aspects. However, they both strongly rely on the scalability of the chosen image representation: JPEG 2000. Besides a high compression efficiency, this algorithm organizes the image information in such a way that almost any spatial area can be extracted at any resolution and bit-depth, without even scanning the remainder of the image. This image compression standard has been widely studied during our work. In particular, a deep understanding of the algorithm details -and the implementation issues it raises- has been made possible thanks to the design of an FPGA hardware architecture.

Seamless remote browsing

Techniques and guidelines have been proposed to improve the server responsiveness when remotely and interactively browsing large JPEG 2000 images. The main contribution of this work has been to propose and evaluate pre-fetching solutions to anticipate future navigation commands by scheduling JPEG 2000 packets that are expected to become relevant to expected future WoIs. Our results demonstrate that, by speeding up the increase of quality in future WoIs, pre-fetching has the capacity to accelerate the rhythm of user requests, which corresponds to an increase of the system reactivity and a smoother browsing experience for the user.

Coarse-to-fine compressed retrieval

In the last chapter, we have presented solutions to efficiently perform categorization and retrieval tasks on JPEG 2000 code-streams. We first investigated the features that could be extracted from such code-stream to characterize an image. We showed that thanks to the JPEG 2000 scalability, the discriminant level of such characterization is directly related to the amount of extracted data. Taking this finding into account, we proposed an original representation, called *integral volume*, that can be progressively refined and that stores the extracted feature values. Moreover, this representation allows to very easily obtain relevant information on any area in the image, independently from code-block borders.

A JPEG 2000 classifier has then be proposed. It combines the integral volume representation mentioned above, and recent developments in classification based on random decision trees. Performances obtained on freely available databases are similar to the best results known so far, validating the relevance of the approach.

Eventually, a cascade of JPEG 2000 classifiers has been introduced. It draws benefit from the JPEG 2000 scalability to perform a coarse-to-fine retrieval process. In particular, increasingly discriminant features are used along the cascade and the amount of processing achieved on a given sample is therefore directly related to the similarity of this sample with the submitted query. Optimization of such cascade is then made in a way that takes into account the global computational load. This aspect is of high interest in the perspective of an online retrieval system.

5.2 Perspectives

The work that has been done in the context of this thesis has opened several new research opportunities. Besides small algorithm tunings that have been mentioned in their respective chapter, we have identified three more global research perspectives that we would find interesting to investigate. This will hopefully be the case in the context of a Post-doc contract called MOSAIC (Module for Online Search and Access to Image Content).

The main motivations behind these future research directions are (*i*) a better integration of the three components investigated in this work, and (*ii*) an evolution towards an online user-centered system. Figure 5.1 reproduces Figure 1.1 from the introduction, with the suggested improvements marked in red.

Relevance feedback

The retrieval system proposed in our work has currently only be validated on pre-computed samples sets. For this system to be integrated in a remote image server, an online interaction with the user has to be defined. Practically, the learning set will have to be built in real-time. To do so, an iterative relevance feedback process is envisioned: the user submits an image query to the server which in turn replies with several intermediate results, then the user gives his feedback and the system further refines its classification model... And so on until the model is sufficiently accurate. The retrieval process can then be started on the whole database. It should be noted that most choices in the presented system have been made in the perspective of such user-centered system: the ability to generate a lot of random samples from a few learning images, the progressive generation of integral volumes, the possibility to return intermediate results in the cascade, the trade-off between performances and complexity, etc.

A lot of research has already been done in the *relevance feedback* field. In particular, *active learning* techniques [1, 85, 125] are well suited to our case and give promising results. They consist in carefully selecting the samples submitted to the user so as to maximize the expectation of the accuracy improvement. The user feedback is then only requested on ambiguous samples, which can drastically speed up the learning phase.

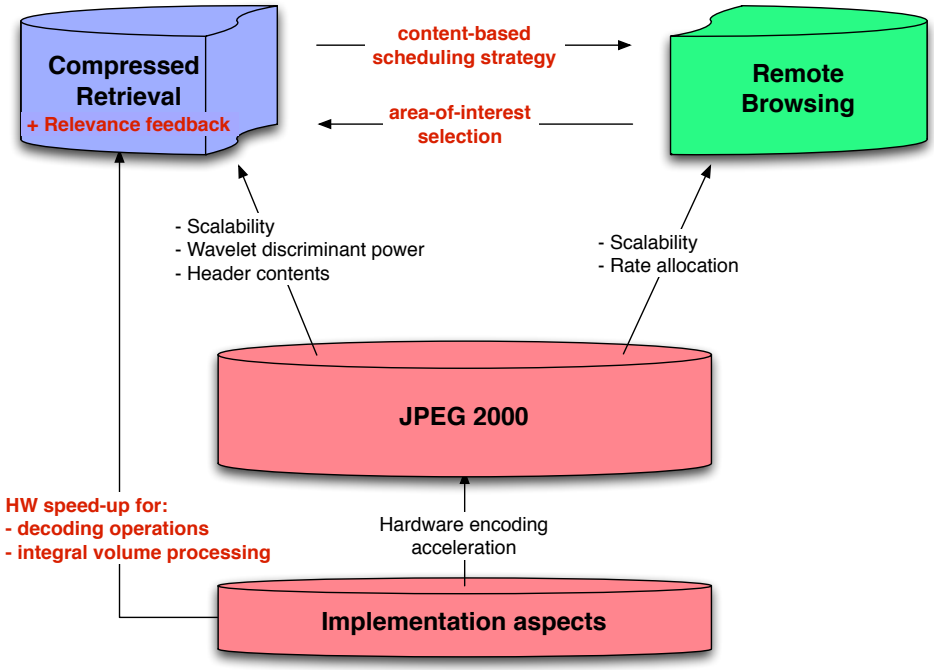


Figure 5.1: *Future research directions (marked in red).*

Browsing and retrieval interaction

Synergies could be exploited between the browsing and retrieval modules. For instance, in the presented system, when the user browses an image, packets are sent in decreasing order of the benefit provided on current and expected future WoIs. These future WoIs are guessed based on the navigation model used by the system. If browsing is now coupled with a retrieval task, the scheduling strategy could take into account the relevance of the packet content, according to the data submitted by the user about his interests. In this case, the system would first locate the areas of interest, then indicate their position to the user, and pre-fetch these zones in priority. On the other side, the retrieval module could draw benefit from the browsing ability by letting the user easily select the initial zone of interest before beginning to seek similar areas.

Hardware acceleration

As the amount and size of images in the server increase, dedicated hardware resources could be required to guarantee fast and efficient processing of users requests. As already suggested in Figure 1.1, such hardware acceleration could obviously be useful when a bunch of new images are recorded in the server, by speeding up the JPEG 2000 encoding process. Another part of the server that could greatly benefit from hardware power is the retrieval module (Figure 5.1). When processing the whole database with a cascade of random decision trees, we have shown that the proposed coarse-to-fine approach already limits the amount of JPEG 2000 decoding operations that has to be done. However, performances could be further improved by leaving hardware resources take care of these operations. Moreover, progressive building of integral volumes and feature values computation are made of repetitive simple steps that could adequately be transfered to a hardware processing unit. In this perspective, a collaboration with an industrial partner like the intoPIX spin-off should greatly facilitate this vertical integration.

Publications

Journal papers

- A. Descampe, C. De Vleeschouwer, P. Vandergheynst and B. Macq. Coarse-to-fine image classification in the JPEG 2000 compressed domain for fast processing of large datasets. To be submitted to *IEEE Trans. on Pattern Analysis and Machine Intelligence*.

- A. Descampe, C. De Vleeschouwer, M. Iregui, B. Macq, and F. Marqués. Pre-fetching and caching strategies for remote and interactive browsing of JPEG 2000 images. *IEEE Trans. on Image Processing*, 16(5):1339-1354, May 2007.

- A. Descampe, F. Devaux, G. Rouvroy, J.-D. Legat, J.-J. Quisquater, and B. Macq. A Flexible Hardware JPEG 2000 Decoder for Digital Cinema. *IEEE Trans. on Circuits and Systems for Video Technology*, 16(11):1397-1410, November 2006.

Conference papers

- A. Descampe, C. De Vleeschouwer, M. Iregui, B. Macq, and F. Marqués. Pre-fetching strategies for remote and interactive browsing of JPEG 2000 images. In *Proc. of IEEE International Conference on Image Processing (ICIP)*, pages 3201-3204, October 2006.

- A. Descampe, P. Vandergheynst, C. De Vleeschouwer, and B. Macq. Coarse-to-fine textures retrieval in the JPEG 2000 compressed domain for fast browsing of large image databases. In *International Workshop on Multimedia Content Representation, Classification and Security (IWMRCS)*, September 2006.

- A. Descampe, J. Ou, P. Chevalier, and B. Macq. Data Prefetching for Smooth Navigation of Large Scale JPEG 2000 images. In *Proceedings of the IEEE Conference on Multimedia and Expo*, pages 908-911, July 2005.

- A. Descampe, F. Devaux, G. Rouvroy, J.-D. Legat, and B. Macq. An Efficient FPGA Implementation of a Flexible JPEG 2000 Decoder for Digital Cinema. In *Proceedings of the 12th European Signal Processing Conference (EUSIPCO-2004)*, pages 2019-2022, September 2004.

- A. Descampe and F. Devaux. A Flexible, Line-Based, JPEG 2000 Decoder for Digital Cinema. In *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference (MELECON-2004)*, pages 1165-1170, May 2004.

Bibliography

[1] M. Almgren and E. Jonsson. Using active learning in intrusion detection. *Computer Security Foundations Workshop, 2004. Proceedings. 17th IEEE*, pages 88–98, 2004.

[2] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Comput.*, 9(7):1545–1588, 1997. ISSN 0899-7667.

[3] K.M. Au, N.F. Law, and W.C. Siu. Direct Image Retrieval in JPEG and JPEG2000. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 1, pages 529 – 532, Sept 2005.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[5] J. Bi, S. Periaswamy, K. Okada, T. Kubota, G. Fung, M. Salganicoff, and R.B. Rao. Computer aided detection via asymmetric cascade of sparse hyperplane classifiers. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 837–844, 2006.

[6] M. Boliek, C. Christopoulos, and E. Majani. JPEG 2000 image core coding system (Part 1). Technical report, ISO/IEC JTC1/SC29 WG1, July 2001.

[7] Vincent Botta. Segmentation active pour la classification d'images. Master's thesis, Université de Liège, 2006.

[8] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

[9] L. Breiman, JH Friedman, RA Olshen, and CJ Stone. CART: Classification and Regression Trees. *Wadsworth: Belmont, CA*, 1983.

[10] S. Brubaker, Charles Wu, Jianxin Sun, Jie Mullin, Matthew D. Rehg, and M. James. On the design of cascades of boosted ensembles for face detection. Technical report, Georgia Institute of Technology, 2005.

[11] E.H. Chi. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 69–75, Salt Lake City, Utah-USA, October 2000.

[12] P. Chou and Z. Miao. Rate-distortion optimized streaming of packetized media. *IEEE Transactions on Multimedia*, 8(2), April 2006. accepted for publication.

[13] T.T.A. Combs and B.B. Bederson. Does zooming improve image browsing? In *Proceedings of the fourth conference on Digital Libraries (DL)*, pages 130–137, CA-Berkeley, USA, August 1999.

[14] F.C. Crow. Summed-area tables for texture mapping. *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 207–212, 1984.

[15] J. Davis, V. Santos Costa, I.M. Ong, D. Page, and I. Dutra. Using bayesian classifiers to combine rules. 3rd SIGKDD Workshop on Multi-Relational Data Mining (MRDM 2004). In conjunction with KDD 2004., 2004.

[16] DCI. Digital Cinema System Specifications. Digital Cinema Initiatives (DCI), March 2005. URL `http://www.dcimovies.com/`.

[17] A. Descampe and F. Devaux. A Flexible, Line-Based, JPEG 2000 Decoder for Digital Cinema. In *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference (MELECON-2004)*, pages 1165–1170, May 2004.

[18] A. Descampe, F. Devaux, G. Rouvroy, J-D. Legat, and B. Macq. An Efficient FPGA Implementation of a Flexible JPEG 2000 Decoder for Digital Cinema. In *Proceedings of the 12th European Signal Processing Conference (EUSIPCO-2004)*, pages 2019–2022, September 2004.

[19] A. Descampe, J. Ou, P. Chevalier, and B. Macq. Data Prefetching for Smooth Navigation of Large Scale JPEG 2000 images. In *Proceedings of the IEEE Conference on Multimedia and Expo*, pages 908–911, July 2005.

[20] A. Descampe, G. Rouvroy, and B. Macq. A High-Throughput, Pipelined Architecture for the JPEG 2000 MQ Decoder. Technical report, TELE - UCL (Université catholique de Louvain), Belgium, May 2005.

[21] A. Descampe, F. Devaux, G. Rouvroy, J-D. Legat, J-J. Quisquater, and B. Macq. A Flexible Hardware JPEG 2000 Decoder for Digital Cinema. *IEEE Trans. on Circuits and Systems for Video Technology*, 16(11):1397–1410, November 2006.

[22] A. Descampe, P. Vandergheynst, C. De Vleeschouwer, and B. Macq. Coarse-to-fine textures retrieval in the JPEG 2000 compressed domain for fast browsing of large image databases. In *International Workshop on Multimedia Content Representation, Classification and Security (IWMRCS)*, September 2006.

[23] A. Descampe, C. De Vleeschouwer, M. Iregui, B. Macq, and F. Marques. Pre-fetching strategies for remote and interactive browsing of JPEG2000 images. In *Proc. of IEEE International Conference on Image Processing (ICIP)*, pages 3201–3204, 2006.

[24] A. Descampe, C. De Vleeschouwer, M. Iregui, B. Macq, and F. Marques. Pre-fetching and caching strategies for remote and interactive browsing of JPEG2000 images. *IEEE Trans. on Image Processing*, 16(5):1339–1354, May 2007.

[25] S. Deshpande and W. Zeng. Scalable streaming of JPEG2000 images using Hypertext Transfer Protocol. In *ACM*, pages 372–281, October 2001.

[26] Analog Devices. ADV212: JPEG 2000 video codec. Analog Devices website, 2006. URL `http://www.analog.com`.

[27] T.G. Dietterich and E.B. Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. *Dept. of Computer Science, Oregon State University Technical Report*, 1995.

[28] Thomas G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.

[29] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[30] Minh N. Do and Martin Vetterli. Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Trans. Image Process.*, 11(2), 2002. ISSN 1057-7149.

[31] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley Interscience, 2001. ISBN 0-471-05669-3.

[32] M. Murat Dundar and Jinbo Bi. Joint Optimization of Cascaded Classifiers for Computer Aided Detection. In *Computer Vision and Pattern Recognition Conference*, 2007.

[33] Umberto Eco. *The Name of the Rose*. Harcourt Brace, 1994.

[34] R. Ferreira, B. Moon, J. Humphries, A. Sussman, J. Saltz, R. Miller, and A. Demarzo. The virtual microscope. In American Medical Informatics Association, editor, *Proc. of the Annual Fall Symposium*, pages 449–453, Nashville, USA, 1997.

[35] Francois Fleuret and Donald Geman. Coarse-to-fine face detection. *International Journal of Computer Vision*, 41(1-2):85, Jan 2001.

[36] D.H. Foos, E. Muka, R.M. Slone, B.J. Erickson, M.J. Flynn, D.A. Clunie, L. Hildebrand, K.S. Kohm, and S.S. Young. JPEG 2000 compression of medical imagery. *Proceedings of SPIE*, 3980:85, 2003.

[37] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal Comput. System Sci.*, 55:119–139, 1997.

[38] J.H. Friedman. On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997.

[39] John F. Gantz, David Reinsel, Christopher Chute, Wolfgang Schlichting, John McArthur, Stephen Minton, Irida Xheneti, Anna Toncheva, and Alex Manfrediz. The Expanding Digital Universe, a Forecast of Worldwide Information Growth Through 2010. IDC white paper, March 2007.

[40] Laurent Gervereau. *Dictionnaire mondial des images*. Éditions Nouveau Monde, 2006.

[41] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 36(1):3–42, 2006.

[42] A. Grossman and J. Morlet. Decomposition of Hardy functions into square integrable wavelets of constant shape. *SIAM J. Math. Anal*, 15(4):723–736, 1984.

[43] B. B. Hubbard. *The World According to Wavelets: The Story of a Mathematical Technique in the Making*. A K Peters Ltd, 1997.

[44] intoPIX. IPX-JP2K datasheet: JPEG 2000 2K decoder module. intoPIX website, 2006. URL `http://www.intopix.com`.

[45] intoPIX. IPX-JP4K datasheet: JPEG 2000 4K decoder module. intoPIX website, 2007. URL `http://www.intopix.com`.

[46] intoPIX. IPX-JPHD datasheet: JPEG 2000 encoder/decoder. intoPIX website, 2007. URL `http://www.intopix.com`.

[47] Laurent Jacques. *Ondelettes, repères et couronne solaire*. PhD thesis, Institut de Physique Mathématique, Université Catholique de Louvain, June 2004.

[48] James Janosky and Rutherford W. Witthus. Using JPEG2000 for Enhanced Preservation and Web Access of Digital Archives. In *IS&T's 2004 Archiving Conference*, pages 145–149, April 2004.

[49] C. S. Jao, D.B. Hier, and S.U. Brint. The display of photographic-quality images on the web: a comparison of two technologies. *IEEE Transactions on Information Technology in Biomedicine*, 3(1):70–73, March 1999.

[50] J. Jiang, B.F. Guo, and S. Ipson. Shape-based image retrieval for JPEG-2000 compressed image databases. *Multimedia Tools and Applications*, 29(2):93–108, 2006.

[51] Jianmin Jiang, Baofeng Guo, and Pengjie Li. Extracting shape features in jpeg-2000 compressed images. In *ADVIS '02: Proceedings of the Second International Conference on Advances in Information Systems*, pages 123–132, London, UK, 2002. Springer-Verlag. ISBN 3-540-00009-7.

[52] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, August 2005.

[53] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2, 2003.

[54] Mun-Kew Leong. JPSearch Part 1: System framework and components - TR 24800-1:2007(E). Technical report, ISO/IEC JTC1/SC29 WG1, July 2007.

[55] Jin Li and Hong-Hui Sun. On interactive browsing of large images. *IEEE Transactions on Multimedia*, 5(4):581–590, December 2003.

[56] SZ Li and Z. Zhang. FloatBoost learning and statistical face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1112–1123, 2004.

[57] Chung-Jr Lian, Kuan-Fu Chen, Hong-Hui Chen, and Liang-Gee Chen. Analysis and Architecture Design of Block-Coding Engine for EBCOT in JPEG 2000. *IEEE Trans. on Circuits and Systems for Video Technology*, 13(3):219–230, March 2003.

[58] K. C. Liang and C. C.Kuo. Waveguide: a joint wavelet-based image representation and description system. *IEEE Trans. Image Processing*, 8(11):1619–1629, 1999.

[59] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. *DAGM 25th Pattern Recognition Symposium*, 2003.

[60] Chengjiang Lin and Yuan F. Zheng. Fast browsing of large scale images using server prefetching and client cache techniques. In *Applications of Digital Image Processing XXII SPIE*, pages 376–387, July 1999.

[61] C. Liu and H.Y. Shum. Kullback-Leibler boosting. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 1, 2003.

[62] Hao Liu, Xing Xie, Wei-Ying Ma, and Hong-Jiang Zhang. Automatic browsing of large pictures on mobile devices. In *Eleventh ACM international conference on Multimedia*, pages 148–155, November 2003.

[63] S. Livens, P. Scheunders, G. van de Wouwer, and D. Van Dyck. Wavelets for texture analysis, an overview. *Image Processing and Its Applications, 1997., Sixth International Conference on*, 2, 1997.

[64] H. Luo. Optimization design of cascaded classifiers. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 1, 2005.

[65] S. Mallat. Multiresolution approximation and wavelet orthonormal bases of L^2. *Transaction of the American Mathematical Society*, 315: 69–87, September 1989.

[66] S. Mallat and W.L. Hwang. Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38(2):617–643, March 1992.

[67] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1998. 577p.

[68] M. K. Mandal and C. Liu. Efficient image indexing techniques in the JPEG2000 domain. *Journal of Electronic Imaging*, 13:179–187, January 2004. doi: 10.1117/12.565762.

[69] MK Mandal, T. Aboulnasr, and S. Panchanathan. Image indexing using moments and wavelets. *Consumer Electronics, IEEE Transactions on*, 42(3):557–565, 1996.

[70] R. Maree, P. Geurts, J. Piater, and L. Wehenkel. A generic approach for image classification based on decision tree ensembles and local sub-windows. *Proc. of the 6th Asian Conf. on Computer Vision*, 2: 860–865, 2004.

[71] Raphaël Marée, Pierre Geurts, Justus Piater, and Louis Wehenkel. Random subwindows for robust image classification. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 1, pages 34–40, June 2005.

[72] Raphaël Marée, Pierre Geurts, Justus Piater, and Louis Wehenkel. Decision trees and random subwindows for object recognition. In *ICML workshop on Machine Learning Techniques for Processing Multimedia Content (MLMM2005)*, 2005.

[73] Raphaël Marée, Pierre Geurts, Justus Piater, and Louis Wehenkel. Biomedical image classification with random subwindows and decision trees. In *Proc. ICCV workshop on Computer Vision for Biomedical Image Applications (CVIBA 2005)*, volume 3765 of *LNCS*, pages 220–229. Springer-Verlag, oct 2005.

[74] J. Matas and S. Obdrzalek. Object recognition methods based on transformation covariant features. In *Proc. 12th European Signal Processing Conference (EUSIPCO)*, 2004.

[75] J. Meessen, T. Suenaga, M. Iregui, C. De Vleeschouwer, and B. Macq. Layered architecture for navigation in JPEG2000 mega-images. In *4th European Workshop in Image Analysis for Multimedia Interactive Services (WIAMIS)*, pages 92–95, April 2003.

[76] Jerome Meessen, Li-Qun Xu, and Benoit Macq. Content Browsing and Semantic Context Viewing through JPEG 2000 Based Scalable Summary. *IEE Proc. Vision, Image and Signal Processing*, 153(3): 274–283, June 2006.

[77] J. L. Mitchell and W. B. Pennebaker. Software implementations of the Q-coder. *IBM J. Res. Develop.*, 32(6):753–774, November 1988.

[78] M. Moshfeghi and J. Ta. Efficient image browsing with JPEG2000 Internet protocol. In *Proceedings of the SPIE, Medical Imaging: PACS and Imaging Informatics.*, volume 5371, pages 31–34, San Diego, CA-USA, February 2004.

[79] H. Murase and S.K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[80] A. Natu and D. Taubman. Unequal protection of JPEG2000 codestreams in wireless channels. In *IEEE Global Telecommunications Conference (GLOBECOM)*, volume 1, pages 534–538, Taipei, Taiwan, November 2002.

[81] R. Neelamani and K. Berkner. Adaptive representation of jpeg2000 images using header-based processing. In *IEEE International Conference on Image Processing (ICIP)*, volume 1, pages I–381– I–384, 2002.

[82] P. Norris. *Digital divide.* Cambridge University Press Cambridge, England, 2001.

[83] K. Ong, W. Chang, Y. Tseng, Y. Lee, and C. Lee. A high throughput context-based adaptive arithmetic codec for jpeg2000. In *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS 2002), vol4*, pages 133–136, May 2002.

[84] J.P. Ortiz, V.G. Ruiz, and I. Garcia. An efficient technique for remote browsing of JPEG2000 images on the web. In *Proceedings XV Jornadas de Paralelismo*, pages 322–327, Almeria, Spain, September 2004.

[85] D. Pelleg and A. Moore. Active Learning for Anomaly and Rare-Category Detection. *Advances in Neural Information Processing Systems*, 18, 2004.

[86] Alejandro Piscitelli. *Internet, la imprenta del siglo XXI.* Gedisa, 2005.

[87] C. Plaisant, D. Carr, and B.Shneiderman. Image-browser taxonomy and guidelines for designers. *IEEE Software*, 12(2):21–32, March 95.

[88] Eugenia A. Politou, George P. Pavlidis, and Christodoulos Chamzas. Jpeg2000 and dissemination of cultural heritage over the internet. *IEEE Trans. on Image Processing*, 13(3):293–301, March 2004.

[89] R. Prandolini. 15444-9:2004 jpeg 2000 image coding system - part 9: Interactivity tools, apis and protocols. Technical report, ISO/IEC JTC1/SC29 WG1, March 2004.

[90] JR Quinlan. Induction of decision trees. *Machine Learning*, 1(1): 81–106, 1986.

[91] J.R. Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1993.

[92] M. Rabbani and R. Joshi. An overview of the jpeg 2000 still image compression standard. *Signal Processing: Image Communication*, 17 (1):3–48, January 2002.

[93] U. Rauschenbach and H. Schumann. Demand-driven image transmission with levels of detail and regions of interest. *Computers and Graphics*, 23(6):857–866, December 1999.

[94] R. Rosenbaum and H. Schumann. JPEG2000-based image communication for modern browsing techniques. In *Proceedings SPIE - Electronic Imaging*, San Jose, CA-USA, January 2005.

[95] R. Rosenbaum and H. Schumann. Grid-based interaction for effective image browsing on mobile devices. In *Proceedings SPIE - Electronic Imaging: Multimedia on mobile devices*, San Jose, CA-USA, January 2005.

[96] R. Rosenbaum and D. Taubman. Remote display of raster images using JPEG2000 and the rectangular fisheye-view. *Journal of the Int. Conf. WSCG*, 11(3):387–393, 2003.

[97] HA Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):23–38, 1998.

[98] Joanna Rutkowska. Introducing blue pill. In COSEINC Research, editor, *SyScan'06 conference*, 2006.

[99] E. Salemi, C. Desset, J. Cornelis, and P. Schelkens. Additive distortion modeling for unequal error protection of scalable multimedia content. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Philadelphia, PA, USA, March 2005.

[100] D. Santa-Cruz, R. Grosbois, and T. Ebrahimi. Jpeg 2000 performance evaluation and assessment. *Signal Processing: Image Communication*, 17(1):113–130, January 2002.

[101] Henry Schneiderman. Feature-centric evaluation for efficient cascaded object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2004.

[102] H. Schwarz, D. Marpe, and Thomas Wiegand. Overview of the Scalable Video Coding extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9): 1103–1120, September 2007.

[103] H. Shao, T. Svoboda, and L. Van Gool. ZuBuD—Zurich Buildings Database for Image Based Recognition. *Technique report No. 260, Swiss Federal Institute of Technology*, 2003.

[104] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG 2000 still image compression standard. *Signal Processing Magazine, IEEE*, 18 (5):36–58, 2001.

[105] Sridhar Srinivasan, Chengjie Tu, Zhi Zhou, Dipankar Ray, Shankar Regunathan, and Gary Sullivan. An Introduction to the HD Photo Technical Design. ISO/IEC wg1n4183, Microsoft Corporation, 2007.

[106] Peter Symes. Jpeg 2000, the professionnal compression scheme. *Content Technology Magazine*, 3(3), June 2006. URL `http://svc126.wic003tv.server-web.com/CT-pdf/CT-May-June-2006.pdf`.

[107] Ali Tabesh, Ali Bilgin, Karthik Krishnan, and Michael W. Marcellin. Jpeg2000 and motion jpeg2000 content analysis using codestream length information. In *Proceedings of the Data Compression Conference (DCC'05)*, 2005.

[108] J. Tang, W. Zhang, and C. Li. An Approach to Compressed Image Retrieval Based on JPEG2000 Framework. *LNCS*, 3584:391, 2005.

[109] M. Tarui, M. Oshita, T. Onoye, and I. Shirakawa. High-speed implementation of jbig arithmetic coder. In *Proc. of TENCON'99, Region-10, vol. 2*, pages 1291–1294, 1999.

[110] D. Taubman. Remote browsing of JPEG2000 images. In *IEEE Int. Conf. on Image Processing (ICIP)*, volume 1, pages 229–232, September 2002.

[111] D. Taubman. High performance scalable image compression with ebcot. *IEEE Trans. on Image Processing*, 9(7):1158–1170, July 2000.

[112] D. Taubman and M. W. Marcellin. *JPEG 2000: Image Compression Fundamentals, Standards and Practice.* Kluwer Academic, Boston, MA, USA, 2002.

[113] D. Taubman and R. Prandolini. Architecture, philosophy and performance of JPIP: Internet protocol standard for JPEG2000. In *International Symposium on Visual Communications and Image Processing (VCIP)*, Lugano, Switzerland, July 2003.

[114] D. Taubman and R. Rosenbaum. Rate-distortion optimized interactive browsing of jpeg2000 images. In *IEEE International Conference on Image Processing (ICIP2003)*, September 2003.

[115] A. Teynor, W. Muller, and W. Kowarschick. Compressed Domain Image Retrieval Using JPEG2000 and Gaussian Mixture Models. *Lecture Notes in Computer Science*, 3736:132, 2006.

[116] Alexis Tzannes and Touradj Ebrahimi. Motion JPEG2000 for Medical Imaging. ISO/IEC wg1n2883, Medical Imaging Ad Hoc Group, 2003.

[117] P. Viola and M. Jones. Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade. *Neural Information Processing Systems*, 2002.

[118] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on CVPR*, pages 609–615, 2001.

[119] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[120] T. Wiegand, G. Sullivan, G. Bjøntegaard, and A. Luthra. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits Systems and Video technology*, 13(7):560–576, July 2003.

[121] J. Wu, J. Rehg, and M. Mullin. Learning a rare event detection cascade by direct feature selection. Neural Information Processing Systems, 2003.

[122] Jianxin Wu, Matthew D. Mullin, and James M. Rehg. Linear asymmetric classifier for cascade detectors. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 988–995, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-180-5.

[123] Xilinx. Virtex-5 platform FPGAs: Advance Product Specification. Xilinx website, 2007. URL `http://www.xilinx.com`.

[124] Ziyou Xiong and Thomas S. Huang. Subband-based, memory-efficient jpeg2000 images indexing in compressed-domain. In *SSIAI*, pages 290–294, 2002.

[125] C. Zhang and T. Chen. An active learning framework for content-based information retrieval. *Multimedia, IEEE Transactions on*, 4 (2):260–268, 2002.

[126] D.R. Zhang and X.X. Wang. The manipulation approach of JPEG 2000 compressed remote sensing images. *Proc. SPIE*, 6044:315–324, 2005.

[127] R. Zhao and W.I. Grosky. Bridging the Semantic Gap in Image Retrieval. *Distributed Multimedia Databases: Techniques and Applications*, 2003.