N d'ordre : 3065

THÈSE

EN CO-TUTELLE

entre

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

et

L'UNIVERSITÉ DE BRESCIA

PRÉSENTÉE À

L'UNIVERSITÉ DE BRESCIA

Par **Francesca MANERBA**

Pour obtenir le grade de

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Efficient Object Identification in Image Sequences for Content Indexing

---

Soutenue le : 30 Novembre 2005

Après avis des rapporteurs :

Michel Barlaud	Professeur
Stefano Tubaro	Professeur

Devant la commission d'examen composée de :

Jenny Benois-Pineau ..	Professeur	Examinateur
Michel Barlaud	Professeur	Président
Stefano Tubaro	Professeur	Rapporteur
Jean-Pierre Braquelaire	Professeur	Examinateur
Riccardo Leonardi	Professeur	Examinateur
Francesco De Natale ...	Professeur	Examinateur

– 2005 –

Università degli Studi di Brescia - Facoltà di Ingegneria
Corso di Dottorato di Ricerca in Ingegneria dell'Informazione
Ciclo XVII
Dipartimento di Elettronica per l'Automazione

Tesi di Dottorato di Ricerca

Efficient Object Identification in Image Sequences for Content Indexing

Tutor
Ch.mo Prof.
Riccardo Leonardi

Tesi di dottorato di
Francesca Manerba

Tutor
Prof.
Jenny Benois-Pineau

Coordinatore del dottorato
Ch.mo Prof.
Valeria De Antonellis

Anno Accademico 2004-2005

Ai miei genitori,
a Flavia,
a Giambattista

Ringraziamenti

Inizialmente non volevo scrivere queste pagine, non perché non sentissi di dover ringraziare qualcuno, quanto piuttosto perché dopo aver passato sei mesi a scrivere, riscrivere, cancellare e correggere, l'idea di rimettermi alla tastiera non mi piaceva proprio per nulla! Ma credo sia comunque doveroso dire qualche parola perché ritengo che il merito di questo traguardo non sia soltanto mio. Molti sono coloro che hanno, bene o male, attraversato la mia vita in questi tre, anzi no, quattro anni, per cui sarà difficile, per non dire impossibile, ringraziarli tutti; pertanto coloro che non si trovano qui sotto considerino che comunque sono stati ringraziati più e più volte nella mia mente se non lo sono in queste righe. E ora cominciamo.

Per prima cosa, i miei ringraziamenti vanno al **Prof. Riccardo Leonardi** e alla **Prof. Jenny Benois-Pineau** per l'opportunità che mi è stata data. Mi rendo conto di aver imparato molto e credo che questa co-tutela sia stata una valida esperienza, anche se, col senno di poi, qualcosa probabilmente cambierei. Sento comunque di dover ringraziare entrambi per la grande disponibilità.

Ringrazio inoltre, in ordine rigorosamente sparso, tutto il gruppo di lavoro, e in particolare **Sergio**, per l'aiuto morale e materiale, per le correzioni e i consigli, e per aver sopportato tutta le mie arrabbiature, che sono state tante, **Marzia**, per avermi costantemente pungolata in questi anni e per la sua risata che si sente anche nel mio ufficio (ovvero tre porte più in là), **Marco**, per le discussioni costruttive, **Manuel** per le cene e per 'References', **Dario**, perché si lascia bistrattare senza protestare troppo.

Un grazie anche al gruppo di lavoro d'Oltralpe: grazie a **Lionel**, per i botta e risposta via mail quando il boss era in giro, grazie a **Nicolas** per le 'rischiose' imitazioni, a **Laurent** per il supporto tecnico e a **Petra** per aver condiviso le ore piccole in ufficio. Un grazie inoltre ad **Alexis** e **Pierre**.

Grazie a coloro che sono stati la mia famiglia per un semestre: grazie a **Concetta**, **Luciana**, **Daniela** e **Sara**. Un sentito grazie anche ad **Javi**, per essere stato il punto fisso dei miei viaggi a Bordeaux, e a **David**, per la sua sana follia, con l'augurio che a Cambridge possa finalmente sentirsi realizzato. E come non ringraziare **Colouche**, il kebabbaro vicino all'Utopia!

Un grazie anche a coloro che dividono la corsia con me: grazie a **Marco** per avermi fatto compagnia nella scrittura ad agosto, grazie ad **Alessio** per la sua tranquillità quasi esasperante, grazie a **Sara** per avermi insegnato che i momenti difficili sono ben altri, grazie a **Pier** per le sue battute pungenti, grazie ad **Ago** perché non manca mai ad una cena, grazie a **Flavio** e **Michela** per aver condiviso la bella

avventura del piccolo **Marco**, grazie a **Laura** senza la quale saremmo tutti allo sbando. Ringrazio inoltre tutti i compagni di cene e bevute ossia **Massimo**, **Lazza**, **Paolo**, **Lorenz**, **Richi**, **Davide** e **Chiara**. Grazie a **Elena**, **Marco**, **Stefano** e al **Lazzino**.

Grazie a **Silvia** perché anche se sono cambiate molte cose in questi anni, so che posso sempre contare sul suo aiuto.

Grazie a **Sergio** perché trova sempre il modo di farmi sorridere.

Un grazie di cuore ai miei cugini: a **Marco**, anche se ormai non riusciamo quasi più a vederci, a **Gloria**, che invece mi sa che vedrò ancora per un bel pezzo, a **Giulia**, lontana ma sempre presente.

E ora i ringraziamenti davvero importanti: grazie a mio **papà**, compagno constante di viaggio, perché ha sostenuto con entusiasmo il mio desiderio di rimanere 'studente' per così tanto tempo ancora; grazie a mia **mamma**, per tutto ciò che rappresenta e per tante altre cose che non posso star qui ad elencare; grazie a **Flavia**, perché ogni tanto mi serve qualcuno con cui litigare, e per aver scoperto la 'faccina' con la zucca; grazie a **Giambattista**, anche se dovrebbe essere lui a ringraziarmi visto che sono il bersaglio costante delle sue battute; grazie a **Stefania** che ormai, dopo che si è appropriata del mio posto a tavola, non posso non considerare di casa.

Grazie a mia zia **Carmela** per aver reso questi ultimi sei mesi più ricchi.

Francesca

Perchè se si mettesse quel lavoro insieme a tutti gli altri [...] allora lo si vedrebbe per quello che è: una voce sperduta in un coro immenso. E tale era il canto che quel coro levava all'unisono che il solo farne parte mi bastava.

Abstract

The advances in data capturing and storage have made large amounts of video data available to consumer applications. However, interacting with multimedia data, and video in particular, requires tools to describe, organize and manage video data. Since, due to storage and diffusion problems, all multimedia documents are provided in compressed form, no retrieval nor indexing can be performed until the documents have been decompressed first. Besides, as often the users require their retrieval results in the fastest possible way, extracting information in real time, even if it is not precise, has become an important objective.

Based on the assumptions above, a new trend in analysis methods for indexing multimedia content has appeared which can be qualified as a "rough indexing" paradigm. Thus many have started to work with rough data - that is data extracted directly from an MPEG compressed stream - to perform tasks such as shot-cut detection, camera motion estimation, scene grouping or foreground object extraction.

In this context of fast and rough indexing we can include our work where only "rough data" - that is motion vectors and DC images - have been used for fine indexing of foreground objects. In this paradigm, we propose to combine both motion information and region-based color segmentation to extract meaningful objects from compressed video with arbitrary camera and object motion. The first step is to extract from P frames, using a robust camera motion estimation algorithm, the regions that present a motion model that does not follow the camera one. We call these regions "object masks" because it is supposed that foreground objects can be located with high probability in these regions. Then a morphological color segmentation algorithm is performed on I frames to refine the result of the motion mask segmentation that can be projected onto such I frames.

It may happen that if the object movement does not differ a lot from camera motion or the object is still, no objects get detected. But, as we can suppose that an object cannot reasonably appear and disappear along a short sequence of frames, we can filter the sequence of objects with I frame temporal resolution in the direction of time and try to identify them even when no motion information is present. The objective here is to build a rough object trajectory along the sequence of I frames where it was detected and then to approximate the shape with a conic function in those frames where the objects could not be detected. It means building a sort of tube where each section gives the object position in the frame and its approximated shape along time.

The described method gives promising results: in general the goal of detecting

foreground objects at I frame resolution is achieved even if slight imprecisions exist in object boundaries due to the reduced resolution. Isolated miss-detection or over-detection episodes can be corrected by the proposed spatio-temporal filtering.

Compendio

Negli ultimi anni si è assistito ad un notevole sviluppo della tecnologia applicata agli strumenti atti a registrare ed immagazzinare enormi quantità di dati multimediali. Elaborare tali moli di dati per estrarne informazioni utili relativamente a un certo criterio, costituisce un processo che può risultare altamente dispendioso in termini di tempo. A tale scopo quindi si è cominciato a sviluppare una serie di applicazioni aventi la finalità di estrarre in maniera automatica informazioni salienti da dati multimediali quali per esempio un video. Se a questo scenario si aggiunge il fatto che i dati di un video vengono spesso trasmessi e immagazzinati in forma compressa, è chiaro che una qualsiasi analisi che si voglia effettuare su di essi, implica che tali dati vengano prima decompressi e poi elaborati. Infine, nel caso i risultati di tali elaborazioni vengano richiesti in tempi brevi, o addirittura in tempo reale, anche a costo di una minor precisione nei risultati, ben si comprende come lo sviluppo di algoritmi adatti allo scopo diventi un obiettivo fondamentale di ricerca.

Sulla base di tali considerazioni, una nuova tendenza nei metodi di analisi video è apparsa ultimamente nel panorama delle metodologie di indicizzazione e *retrieval*. Tale trend, che prende il nome di *"rough indexing"* prevede di lavorare con dati solo parzialmente decodificati estratti direttamente dal flusso MPEG, e quindi a risoluzione ridotta. Questo metodo è già stato utilizzato per la realizzazione di algoritmi per individuare gli *shot-cut* in un video, per stimare il moto di camera e per estrarre gli oggetti in primo piano, sempre a partire da dati nel dominio compresso.

In questo contesto di indexing veloce e *"rough"* si inquadra anche questo lavoro di tesi nel quale solo dei dati estratti dal flusso compresso (cioè vettori di moto ed immagini a risoluzione spaziale ridotta) vengono utilizzati al fine di identificare ed estrarre gli oggetti in primo piano. In accordo al paradigma di *"rough indexing"*, le informazioni di moto vengono combinate con quelle ottenute mediante una segmentazione sul colore, per identificare gli oggetti in primo piano di una sequenza video con moto di camera e degli oggetti arbitrario. Il primo passo del metodo proposto consiste nell'estrarre dai frame di tipo P, utilizzando un algoritmo di stima del moto di camera molto robusto, le regioni che non seguono il moto di camera stimato. A queste regioni convenzionalmente si assegna il nome di *"object masks"* poiché è legittimo supporre che gli oggetti in primo piano si trovino con maggior probabilità al loro interno. Successivamente, per raffinare le *"object masks"* che altrimenti risulterebbero troppo approssimative essendo molto sensibili al rumore che i vettori di moto presentano per loro stessa natura, viene realizzato un algoritmo di segmentazione in regioni sulla base del colore sui frame di tipo I.

Come tutti i metodi basati su informazioni di moto noti in letteratura, dal momento che le *"object masks"* sono ricavate proprio a partire da questo tipo di informazione, ne risulta che se il moto dell'oggetto in primo piano non si distingue sensibilmente da quello della camera o se è fermo, nessun oggetto può essere identificato. Tuttavia dal momento che possiamo supporre che tale oggetto non possa apparire e scomparire lungo una sequenza di frame, proponiamo di filtrare la sequenza degli oggetti trovati lungo l'asse temporale, in modo da poter approssimativamente ricavare la posizione e la forma dell'oggetto qualora l'informazione di moto non fosse sufficiente. L'obiettivo di questo filtraggio spazio-temporale, che costituisce una assoluta novità rispetto a quanto proposto in letteratura, è di ricavare una traiettoria approssimativa del baricentro dell'oggetto a partire dai frame nei quali è stato identificato e quindi di costruire una sorta di tubo che ne segua la traiettoria e la cui sezione in ogni istante di tempo dia l'approssimazione della posizione dell'oggetto nell' istante stesso.

Il metodo proposto fornisce risultati molto promettenti: infatti l'obiettivo di identificare gli oggetti in primo piano viene raggiunto nella maggioranza dei casi. Sebbene l'approccio a risoluzione ridotta utilizzato permetta da una parte di conseguire risultati in tempo reale, dall'altra è responsabile di una certa imprecisione sui bordi degli oggetti estratti. Gli episodi di *miss-detection* e *over-detection* inoltre hanno potuto essere corretti mediante il filtraggio spazio-temporale da noi proposto.

Contents

Table of contents **xi**

Introduction **1**

1 An overview of the MPEG standard family **7**
1.1 Introduction 7
1.2 MPEG2 standard for video coding 8
1.2.1 Video signal properties 8
1.2.2 Coding methods 9
1.2.3 Different levels of a video sequence 11
1.2.4 Spatial redundancy 12
1.2.5 Statistical redundancy 15
1.2.6 Temporal redundancy 16
1.2.7 Some conclusions about MPEG2 video part 18
1.3 MPEG4 Visual part 1 18
1.3.1 Overview of MPEG4 visual 19
1.3.2 MPEG4 visual properties 19
1.3.3 Coding arbitrary-shaped regions 20
1.3.4 Some conclusions on MPEG4 Visual 23
1.4 MPEG7 standard 23
1.4.1 MPEG7 tools 24
1.4.2 Overview of MPEG7 visual descriptors 25
1.5 Conclusions 29

2 Object-based segmentation and indexing of video streams **31**
2.1 Introduction 31
2.2 Intraframe and motion segmentation 32
2.3 Combined approaches to object segmentation 35

2.4 Semi-automatic approaches 43
2.5 Spatio-temporal video object segmentation 46
2.6 Spatio-temporal object segmentation in compressed domain 47
2.7 Real-time segmentation of video object 51
2.8 Conclusions 58

3 Object-based segmentation of video streams for 'Rough Indexing' 59
3.1 Introduction 59
3.2 Rough indexing paradigm 60
3.3 Methodology for foreground object extraction 60
3.4 Motion mask extraction 62
3.4.1 Camera motion estimation 62
3.5 Motion mask extraction from a single P frame 67
3.5.1 Filtering of outliers due to camera motion 71
3.5.2 3D filtering of the motion masks 73
3.6 Motion mask extraction in I frame 76
3.7 Object mask refinement by color segmentation 77
3.7.1 Pre-processing filtering 78
3.7.2 Gradient extraction 80
3.7.3 Region growing algorithm 81
3.8 Merging of motion masks and color segmentation results 83
3.8.1 Flat region removal 84
3.9 Conclusion 85

4 Feature extraction and spatio-temporal filtering 87
4.1 Introduction 87
4.2 Extraction of object characteristics 87
4.3 Spatio-temporal filtering 91
4.3.1 Identification of objects at I frame resolution along the time . 92
4.3.2 Spatio-temporal tube construction 96
4.4 Conclusions 106

5 Results 109
5.1 Introduction 109
5.2 Results 109
5.2.1 Evaluation method 110
5.3 Analysis of a generic video sequence 114

5.3.1 "De l'arbre à l'ouvrage" 115
5.3.2 Generic video results 124
5.4 Cartoons 131
5.4.1 Ferrailles 131
5.4.2 Cartoon results 134
5.5 Conclusions 140

Conclusions **141**

References **145**

Introduction

The creation of large databases of audio-visual content in professional world and the extensively increasing use of home multimedia devices able to store hundreds of hours of multimedia content strongly require the development of new methods for processing and indexing multimedia documents. The main objective consists in extraction of meaningful information and organization of multimedia content.

New multimedia standards have been developed to produce efficient instruments able to compress (MPEG4) and index (MPEG7) huge amounts of data. In the case of MPEG4 new techniques have been proposed to semantically compress the content, and in particular to distribute compression differently between objects of interest, as foreground objects, and the less interesting parts of the scene. To this aim, according to this *content-based concept*, a scene is viewed as a composition of video objects with intrinsic spatial (shape and texture) features and motion behavior (trajectory).

The MPEG7 standard, instead, is not a compression standard, but it proposes many detailed schemes for multimedia content indexing and description, enabling the description of specific objects inside visual scenes. Although, the development of algorithms to provide object-based information is out of the scope of the standard , so methods able to identify meaningful objects are left to the content developer.

A variety of methods has recently been developed to fulfill the objective of detecting foreground objects or meaningful components in a video scene, but the problem of object extraction from raw or compressed video still remains a challenge. The objective of this thesis work is to develop an efficient method to automatically extract in real-time the foreground objects and some of their characteristics.

It is still not possible to have an accurate segmentation of the object boundaries at a low computational cost. For this reason on one side some methods pursue the objective of an accurate object segmentation and on the other side other algorithms try to obtain a more approximate segmentation, but in real-time. It is clear that these conflicting requirements for the precision and complexity of object extraction

are very much application dependent, so the problem is far from being solved.

First approaches from an historical point of view where directed either towards spatial segmentation or towards motion segmentation but, since both approaches have their own difficulties, most of the video object segmentation tools integrate both spatial and temporal segmentation techniques.

Moreover, most of these combined approaches concentrate on segmentation in the pixel domain, suffering of high computational complexity and requiring that the sequence is fully decoded before processing. To tackle these drawbacks of pixel-domain approaches, a few compressed domain methods have been proposed for spatio-temporal segmentation. These approaches, although significantly faster than pixel-domain algorithms, cannot however be executed in real-time on today computers.

So now efforts are oriented in two different directions: the first one is devoted to obtain foreground object extraction with the objective of MPEG4 coding; even if performed in compressed domain, these algorithms need a refinement phase after object extraction and the coded stream must be at least partially decoded at pixel resolution. The second type of algorithms is instead devoted to semantic indexing of multimedia content. For this kind of approach it can be useful to obtain the results in real-time since this kind of approaches is often projected towards user-dedicated applications. It becomes important to extract object information such as its shape, its dimensions, its trajectory, and all the characteristics useful to describe the objects themselves. In this case a lower precision in object boundaries can be tolerated. This thesis work belongs to the second set of approaches. In fact our objective is to extract in real-time from an MPEG2 video sequence the foreground objects present and to obtain some useful information about them. In order to obtain results in real-time, we decide to sacrifice perfect object boundary detection and to work at low resolution.

For this purpose in this work we define a new paradigm of *rough indexing* which means fast and approximate analysis of multimedia content at poor resolution. So our algorithm is developed using motion and color information directly extracted from the compressed stream, that is MPEG2 motion vectors and DC coefficients of DCT transform.

The work is mainly organized in two parts: the first part concerns the extraction of the foreground objects in each I frame of each GOP of the analyzed sequence; as this extraction is performed for each GOP independently from the others, if an error occurs, and this is quite easy due to the rough data processing, this error cannot

be corrected. In literature no solution is proposed to solve this problem. For this reason we have developed a correction procedure, that is a spatio-temporal filtering that takes into account the results along the time to correct some detection errors.

For the first part, that is the object extraction part, we have adopted a combined motion and color based segmentation approach: first we estimate the camera motion using a robust estimator, then we extract all the outlier macro-blocks and we filter them to separate the outliers due to the noise from the ones due to the presence of a foreground objects and that are part of the so called *motion masks*. After this filtering we obtain the object motion masks for P and I frames.

Once the motion masks for the I frames have been extracted, color information is used to refine the object masks. In the rough indexing context, only low resolution I frames are taken into account. The objective is to segment the DC color image and to overlap it to the object motion masks to finally extract the foreground objects. As for the macro-block resolution, it is difficult to obtain a good segmentation due to possible high gradient values between adjacent macro-blocks; thus, a pre-filtering process is necessary. In this case the use of morphological operators can eliminate local gradient peaks without modifying the object boundaries. Such morphological processing flattens the interior part of objects without changing their boundaries. Once the images have been filtered, a modified watershed algorithm is performed on I frames to subdivide them into homogeneous color regions.

The results of color segmentation and motion analysis, if taken separately, are not sufficient to detect foreground objects, because motion masks may be too approximative and color segmentation cannot discriminate background from foreground, but their combination can offer very good results. The outcome of this first part are the foreground objects present for each I frame in real-time. As this process is performed separately for each GOP, if an error occurs in the process, even if it will not influence the process for the following GOP, the result cannot be corrected.

So the second part of the thesis work is devoted to correct the errors that can affect the first part by implementing a spatio-temporal filter that takes into account the global results along the sequence. Obviously, this kind of filtering has to be performed once the object have been extracted, so it can not take place in real-time.

To filter the miss-detections or the cases of false alarm we propose to follow the object with a sort of tube which can be built using simple quadric functions such as cones or cylinders. The idea is to use the objects that have been correctly detected in many frames to correct the cases when the same object, due to an error, is not or has only been partially detected, following the principle that an object cannot

suddenly disappear or instantaneously change its size and shape.

If more than one object is present for each frame it is important to establish a correspondence between the different objects at different moments of time. To do that, we estimate the motion of each foreground object and we project it on the subsequent frames; the projection overlaps with detected object of such frames so that a correct correspondence can be established. Then, the trajectory of each object is computed by approximating with a straight line the line of the centers of mass of the detected objects.

This straight line, that represent the trajectory of the object along the sequence, is also considered to represent the main axes around which the quadric function is built. In each frame with a detected object, a set of concentric ellipses centered in the center of mass is built. Then starting from these ellipses a quadric function that approximate the different ellipses in the different images is computed. In the frames where no objects can be detected the section of the quadric offers the elliptic approximation of the object.

Concluding, this algorithm helps to extract foreground objects from an MPEG2 video sequence, mainly executed in real-time thanks to rough data and low resolution processing.

In the next the detailed description of how this work is organized is given:

Chapter 1 - An overview of the MPEG standard family. Since the method of object extraction is performed in the compressed domain using color and motion information from MPEG2 compressed stream, in order to understand the nature of this data, the MPEG2 standard is first introduced. Then, MPEG4 basic principles are described and in particular its object oriented compression part. Finally, as the context of this work is an object-based indexing of compressed streams, the MPEG7 standard is discussed. The standard is not analyzed in all its details, the presentation focusing mainly on shape and motion descriptors.

Chapter 2 - Object-based segmentation and indexing of video streams. In the case of MPEG4, the approach taken for coding the video sequences may rely on an object-based representation of the scenes and on a separation of foreground from the background. The MPEG7 standard is also facing the same kind of challenge since indexing and retrieval may require to structure the data in terms of objects, regions and associated semantics. So, in this chapter, different approaches to the problem are presented and different algorithms are discussed in details. We start from traditional techniques of image segmentation; then we present combined approaches until the most recent techniques realized in compressed domain including those attempts to

achieve the results in real-time.

Chapter 3 - Object-based segmentation of video streams for 'Rough Indexing' In this chapter we propose our object extraction method and in particular we describe in detail the object extraction part, based on joint motion and color segmentation applied at low resolution and on rough data in a 'rough indexing' context: first we use motion information extracted from MPEG compressed stream to estimate camera motion and to detect those regions that do not follow the camera movement and we call them motion masks. Then, from the compressed stream, we extract DC resolution color information and we perform color segmentation. Then motion masks and color segmentation results can be merged together to obtain more accurate foreground objects.

Chapter 4 - Feature extraction and spatio-temporal filtering. Sometimes the algorithm at its first stage may produce some detection errors on isolated frames. These kinds of error, specially in the miss case, can be avoided by taking into account the results of the neighboring frames with a sort of filtering of the results along time. In this chapter we are going to explain how we extract the trajectory of the foreground objects and we realize our spatio-temporal filtering, in which the object sequence along the time is approximated with a sort of 'tube' obtained by a quadric function. The sections of this quadric can indicate the location of the missing object and its approximate shape.

Chapter 5 - Results. In this chapter we present the performance of our method on all results obtained for the analyzed video sequences. We also present some video sequence in details, showing the intermediate steps.

Chapter 1

An overview of the MPEG standard family

1.1 Introduction

As we stated in the Introduction, in this work we address the problem of foreground object extraction and modelling in compressed domain. The partially decoded stream represents the "rough" data for our object-extraction method. Therefore, in order to understand the nature of this data, we will have to introduce actual standards the most common for video broadcast and storage (MPEG1,2) we will use.

We are going to introduce such standards [1] since color and motion information extracted from the compressed stream will be used in the work. MPEG1 is a reduced version of MPEG2 and there are not many difference between the two data streams, so only MPEG2 will be treated in this chapter but the same items can be formed for MPEG1 streams.

The final goal of this work is to supply a powerful tool for object-based video indexing. The method will be based on approaches extensively developed since last 10 years for object-based video segmentation in a coding process. Therefore, it seems natural to present MPEG4 which has been a stimulator for the research, as it proposes object-based coding concept [2].

Finally, as the context of this work is an object-based indexing of compressed streams, then we will have to introduce MPEG7 [3]. MPEG7 standard supplies a normalized multimedia content description interface. MPEG7 deals with objects but with a description aim, that is to say that MPEG7 standardizes among other things a way to describe the objects in video sequences. However, in both standards, no

indication is given on the method that has to be used to extract such objects. The result is that so many independent works have addressed the issue, but the problem has not been solved yet.

1.2 MPEG2 standard for video coding

Video compression is considered a very important task and has been studied since the beginning of the '50s. However, only recently, the image quality - compression ratio trade-off has increased significantly, thanks to new coding techniques and to the better computer computational power. The MPEG[1] standard has been developed since the end of the '80 with the objective of defining a video coding standard for digital supports. As an outcome, in 1992 the MPEG1 standard was first introduced. The characteristics of this standard is a reduced spatial resolution (320×240) of progressive sequences with typically a bit rate of 1,5Mb/s that is a compression rate of 100:1.

Just after the MPEG1 standardization, it was clear that there was the need of a second standard providing better performance for progressive and interlaced sequences. So, at the beginning of 1992, the work on MPEG2 standards for video signal compression began.

1.2.1 Video signal properties

An analogue video signal is characterized by 25 frames per second, 625 lines per frame and 720 pixels per line. The digitization of an analogue signal is obtained by sampling each image and quantizing the values of the samples. The image is then reduced to a matrix of pixels, described by their color components values. For each pixel three components representing their visual primitive, typically Red(R), Green(G) and Blue(B), are estimated. But, since many research works have shown that the human eye is less sensitive to color components than to luminance components, the RGB color space is transformed typically in a YUV space representation. In this color space, Y represents the luminance component, while U and V represent chrominance components.

The digital equivalent of YUV is represented by Y, Cb and Cr, in which the two chrominance difference components Cb and Cr are less correlated than U and V for digital ITU-R 601 sequences and so they can be coded separately. Moreover, as

[1] Motion Picture Expert Group

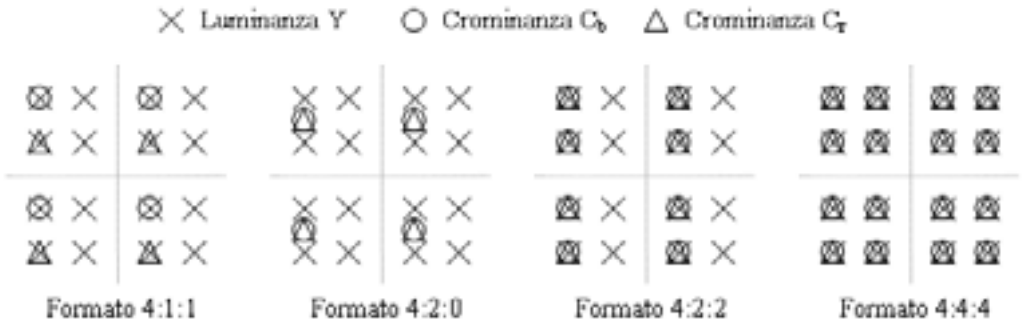


Figure 1.1: Different type of ITU-R video sampling formats.

the human eye is less sensitive to these components, they can be subsampled. The different types of subsampling employed are (see also Figure 1.1):

- **4:4:4** - this represents the canonical sampling, that is each pixel is obtained by sampling at the same rate the three components Y, Cb, and Cr;
- **4:2:2** - Cb and Cr components are subsampled by 2 with respect to 4:4:4 in the horizontal direction;
- **4:1:1** - Cb and Cr components are subsampled by 2 in both direction;
- **4:2:0** - Cb and Cr components are subsampled in both directions but the location of the Cb and Cr components is not the same of the Y component in the image plane.

1.2.2 Coding methods

The number of necessary bits to describe the information of a video sequence can be reduced by eliminating the information redundancy. MPEG2 standardizes various methods and tools to reduce such redundancy.

In a video sequence we can find different types of redundancies: *spatial*, *temporal* and *statistical*. Spatial and temporal redundancies are a consequence of the fact that pixels in a sequence are not independent from their neighbors, there a is correlation between closer ones both in space and time, so their value can be estimated by prediction.

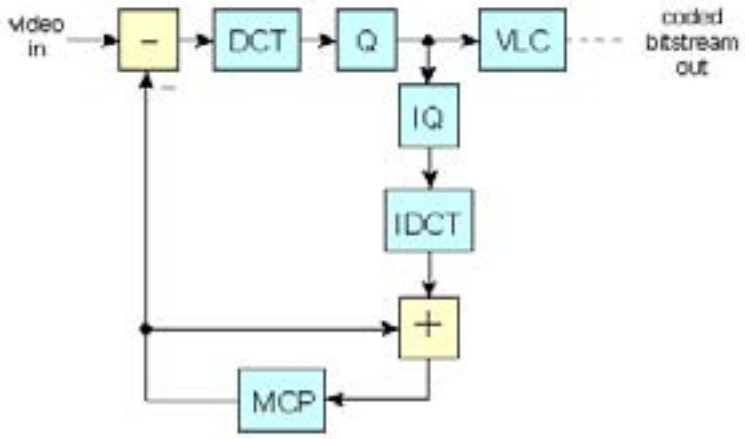


Figure 1.2: Motion-compensated DCT coder.

Reducing the *temporal redundancy* means coding the images using, as a reference image, one taken from the previous or future images. Then the redundancy is reduced by employing some motion estimation first, then using it for motion compensation techniques; this form of prediction enables the so called *Interframe coding* used in MPEG.

For *spatial redundancy* the image is considered as a static one and not as a part of a video sequence and it is coded independently from the others thanks to transform techniques such as the DCT (Discrete Cosine Transform) followed by quantization; this coding type is called *Intraframe coding*.

To eliminate the *statistical redundancy* tipycally *entropyc coding* is used.

In addition, *psychovisual irrelevance* present in human eye perception, can be used so as to mash for example some coding artifacts that might occur close to edge object boundaries present in the frame.

In Figure 1.2 an MPEG2 coder is shown. The coder subtracts the motion-compensated prediction from the source picture to form a 'prediction error' picture. The prediction error is transformed with the DCT, the coefficients are quantized and these quantized values coded using a VLC. The coded luminance and chrominance prediction error is combined with 'side information' required by the decoder, such as motion vectors and synchronising information, and formed into a bitstream for transmission. The single blocks of the proposed scheme will be explained in details in the next paragraph.

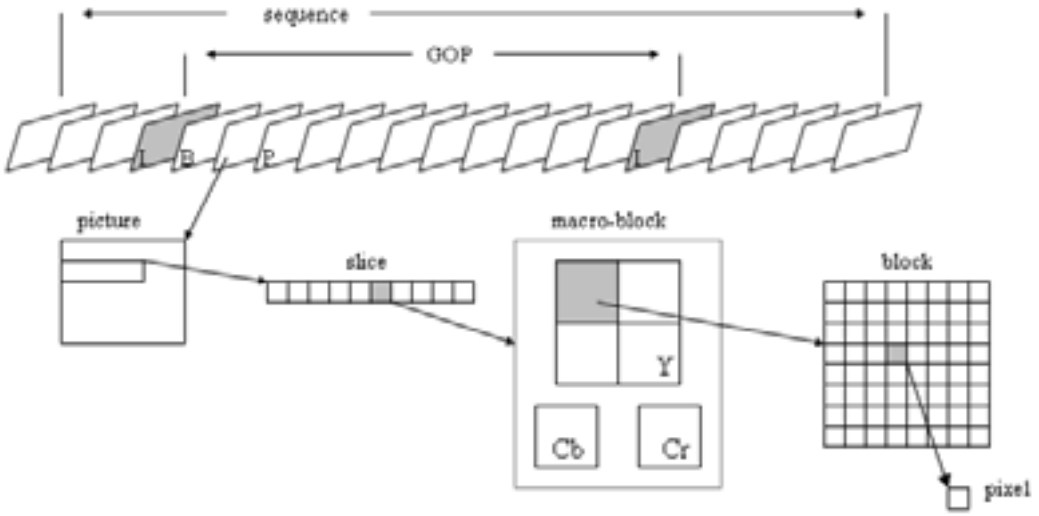


Figure 1.3: Level subdivision from video sequence to single pixel.

1.2.3 Different levels of a video sequence

To simplify the understanding of the coding process, transmission and decoding, we can decompose the video sequence at different levels. Going from the finest to the largest one we define: *Blocks*, *Macro-Blocks*, *Slices*, *Pictures*, *GOP (Group of Pictures)* and *Sequences* (see fig. 1.3).

- A **Block** is a section of the image composed by 8×8 pixels; it can be a block of luminance values or chrominance values and it is the smallest unit that is coded using DCT Transform;
- A **Macro-Block** contains, if we suppose for example a 4:2:0 subsampling pattern, a section of luminance with dimensions 16×16 pixels and two chrominance sections of 8×8 pixels. The macro-block is the basic unit taken into account for example in motion compensation and it is composed by 4 blocks of luminance and 2 blocks of chrominance;
- A **Slice** is a stripe composed by adjacent macro-blocks built from left to right then from top to bottom. The coding of a slice can be made completely independent from the adjacent slices. It is also used as synchronizing unit;
- A **Picture**, called also **Frame**, is composed in MPEG2 of 720×576 pixels and

is classified into three different types: *Intra (I)*, *Predicted (P)* and *Interpolated (B)*:

  - **I frames** - The *Intra* type images are coded as static images, using only a spatial DCT transform;
  - **P frames** - The *Predicted* images are coded starting from the previous I or P frame using a predictive technique (motion compensated or not);
  - **B frames** - The *Interpolated* images are coded with reference to the previous or following frame or both of them, using a predictive technique based on up to two reference frames, one in the past and the other in the future. A B frame cannot serve as a reference frame by other P or B frames;

- A **GOP**(Group of Picture) is formed by a fixed number of pictures, usually 12, 15 or 18, and it is the minimum unit that specifies the temporal prediction structure. A GOP must always start with an I frame; it can contain only one I frame and it ends with the last frame before the next I frame. The GOP can be of two types: an *open* GOP uses the first frame of the next GOP for coding the last frame; a *closed* GOP instead has no predictive links with the next GOP. This type of GOP must necessarily end with a P frame. In Figure 1.4 an open GOP with 12 frames is presented. In this GOP three P frames and eight B frames are present: each P frame is coded taking the previous I or P frame as a reference, while the B frames present a bidirectional prediction with reference to the previous and the following I or P frame.

- A **Sequence** is a series of GOPs.

1.2.4 Spatial redundancy

The first step in many compression systems consists in identifying spatial redundancy in video pictures using DCT transform and then quantizing the result. This coding technique is applied with different characteristics both on macro-blocks of intraframe coded images and on macro-blocks of prediction errors in interframe coding process.

Discrete Cosine Transform

The DCT transform is a mathematically lossless process that is able to pass from the pixel domain of an image to a frequency domain representation. The advantage

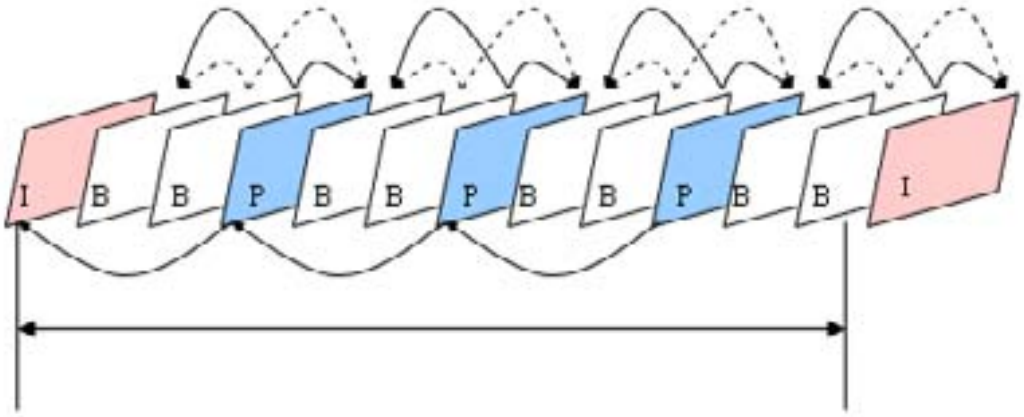


Figure 1.4: An example of an *open* GOP.

of the DCT transform is that it can be implemented in an easy way with a simple algorithm that can be exactly inverted. The two-dimensional DCT transform is the following:

$$B(k_1, k_2) = \sum_{i=0}^{N_1-1} \sum_{j=0}^{N_2-1} 4A(i,j) \cos\left[\frac{\pi k_1}{2N_1}(2i+1)\right] \cos\left[\frac{\pi k_2}{2N_2}(2j+1)\right] \tag{1.1}$$

where N_1 and N_2 are the image dimensions, $A(i,j)$ is the intensity value of the pixel (i,j) and $B(k_1,k_2)$ is the resulting coefficient.

The first coefficient of the DCT is called the *DC value* of the block and it is proportional to the average amplitude value of the different components Y, Cb and Cr within a block. The principal characteristic of this transform is that it often concentrates the energy of the block in a small number of coefficients at lower frequencies, given the typical status of static images: this means that we have the possibility to reduce the amount of data necessary to represent an image. In fact, as we know that the first coefficients concentrate the energy of the signal, most coefficients will tend to be zero-valued which eases the subsequent entropy coding work.

Quantization

The quantization process consists in a discretization of the spatial frequency coefficients in image blocks after they have been transformed by DCT. This leads to good

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

(a) non linear quantization matrix

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

(b) linear quantization matrix

Figure 1.5: The quantization matrix used for intraframe and interframe coding.

compression with small loss of information.

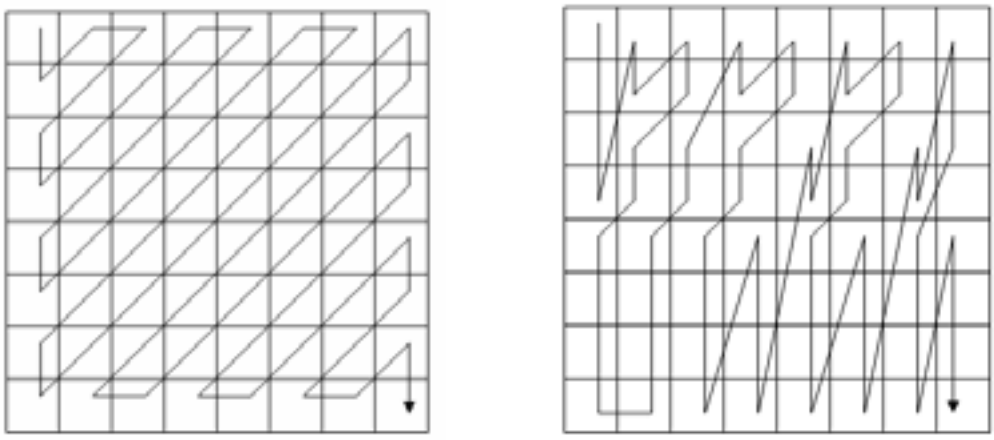
Since the human eye is very sensitive to luminance change the DC value is the coefficient that is coded with the highest precision (even 11 bits). On the other side, since the human eye does not perceive well the high frequencies, very few bits can be used to represent them. Moreover, since the DCT typically leads to very low values at high frequencies, the quantization results in a long set of zeros so that very few bits are needed for their representation.

Depending on the target bit-rate and perceptual significance of a macro-block, a more efficient quantization process is obtained by specifying different quantization step sizes for each macro-block or group of macro-blocks.

In Figure 1.5 two different types of quantization matrixes are shown. The non constant coefficient matrix is used for intraframe coding, because the DCT transform of the signal presents high energy values in the low frequency coefficients (the top corner at the left). The constant coefficient matrix is used instead for interframe quantization; in fact, the DCT transform of the prediction error presents the same distribution of values at all frequencies. For this reason the quantization will be done using a constant coefficient matrix.

Assume that F is an unquantized DCT coefficient, QF is a quantized DCT coefficient, W is the step size and S is the scale factor. For AC coefficients of intra coded blocks, $QF = [(32 \times F) + (\text{sign}(F) \times W \times S)]/(2 \times Q \times W)$.

For all coefficients of non intra coded blocks, $QF = (32 \times F)/(2 \times S \times W)$.



(a) (b)

Figure 1.6: Two different types of coefficient reading: a) zigzag and b) alternate.

This results in a larger encoded dead zone around 0 for the non intra case. For DC coefficients of intra coded blocks, $QF = F/k$ where k is a constant determined by the selected DCT precision.

Coefficient coding

The coding order of DCT quantized coefficients is chosen to optimize the coding process efficiency. The process starts from the DC coefficient and reads all the coefficients that form an 8×8 matrix to place them in a one-dimensional vector. These coefficients are not read line by line but following a path that maximizes the consecutive number of zeros.

In Figure 1.6 two types of coefficient readings are shown; both are able to maximize the number of consecutive zeros and so they are more efficient than a raster scanning which reads the coefficients from left to right and from top to bottom.

1.2.5 Statistical redundancy

Run-Length coding

Once the DCT quantized coefficients have been placed in a one-dimensional vector, instead of coding each value alone, consecutive values with the same amplitude are coded together by defining the set of couples of values (N,M) where N represents the number of repeats of a same value M. Some couples will have N set to one. At high

frequencies, where long sequence of zeros occur, many coefficient can be combined to form a single couple, which improves on average the coding efficiency.

VLC coding

The Variable Length Coding algorithm, is an Entropy Code, which assigns short/long codewords to frequently/rare symbols (represented by the couples defined above), respectively.

1.2.6 Temporal redundancy

Analyzing a video sequence we can notice that the content does not change a lot between two adjacent frames; so the main idea to reduce temporal redundancy is to consider that the current image can be predicted by performing local translation in space of the past and future images. The considered motion is not equal for the entire image but can vary between two different regions. The temporal redundancy will be taken into account only for interframe coding, that is to say for P or B images.

The image has to be subdivided into blocks and the block dimension where the motion compensation has to be applied is an important choice. In fact using small blocks the estimation accuracy will grow, but in this case the computational cost and the number of motion vectors will grow as well.

MPEG2 standard uses macro-blocks of 16×16 pixels as fundamental blocks, even if it is possible to consider 16×8 pixel for interlaced sequences.

Motion compensation

The concept of *motion compensation* consists in building the current image using the macro-blocks of a past or of a future image by displacing them on the basis of the motion vectors associated to them; these macro-blocks are then corrected to fully match the current frame.

To determine the motion vectors the algorithm attempts to displace same size reference pixel blocks so as to best match the current frame macro-block. The resulting displacement will be coded to form part of the bit stream. The prediction error obtained by the difference between the pixel values of the reference macro-blocks with respect to the current frame forms what is called the Displacement Frame Difference (DFD).

The next step involves the computation of the DCT of the DFD, its quantization

and its entropy coding; motion vectors differences with respect to previous macro-block motion vectors are instead entropy coded and transmitted to the decoder.

Thus if we denote $I(x, y, t)$ the original image signal at time t, $(dx, dy)^T$ the motion vector associated to the the macro-block, then the coded and decoded version $\hat{I}(x, y, t)$ is computed as:

$$\hat{I}(x, y, t) = \hat{I}(x + dx, y + dy, t - dt) + \hat{e}(x, y, t) \tag{1.2}$$

Here $\hat{I}(x + dx, y + dy, t - dt)$ is the value of coded and decoded reference frame, $(dx, dy)^T$ is an estimated motion vector, $\hat{e}(x, y, t)$ is a coded and decoded error signal. Its original version $e(x, y, t)$ represents a difference between original frame $I(x, y, t)$ in the encoder and coded and decoded compensated reference frame:

$$e(x, y, t) = I(x, y, t) - \hat{I}(x + dx, y + dy, t - dt) \tag{1.3}$$

In MPEG1,2 the error signal $e(x, y, t)$ is DCT transformed and quantized, thus $\hat{e}(x, y, t)$ is obtained by inverse DCT.

Block matching

In the *block matching* technique the macro-block to be estimated is compared with others blocks with same dimensions placed inside a research area in a past or future frame; the estimation can be obtained by searching for the motion vector that minimizes a cost function of the prediction error.

It is possible to estimate for each macro-block the prediction error variance: if such value is lower than the one of the original image the macro-block is coded in interframe mode, otherwise is coded as an intraframe macro-block; in the same P or B frame it is possible to code separately macro-blocks as interframe or intraframe.

To give an example of motion compensation with a P frame we suppose to subdivide the current frame I_n into blocks and we search each block in the past frame I_{n-1}. In this way it is possible to find a motion vector (dx, dy) for the previous frame that represents the block movement. The estimation $\tilde{I}_n$ of the current frame will be obtained from the previous one translating the blocks according to their own motion vectors; then the frame I_n is recovered from the previous one using the motion vectors and the prediction error.

For each macro-block of a B frame there are 4 different choices, (see 1.7):

- prediction with the previous frame as for P frames;

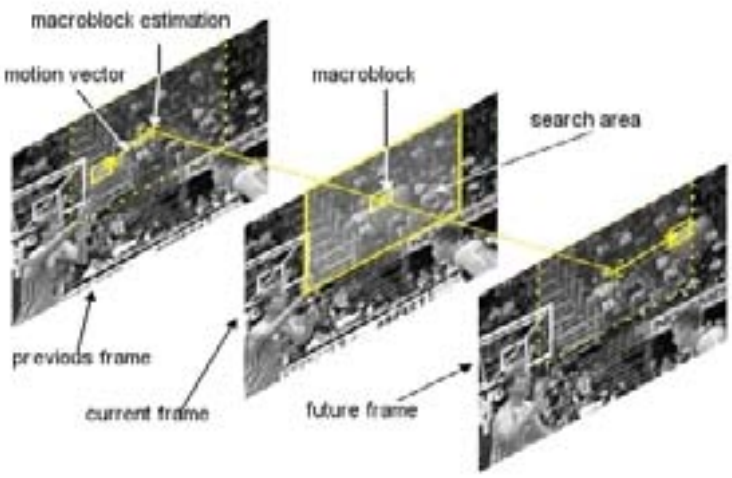


Figure 1.7: Motion estimation with previous and future prediction.

- prediction with the future frame;
- interpolation using previous and future frames;
- the intraframe coding.

1.2.7 Some conclusions about MPEG2 video part

In the sections above the different characteristics of the MPEG2 standard have being presented. The scope of the standard is to remove the redundancy present in a video sequence and take into account the human visual system perception to best reduce the effects of quantization. We will try to use MPEG2 compressed information to speed up the algorithm implementation of the proposed method. For example MPEG2 motion vectors will be used as well as the DC coefficients of the DCT transform of intraframe.

1.3 MPEG4 Visual part 1

MPEG4 Visual [4] improves the MPEG2 standard both in terms of compression efficiency and flexibility, enabling a wider range of applications, by making use of more advanced compression algorithms and by providing an extensive set of “tools”

for coding and manipulating digital media. It supports enhanced compression efficiency, reliable transmission, coding of separate shapes or '*objects*' in a visual scene, mesh-based compression and standardize means of facial or body animation.

1.3.1 Overview of MPEG4 visual

MPEG4 Visual provides a toolkit of coding techniques and resources making it possible to deal with different types of visual data including rectangular frames (such as the 'traditional' video material), video objects of irregular shape, still images or synthetic visual information (computer generated scenes). MPEG4 Visual provides its functionalities through a set of tools, organized into 'profiles', suitable for certain applications.

MPEG4 aims at ensuring that compliant encoders and decoders can interwork with one another, giving the developers the freedom to optimize the encoders as long as they generate a syntax compliant with the standard specifications.

1.3.2 MPEG4 visual properties

MPEG4 attempts to satisfy the requirements of a wide range of visual communication applications through a toolkit-based approach to coding of visual information. Some of the key features that distinguish MPEG4 from previous visual coding standard include for example:

- Efficient compression of progressive and interlaced video sequences. Additional tools are also present to further improve compression efficiency;
- Coding of video objects, that is irregular-shaped regions of a video scene. This is a new concept for standard-based video coding and enables for examples independent coding of foreground and background objects in a video scene;
- Support for effective transmission over practical networks;
- Coding of animated visual object such as 2D or 3D polygonal meshes, animated faces and animated human bodies.

MPEG4 Visual provides its coding functions through a combination of *tools*, *objects* and *profiles*. A *tool* is a subset of coding functions to support a specific feature. An *object* is a video element that is coded using one or more tools; an object can be a sequence of rectangular frames, a sequence of arbitrary-shape regions or

simply a still image. A *profile* is a set of object types that a codec is expected to be capable of handling. The different profiles can handle low-complexity coding of rectangular frames, as does the *Simple Profile*, or object-based coding of high quality video with improved compression efficiency as in the case of the *Core Studio Profile*.

Profiles are an important mechanism for encouraging interoperability between codecs from different manufacturers. The MPEG4 standard describes a wide range of coding tools and it is unlikely that any commercial codec would require the implementation of all the tools. Instead, a codec designer chooses a profile that contains adequate tools for a target application.

One of the contributions of MPEG4 Visual is that it moves from the traditional view of a video sequence as a merely collection of frames to a collection of one or more *video objects*. A *video object* is defined as "a flexible entity that a user is allowed to access and manipulate" [4]. A video object (VO) is an area of the video sequence that may occupy an arbitrarily shaped region and may exist for an arbitrary length of time. It may also vary in size and shape over time. A VO considered at a particular moment of time is called *video object plane* (VOP). In the traditional approach of video coding instead, each VOP is a single frame while a sequence of frame forms a VO. This traditional approach allows less flexible options in video coding. So, for example, a video scene may be made up of a background object and a number of separate foreground object. In Figure 1.8 two different types of video objects are presented. In a) the video object is a sequence of frame and the video object planes are the single frames (as in the traditional approach of MPEG1 and 2); in b) instead the approach of MPEG4 visual: the VO is a sequence of arbitrary shaped VOPs. The approach offered by MPEG4 Visual to encode this kind of video scene is much more flexible than the rectangular frame structure proposed by earlier standard. Moreover the different objects may be coded with different visual qualities and temporal resolution to underline their importance in the final scene.

As this thesis work is not concerned with MPEG4 coding tools, the different coding profiles will not be treated in details, but we would like to describe arbitrary-shaped object coding because objects and in particular foreground objects are one of the key features of this thesis work.

1.3.3 Coding arbitrary-shaped regions

Coding objects of arbitrary shaped requires a number of extensions to the block-based core CODECs [5]. Each VOP is coded using motion compensated prediction and DCT-based coding of the residual (see section 1.2.6). Extensions to deal with

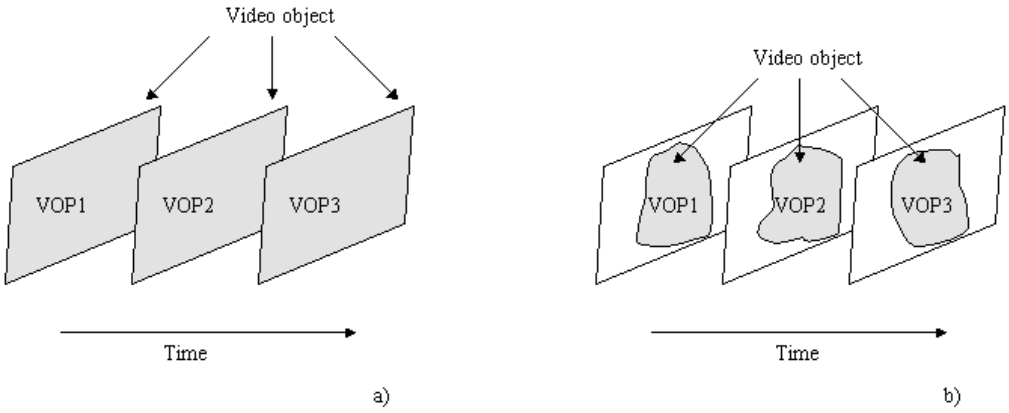


Figure 1.8: a) rectangular VOP and VO and b) arbitrary shaped VOP and VO.

the special case of object boundaries are needed, because it is necessary to cope with shape coding, motion compensation and texture coding of arbitrary shaped video objects.

Shape Coding

The shape of a video object is defined by *Alpha Blocks*, each covering a 16×16 pixel area of the scene. Each Alpha Block may be external to the object and in this case nothing has to be coded; it can be totally internal and in this case it is encoded as in *Simple Profile* (that is motion compensation and DCT of the residual) or it may include the object boundary. In this last case it is necessary to define the shape of the VO edge within the alpha block. The shape is defined using the concept of *transparency*, where the 'transparent' pixel is not part of the current VOP, the 'opaque' pixel is part of the VOP and replaces everything underneath it whereas a 'semi-transparent' pixel will be partially transparent. The shape information may be defined as *binary* (each pixel is either opaque, 1, or transparent, 0) or *grey scale* (each pixel is defined by a value between 0, transparent, and 255, opaque). The binary shape for a boundary macro-block is coded as a binary alpha block (BAB) using arithmetic coding, while grey scale shape information is coded using motion compensation and DCT coding.

Static sprite coding

Another important improvement in MPEG4 visual is the concept of *static sprite*. We can imagine, for example, a sequence of video surveillance. Clearly the background does not change along the sequence (the camera in fact is fixed). The background can be coded as a *static sprite*. A static sprite is treated as a texture image that may move or warp in certain limited ways, in order to compensate for camera movement such as pan, tilt, rotation and zoom. The sprite can also be much larger than the visible area in the scene and it may even have arbitrary shape. As the camera viewpoint changes, the encoder transmits some parameters indicating how the sprite should move and warp to recreate the appropriate visible area in the decoded scene. As the sequence progresses, the sprite is moved, rotated and warped to appropriately change the scene. There are two methods to transmit and manipulate sprites, *basic sprite* and *low-latency sprite*.

In the *basic sprite* the first VOP (I-VOP) contains the entire sprite, encoded in the same way as a normal I frame. The sprite can be larger than the visible display size to accommodate the camera movements. At the decoder the sprite is placed in a Sprite Buffer but it is not immediately displayed. All further VOPs are S-VOP. An S-VOP contains up to four parameters that are used to move and warp the content of the Sprite Buffer to obtain the desired background. A single parameter enables linear translation, two or three parameters enable affine transformation of the sprite and four parameters enable a perspective transform. In this case transmitting an entire sprite in the first VOP may introduce a significant latency because the sprite may be larger than the single VOP.

The *low-latency sprite* enables an encoder to send initially part of the sprite or a low-quality version of the entire sprite and then to update it during the following transmissions. The first VOP contains part of the sprite or all the sprite (at low resolution) with the dimension of the entire sprite. Each subsequent S-VOP may contain the warping parameters, as in basic sprite, and one or more 'pieces'. A sprite 'piece' covers a rectangular area of the sprite and contains macro-block data that construct part of the sprite that has not previously been decoded (*static-sprite-object* piece) or improves the quality of part of the existing sprite (*static-sprite-update* piece). In the first case the sprite macro-blocks are encoded as intra macro-blocks, while in the second case they are encoded as inter macro-blocks using forward prediction from the previous content of the sprite buffer.

1.3.4 Some conclusions on MPEG4 Visual

As it has been explained in the sections above, MPEG4 defines in details how to encode a scene as foreground and background, and proposes different tools to deal with arbitrarily-shaped objects, but it assumes that the segmentation of the scene into objects has already been achieved. So the problem to segment the scene is left to the developers. This is is the reason why in these last years many developers tried to design new techniques for object extraction and scene segmentation.

1.4 MPEG7 standard

The MPEG7 project, formally called *Multimedia content description interface* [6], has the objective to specify a way to describe different types of multimedia contents, in order to facilitate an efficient retrieval and management of the relevant information [7]. Like the other members of the MPEG family, MPEG7 is a standard representation of multimedia information but it is quite different from its predecessors. In fact, while MPEG1, MPEG2 and MPEG4 reproduce the content of the multimedia documents, MPEG7 represents information about the content and the way to describe it. As ultimate objective, MPEG7 tries to create an environment where tools from different providers can work together in order to create an infrastructure for transparent management of multimedia content.

Some of the guiding principles of MPEG7 standard are the following [7]:

- *Wide application base*: MPEG7 must be applicable to any domain, real-time generated or not. The content may be stored and may be made available on-line, off-line or streamed;
- *Relation with content*: MPEG7 allows the creation of descriptions to be used stand-alone, multiplexed with the content itself or linked to one or more version of the content;
- *Wide array of data types*: MPEG7 considers a wide variety of data types such as video, speech, audio, image, synthetic images, etc.;
- *Media independence*: MPEG7 is applicable independently to the medium that carries the content (film, tape, CD, hard disk, etc.);
- *Object-based*: MPEG7 allows to describe object-based representations of the content, i.e. the content can be described as a composition of multimedia

objects while allowing to separately access the information related to each object;

- *Format independence*: MPEG7 should be applicable independently of data format type, analog, digital compressed or uncompressed. However there is a special relation between MPEG7 and MPEG4 [8] [9], since both are built using an object-based data model, they complement each other very well allowing very powerful applications to be created;

- *Abstraction level*: MPEG7 includes description capabilities from low-level features, often statistical feature that can be extracted automatically, to high-level features with semantic meaning that are often extracted manually or semi-automatically.

As for MPEG4, in order to allow industrial competition, MPEG7 standardizes only the minimum necessary and this implies that only the description format, syntax and semantics, and its decoding is standardized. Elements that are explicitly not standardized are algorithms for extraction and encoding. The specification of content analysis tools, automatic or semi-automatic, is out of the scope of the standard and is left the industry.

1.4.1 MPEG7 tools

A number of tools are needed to achieve the objectives of the standard. This tools are *descriptors*, *description schemes*, a *description definition language (DDL)* and some *System tools* [10].

A *Descriptor (D)* is a representation of a feature, where a feature is a distinctive characteristic of the audio-visual information; it defines a syntax and the semantics of a the feature representation. The value of a descriptor represent the evaluation of the corresponding feature and it is possible to have several descriptors representing the same feature (for example to address different functionalities). An example for a descriptor can be a time code for representing duration, a string code for representing a title or, in our case, it can be the number of the objects in foreground, their shape or their trajectories along the sequence.

A *Description Scheme (DS)* specifies the relationship between its components which can be both descriptors or other description schemes. It describes multimedia content in terms of structure and semantics. An example of a description scheme for

a movie, structured in scenes and shots, including textual descriptors at the scene level and color, motion and amplitude descriptors at the shot level.

The *Description Definition Language (DDL)* is a language to create new description schemes and descriptors. It also allows the extension and modification of new existing description schemes, and specifies the syntax of each existing one so that they can be instantiated correctly.

There are also some *System Tools* related to the synchronization, transportation and storage of the descriptions. All these forms are the so called *Normative Tools*. In fact, if these tools are implemented they have to be realized according to the standardized specification to guarantee interoperability.

A *description* consists of one or more description schemes which can be fully or partially instantiated, that means that the values may be set for all or just some of the descriptors [10].

1.4.2 Overview of MPEG7 visual descriptors

The main objective of MPEG7 visual descriptors is to provide a standard description of streamed or stored images to help users to access and identify images and videos. These descriptors describe basic visual content on the basis of visual information. For example, for videos, the content may be described by the shape of objects, object size, color, object movements and so on. The general visual descriptors are subdivided in four classes: *color* descriptors, *texture* descriptors, *shape* descriptors and *motion* descriptors.

Color is the most widely used visual feature because it is robust to viewing angle, translation and rotation of the regions of interest.

Texture refers to the visual patterns that may have property of homogeneity or not, caused by the presence of multiple colors or intensities in the image.

The shape of visual objects provides a powerful clue for similarity matching in retrieval algorithms; MPEG7 provides region and color descriptors suitable for a variety of applications such as for example queries on images with written characters, pre-segmented object contours, trademarks, etc.

All the features described above can be employed to index images or retrieve relevant objects in video sequences. The description of motion features in video sequences can provide even more powerful clues regarding its content. With respect to this feature class, the dominant characteristics are provided by camera motion and object motion descriptors. Now we are going to introduce in details shape descriptors and motion descriptors.

Shape descriptors

To cover the important domain of shape descriptors, MPEG7 provides *contour-based shape* and *region-based shape* tools.

The **region-based SD** expresses pixel distribution within a 2D object or region. Since it is based on both boundary and internal pixels, it can describe complex objects consisting of multiple disconnected regions as well as simple objects with or without holes. This descriptor works by decomposing the shape into a number of orthogonal 2D basis functions (complex-valued), defined by the Angular Radial Transform (ART). The normalized and quantized of coefficients are used to describe the shape.

The ART coefficients are defined by:

$$F_{nm} = \int_0^{2\pi} \int_0^1 V^*_{nm}(\rho, \theta) f(\rho, \theta) \rho \, d\rho \, d\theta \tag{1.4}$$

where F_{nm} is an ART coefficient of order n and m, $f(\rho, \theta)$ is an image function in polar coordinates and $V_{nm}(\rho, \theta)$ is the ART basis function that is separable along the angular and radial direction, that is:

$$V_{nm}(\rho, \theta) = A_m(\theta), R_n(\rho) \tag{1.5}$$

with

$$A_m(\theta) = \frac{1}{2\pi} \exp(jm\theta) \tag{1.6}$$

and

$$R_n(\rho) = \begin{cases} 1 & n = 0 \\ 2\cos(\pi n \rho) & n \neq 0 \end{cases} \tag{1.7}$$

The descriptor is then defined as a set of normalized magnitudes of complex ART coefficients.

The **contour SD** is based on the *Curvature Scale-Space (CSS)* representation of the contour. Objects for which characteristic shape features are contained in their contour are described efficiently by the contour SD.

The concept of CSS representation is based on an observation that when comparing shapes, humans tend to decompose shape into concave and convex sections. The CSS representation also decomposes the contour into convex and concave sections by determining the inflection points. This is done in a multiresolution fashion, where

the contour is analyzed at various scales, obtaining a smoothing process.

Extraction of the CSS representation involves calculation of the curvature of a contour, while progressively smoothing the contour. The curvature of a contour is defined as the derivative of the tangent angle to the contour. Let us assume that the contour C is represented by a parametric vector equation: $C(u) = [x(u), y(u)]$, where x, y are the coordinates of the pixel belonging to the contour and u is a parameter. The curvature can be calculate as:

$$k(u) = \dot{x}(u)\ddot{y}(u) - \ddot{x}(u)\dot{y}(u) \tag{1.8}$$

where $\dot{x}(u)$, $\ddot{x}(u)$ are the first- and second-order derivatives of the x with respect to the parameter u (and similarly $\dot{y}(u)$, $\ddot{y}(u)$). The smoothing is performed iteratively and after each iteration, the inflexion points are computed. The filtering process continues until a convex contour results, that is until there are no more inflexion points.

The contour-based approach is suitable for types of shape in characteristic features are conveyed in the contour. Thinking for example about a fork, the region-based approach tends to confuse it with other elongated objects that expand at one end, for example spoons, while the contour-based can correctly describe it. On the other end, region-based approach is used for example in case of objects with convex shape where contours carry very littler characteristic information or in case of shapes altered by cuts or holes.

Motion descriptors

The main aim of motion-based indexing is to capture essential motion characteristic into concise and effective descriptors. In this paragraph we will describe in details the four motion descriptors standardized by MPEG7: *Motion Activity*, *Camera Motion*, *Motion Trajectory* and *Parametric Motion*.

A human watching a video or animation perceives it as a slow sequence, fast-paced sequence, action sequence and so forth. The **Motion Activity** descriptor captures this intuitive notion of *intensity of action* or *pace of action* in a video segment. Motion Activity includes the following attributes:

- *intensity of activity*: this is expressed by an integer lying in the range [1-5]. A high value of intensity indicates high activity while a low value of intensity indicates low activity;
- *direction of activity*: it expresses the dominant direction of the activity if any;

- *spatial distribution of activity*: it indicates whether the activity is spread across many regions or restricted to one large region; it is an indication of the number and size of *active* regions in a frame;

- *temporal distribution of activity*: it expresses the variation of activity over the duration of the video segment or shot.

The **Camera Motion** descriptor supports all camera translations, rotations and changes of focal length together with the possible combination of these. The eight basic camera operations usually defined are panning (horizontal rotation), tracking (horizontal transverse movement), tilting (vertical rotation), booming (vertical transverse movement), zooming (change of focal length), dollying (translation along the optical axis), rolling (rotation around the optical axis) and finally absence of operation (fixed camera). The core of the description includes timing information (that is start time and duration), the camera motion type and the amount of motion.

The **Motion Trajectory** describes the displacement of object in time, an object being defined as any spatiotemporal region whose trajectory is relevant in the context in which it is used. The trajectory model is a first- or second-order piecewise approximation of the spatial positions of the representative point along time, for each spatial dimension:

- first-order approximation:

$$x(t) = x_i + v_i(t - t_i) \quad \text{with} \quad v_i = \frac{x_{i+1} - x_i}{t_{i+1} - t_i} \tag{1.9}$$

- second-order approximation:

$$x(t) = x_i + v_i(t - t_i) + \frac{1}{2}a_i(t - t_i)^2 \quad \text{with} \quad v_i = \frac{x_{i+1} - x_i}{t_{i+1} - t_i} - \frac{1}{2}a_i(t_{i+1} - t_i) \tag{1.10}$$

and similarly for the other dimensions y and z. With such a model, v_i and a_i do correspond to the object's speed and acceleration respectively, considered constant on $[t_i, t_{i+1}]$, x_i and x_{i+1} are the positions at times t_i and t_{i+1}.

On the basis of this model, the core of the description is a set of *key points* representing the successive spatiotemporal positions of the described object.

The **Parametric Motion** descriptor represents the motion and/or deformation of a region or image by following classical parametric motion models:

- translational (2 parameters):

$$\begin{aligned} v_x(x,y) &= a_1 \\ v_y(x,y) &= a_2 \end{aligned}$$

- rotation/scaling (4 parameters):

$$\begin{aligned} v_x(x,y) &= a_1 + a_3x + a_4y \\ v_y(x,y) &= a_2 - a_4x + a_3y \end{aligned}$$

- affine (6 parameters):

$$\begin{aligned} v_x(x,y) &= a_1 + a_3x + a_4y \\ v_y(x,y) &= a_2 + a_5x + a_6y \end{aligned}$$

- perspective (8 parameters):

$$\begin{aligned} v_x(x,y) &= (a_1 + a_3x + a_4y)/(1 + a_7x + a_8y) \\ v_y(x,y) &= (a_2 + a_5x + a_6y)/(1 + a_7x + a_8y) \end{aligned}$$

- quadratic (12 parameters):

$$\begin{aligned} v_x(x,y) &= a_1 + a_3x + a_4y + a_7xy + a_9x^2 + a_{10}y^2 \\ v_y(x,y) &= a_2 + a_5x + a_6y + a_8xy + a_{11}x^2 + a_{12}y^2 \end{aligned}$$

where $v_x(x,y)$, and $v_y(x,y)$ represent the x and y displacement components of the pixel at coordinates (x,y).

The core of this description specifies which motion model is used among the ones listed above, the time interval on which it is used, the information about the coordinate system and finally the value of each parameter a_i.

1.5 Conclusions

In conclusion, in this chapter we have introduced the MPEG family and in particularly, MPEG2, MPEG4 and MPEG7. We have presented the MPEG2 standard and discussed its characteristics since, from MPEG2 stream, "rough" data for our

object-extraction method will be extracted and for this reason it is important to understand the nature of this data.

Then we have presented MPEG4 and in particularly the object-based coding techniques proposed. These object-oriented coding concepts have stimulated the development of new methods to extract objects and segment video sequences.

Finally, as the context of this work is an object-based indexing of compressed streams, we have introduce MPEG7 standard which standardizes among other things a way to describe the objects in video sequences.

As a result of these standardizations, in the last decade, different methods have been proposed to efficiently separate a video sequence into independent objects; some of this techniques will be presented and discussed in the next chapter.

Chapter 2

Object-based segmentation and indexing of video streams

2.1 Introduction

The problem of objet extraction from video streams is essential both for coding and indexing purposes. The recent MPEG4 (see 1.3) and MPEG7 (see 1.4) standards propose a new approach to compress and describe multimedia data. In the case of MPEG4 the approach taken for coding the video sequences may rely on the content-based visual representation of the scenes. In this kind of approach, the concept of *object* may be essential. The MPEG7 standard is also facing the same kind of challenge, in fact indexing and retrieval requires to structure the data in terms of objects, regions and associated semantics. For this reason, different approaches are present in the literature to efficiently extract objects for MPEG4 coding or MPEG7 description.

However, extracting objects from digital video is still a challenging task among the video processing community. There is still no single and reliable technique for video object detection.

In this chapter different approaches to the problem will be presented, starting from traditional techniques of image segmentation, until the most recent techniques including those attempt to achieve the result in real time.

2.2 Intraframe and motion segmentation

Approaches to video object segmentation we can find in literature can be broadly classified as:

- intraframe segmentation;
- motion-based segmentation.

In the first family of approaches each frame of the video sequence is segmented independently into regions of homogeneous intensity or texture using traditional image segmentation techniques such as [11] [12] [13] and [14]. In [15], for example, a 2D k-means clustering is used using color information, and then appropriate luminance values are associated to these clusters using the k-means algorithm. In [16] artificial neural networks are used to merge homogeneous regions based on the information provided by luminance, chrominance differences, region proximity and size. These methods subdivide the image into homogeneous regions, but they can not detect objects since objects are often composed by different regions of different colors. For example they can detect a car as an object since it is often made of a single region with a uniform color, but they cannot detect a man as a single object since usually it is a composition of differently colored regions (head, trousers and shirt for example).

These methods, in general, suffer from oversegmentation and segmentation using intensity only or texture still remains a problem. Moreover intraframe segmentation alone cannot provide certain types of some information about the same object along the sequence such as for example its trajectory or size and shape change.

Another family of approaches is the motion segmentation one: it is based on the hypothesis that a moving object can be very well characterized by its motion consistency over time. In this case a dense motion field is used for segmentation and the pixels with homogeneous motion field are grouped together (see for example [17], [18], [19], [20] and [21]). In [17] an affine-clustering-based algorithm is presented; in [18], instead of using optical flow, a method to estimate motion models and their layer supports is proposed. In [22] another motion segmentation algorithm is proposed: the image is partitioned into rectangular regions and affine motion parameters are computed for each region; these motion parameters are then clustered to form homogeneous regions with similar motion parameters. In [21] an algorithm to track an object based on the multi-resolution estimation of an affine model from the motion model of the motion field inside the object is proposed.

The drawbacks of all these methods lies in estimating a reliable dense motion vector which is computationally very demanding and difficult to estimate; moreover they concentrate on detecting moving objects and cannot track objects with intermittent motion (e.g people that walk and then stop and then start walking again). Furthermore, due to the accuracy limitation of motion estimation, motion segmentation may not provide precise object boundaries.

Remaining in the motion segmentation field, deformable models have been studied to track non-rigid objects. Active contours (*snakes*) [23] are one of the basic energy-minimizing elastic contour models. As this method can handle only slow motion and is sensitive to textured image regions, many improvements ([24] and [25]) have been developed.

One significant defeat of motion-based object extraction methods comes out of inherent defeats of estimated optical flow (motion field). As it is shown in [26], the problem of motion estimation is ill-posed. Indeed motion estimation is based on the hypothesis of intensity conservation. It means that the intensity of a pixel is constant along its trajectory in time, if there is no noise or changing in lightening.

Mathematically, this hypothesis can be expressed as follows: let us denote $I(x, y, t)$ a spatio-temporal image function and $\vec{d}(x, y) = (dx, dy)^T$ the displacement vector of a pixel (x, y); let us denote $DFD(x, y, t+dt)$ the *Displaced Frame Difference* defined as:

$$DFD(x, y, t + dt) = I(x + dx, y + dy, t + dt) - I(x, y, t) \tag{2.1}$$

The hypothesis of intensity conservation is then expressed as:

$$DFD(x, y, t + dt) = 0 \tag{2.2}$$

Let us now consider the development of $I(x+dx, y+dy, t+dt)$ in a Taylor series around the point (x, y, t) up to first order. Then if we suppose the linearity of image gradients $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial t}$, we have:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt \tag{2.3}$$

From 2.1 and 2.2 it follows:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} = -\frac{\partial I}{\partial t} \tag{2.4}$$

or

$$I_x u + I_y v = -I_t \tag{2.5}$$

where u and v are the components of velocity vector of pixel (x, y). This equation is called OFE (*Optical Flow Equation*); it links velocity field or optical flow with gradient of intensity. Let us now consider this equation in vector form:

$$\nabla \vec{I} \cdot \vec{w} = -I_t \tag{2.6}$$

with $\vec{w} = (u, v)^T$. Let us now present $\vec{w}$ as $\vec{w} = \vec{w}_{\parallel} + \vec{w}_{\perp}$. Here we denote $\vec{w}_{\perp}$ the component orthogonal to the local image contour, that is parallel to the image gradient $\nabla \vec{I}$; $\vec{w}_{\parallel}$, on the contrary, is the component parallel to the local contour and thus orthogonal to the gradient, that is $\vec{w}_{\parallel} \cdot \vec{w}_{\perp} = 0$. The 2.6 becomes:

$$\nabla \vec{I} \cdot (\vec{w}_{\parallel} + \vec{w}_{\perp}) = -I_t \tag{2.7}$$

From the distributive property it can be deduced that:

$$\nabla \vec{I} \cdot \vec{w}_{\parallel} + \nabla \vec{I} \cdot \vec{w}_{\perp} = -I_t \tag{2.8}$$

The first term in the left-side sum is always 0. This means that only the component $\vec{w}_{\perp}$ of vector $\vec{w}$ can be estimated. This is the reason why motion vectors on the border of many objects estimated by pixel-wise methods are very noisy.

Let us consider estimators based on the criteria derived from (2.1) minimizing the DFD or DFD-based quality measures. Such are the block-based motion estimators used for standards MPEG which we will consider in the following chapters. They proceed by matching blocks of the current frame $I(x, y, t)$ with reference frames as it is illustrated in Figure 2.1 minimizing criterion $C(DFD)$ defined on a block.

Then, the ill-poseness of motion estimation is expressed by the so-called "aperture" problem. The corresponding position is searched inside a window. The window is limited, then the risk is as well the estimation of a wrong displacement vector: only $\vec{w}_{\perp}$ can be estimated as it is illustrated in Figure 2.2.

Finally, the estimation is disturbed by noise and texture present in real video sequences. All these factors make the estimation of optical flow on the borders of objects very noisy, thus motion-based segmentation techniques often lead to very noisy object borders. This is why combined techniques based both on motion estimation and intra-frame colour/intensity based frame segmentation became popular.

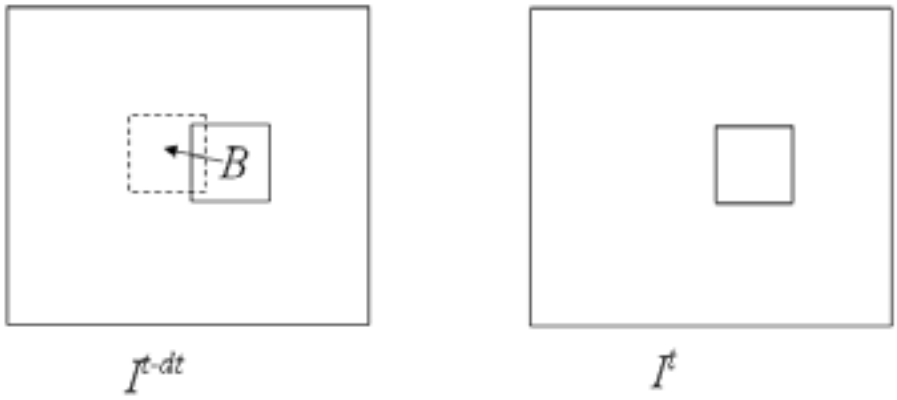


Figure 2.1: Block-matching motion estimation.

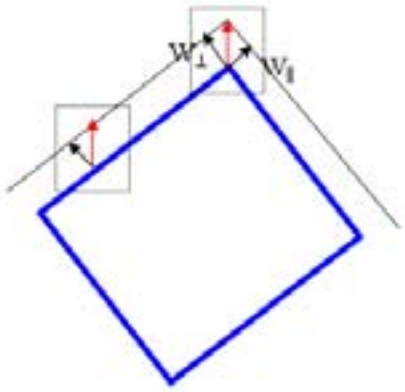


Figure 2.2: Illustration of "aperture" problem.

2.3 Combined approaches to object segmentation

Since both approaches have their own drawbacks, most of the video object segmentation tools integrate both spatial and motion segmentation techniques ([27], [28], [29], [30], [31], [32], [33], [34] and [35]). In fact it can be difficult to group pixels into objects based on similarity of their optical flow vectors, so it seems to be inevitable that additional information such as color or texture must be included to accurately detect boundaries of moving objects.

[30] presents an algorithm based on two different parts: a temporal segmentation to localize moving parts of objects in the image and a spatial segmentation to divide

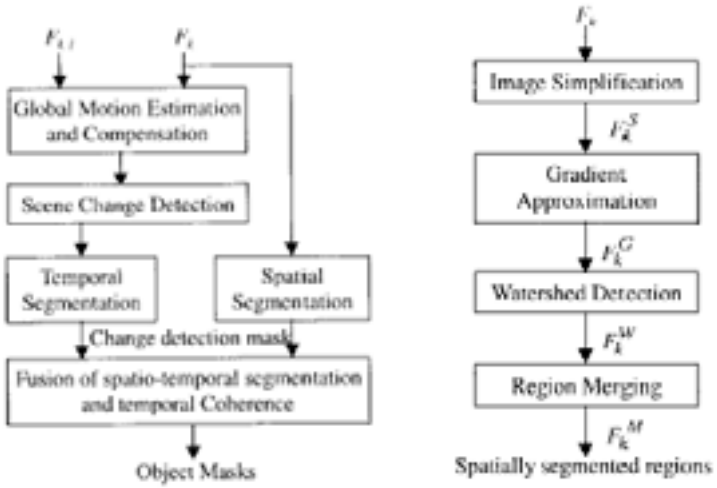


Figure 2.3: in a) the scheme of the algorithm presented in [30] and in b) the spatial segmentation scheme.

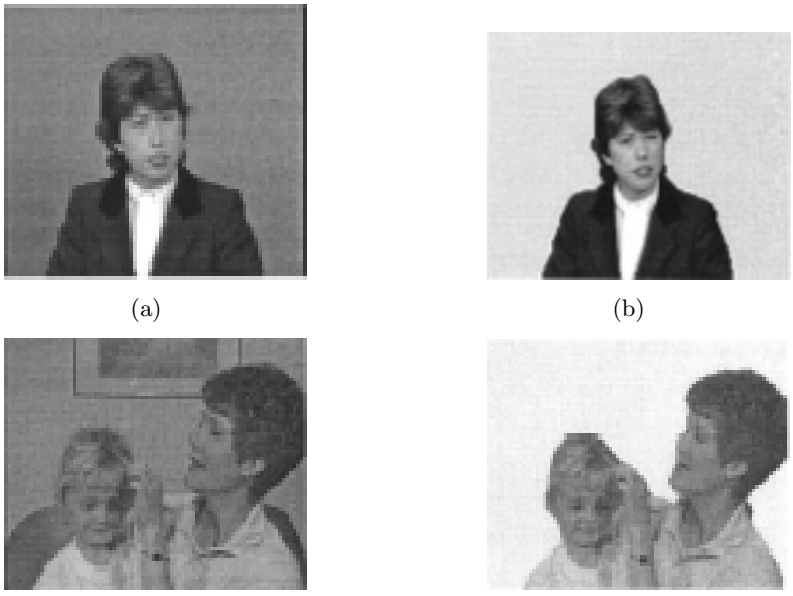
the image into semantic regions with precise object boundaries. The combination of the two results produces an accurate segmentation of the moving objects. In Figure 2.3a) is presented the scheme of the algorithm.

First step is global motion estimation and compensation. For the global motion estimation, an affine motion model is adopted, thus the elementary displacement vector can be modelled as:

$$\vec{d}(x,y) = \begin{pmatrix} dx(x,y) \\ dy(x,y) \end{pmatrix} = \begin{matrix} a_1 + a_2 x + a_3 y \\ a_4 + a_5 x + a_6 y \end{matrix} \tag{2.9}$$

First, local motion vectors are calculated with a block-matching algorithm, then least square method is used to estimate the six affine model parameters. A global motion compensation is then performed to remove the camera motion. To perform the temporal segmentation and extract moving objects, differences of intensity values between two successive image frames are taken to find moving objects in space through time evolution.

While temporal segmentation subdivides the images in term of motion characteristics, spatial segmentation splits the entire image into homogeneous regions in term of intensity. Here, the segmentation is implemented as in the framework of morphological approach [12]. Segmentation process is subdivided in four parts: image simplification, gradient calculation, *Watershed* segmentation and region merging



(a) (b)

(c) (d)

Figure 2.4: Some results obtained in [30]: in a) and c) the original images and in b) and d) the extracted objects.

(see 2.3b)). For image simplification, erosion and dilation are used to build a filter that removes regions that are smaller than a given size but preserves the contours of the remaining objects in the image.

Through the image simplification, the inside of each homogeneous region has small gradient, but large gradients are induced along the region boundaries separating different homogeneous regions in the image. So morphological gradient is calculated to partition the image into homogeneous intensity regions. The gradient image is used as input for the *Watershed* algorithm. A modified *Watershed* segmentation is applied to obtain small regions homogeneous in terms of luminance and color. But as *Watershed* segmentation often produces oversegmented results, a region merging process is further applied. This region merging step uses a graph-based clustering: small regions are recursively merged with their neighbors on the basis of their spatio-temporal similarity. Spatial segmentation and temporal segmentation are then merged together to discriminate foreground objects from background regions. In Figure 2.4 some results are presented.

Different authors have proposed other combined approaches that merge the results of motion and spatial segmentation. In [36] the approach is divided in three

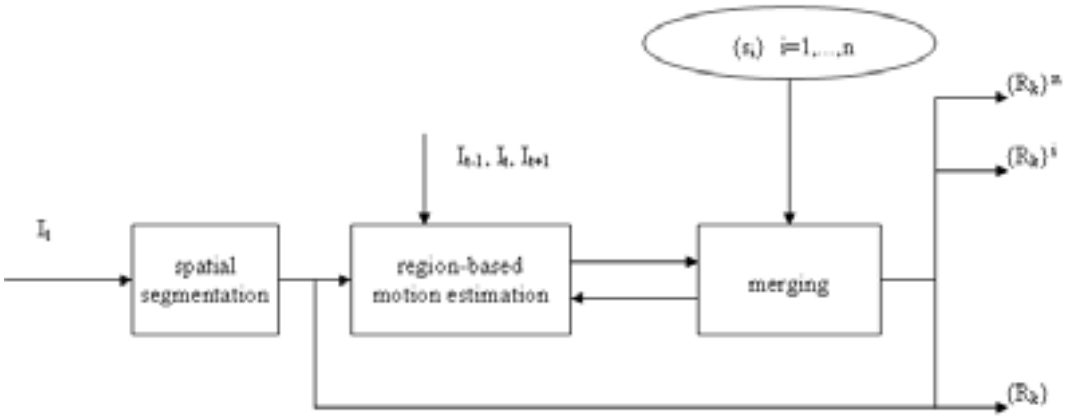


Figure 2.5: Block diagram of the bottom-up strategy for the construction of the hierarchy.

steps, region simplification, region-growing and motion-based region fusion. In [34] an approach is presented to combine motion segmentation using image subtraction with static color segmentation; in [35] an algorithm is developed to match edge detection and line approximation results with motion segmentation to determine object boundaries. In [37] a method is proposed which allows to take into account both global camera motion and local motion homogeneity of adjacent regions in image plane. The method proceeds by estimation of global camera motion and identification of areas which do not follow the global model. The goal of this work is to construct a set of representations for the same video sequence which are characterized by different degrees of detail. Each representation is a segmentation map of an image plane, that is a partition of the image plane into a set of regions.

The goal of the segmentation process proposed in [37] is to construct a hierarchy L of content representations from the very beginning of an image sequence, using a limited number of frames. Two strategies are proposed to develop this hierarchy. The first one is a purely bottom-up approach, which uses a local homogeneity with regard to gray-level and motion-based criteria. The second approach uses a global information of the scene composition, which is extracted by a dominant motion analysis, to guide a bottom-up approach locally.

The bottom-up approach starts with the lowest level of hierarchy ($i = 0$) corresponding to the gray-level based spatial segmentation of the second frame of moving sequence (see Figure 2.5). The image is segmented into regions R_k^i (in this case

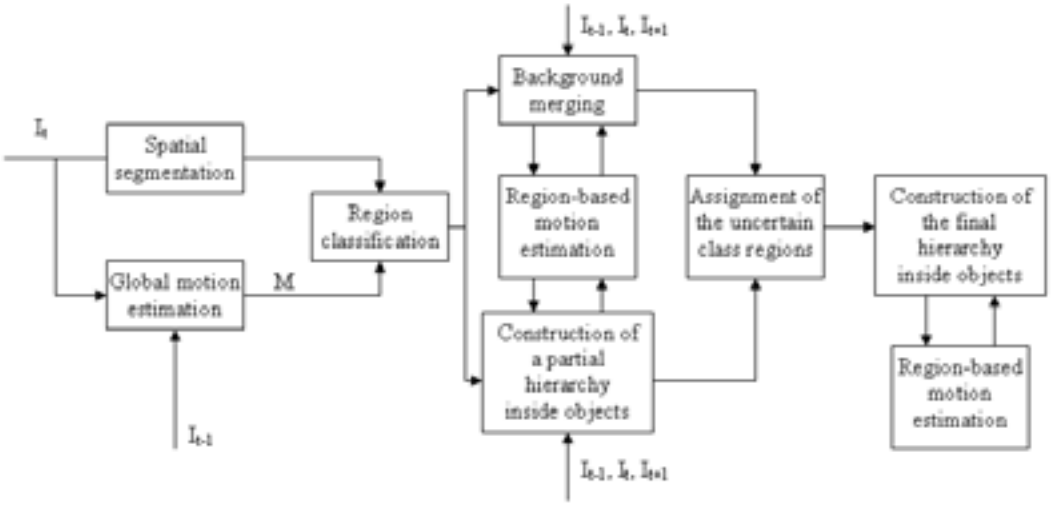


Figure 2.6: Block diagram for the combined strategy of hierarchy construction.

$i = 0$ because it is the lowest level of the hierarchy) and for each region motion parameters θ_k^i are calculated. An ordered discrete set of values of some parameter s is introduced: $s = \{s^i\}; 1 \leq i \leq n; s^1 < s^2 < \cdots < s^n; s^i > 0$. For each value of this parameter s^i, a level l^i in the hierarchy L is matched. Given a current level of the hierarchy l^i, the next level l^{i+1} is constructed by a successive merging of couples of adjacent regions R_q^i, R_p^i in such a way that the variation of the quality of motion compensation on each region in a pair is controlled by the value s^{i+1} (see Figure 2.5).

When $s^{i+1} \leq 1$, then such a merging corresponds to a better motion compensation in a merged region. This situation is possible as the quality of motion compensation on the spatially homogeneous regions can be poor. If $s^{i+1} \geq 1$, it means that regions with similar motions are merged.

If the set of values of s is chosen properly and the moving scene contains objects with similar motions which differ strongly from the dominant motion in the scene, then such a strategy leads to the correct segmentation of a scene into objects and background at the highest level of the hierarchy. But in more complex situations are rather possible, for example a low difference in motion magnitude of objects and background, and also, strongly different motion magnitude of objects. In these cases the merging proposed in the bottom-up approach is not sufficient.

For this reason, a second approach based on a global analysis of motion in the

scene is proposed as well (see Figure 2.6). This analysis is done by a motion estimation on the whole frame. Here a robust motion estimator with outlier rejection is used. Such an estimator allows the extraction of a set of pixels in the frame which do not follow a dominant motion in the scene. This set of pixels called "motion mask" M indicates the location of objects having a different motion (motion masks will be also analyzed in the next chapter). As the motion mask does not exactly correspond to the outlines of objects, the goal of a further local analysis is to exactly define objects and to construct the hierarchy inside them. The motion mask M is superposed to the spatial segmentation map $\{R_k^i\}$ at the first level of hierarchy $i = 0$. At this step, spatial regions are classified into three classes. Regions which contain a small relative number of outliers are assigned to the 'dominant motion' class. Those containing a strong number of outliers are assigned to the 'object' class. Finally, regions with an intermediate percentage of outliers are classified as 'uncertain' ones. The result of this classification step guides further steps of the method which are realized in a bottom-up manner.

Pixels belonging to the 'dominant motion' class represent a motion homogeneous area which, in most cases, corresponds to connected components in the background of the scene. This is why the superposed spatial regions are simply merged into connected components of the background (see Figure 2.6). With regards to the 'object' class, as it can contain objects with different motions and objects with non-homogeneous motions inside them, then the regions of this class are merged in a bottom-up manner according to the specified quality of motion estimation inside 'object' areas. After that, the 'uncertain' class regions are assigned to their best neighbors according to quality of motion compensation. Then using a tracking technique this representation of the content can be refined along the time.

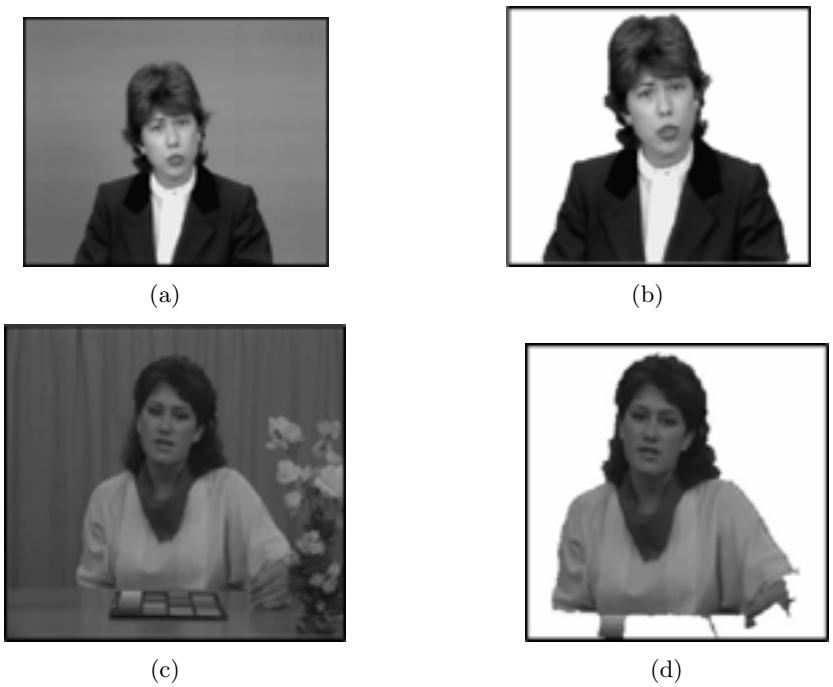
Instead of using purely spatial homogeneous regions, the method builds a hierarchy of motion-based segmentation; it proceeds by progressively merging regions according to motion-based criteria. Despite [37] does not use morphological approach, very much satisfactory results are obtained. In fact, in the context of combined 'spatio-temporal' methods the requirements to spatial segmentation algorithm are very strong. The reason of this is that it is natural to suppose that the motion border of moving object coincides with a luminance border in image frame. Counter examples are known in literature, but they are very much artificial for natural video. This is why morphological methods are so popular: giving a strong oversegmentation, they eliminate the risk to bypass motion borders when color-based or grey-level based segmenting a video frame.

In [27] the assumption is that objects of interest are not homogeneous with respect to low-level features such as color, intensity or optical flow and so conventional low-level semantic concept will fail to obtain correct partitions. The problem is formulated as a separation of the background from the foreground objects based on motion information (in fact physical objects are often characterized by a coherent motion that is different from that of the background). The main hypothesis underlying this approach is the existence of a dominant global motion that can be assigned to the background. Areas that do not follow this background motion then indicate the presence of independently moving objects. Two different methods are used to separate foreground objects from the background. The first approach is based on a morphological motion filtering and it can be applied to sequences where the objects present very slow motion (for example head-and-shoulder sequences): in this case the first step of the algorithm is to compute a dense optical motion field and, starting from it, to calculate the global motion parameters. Then a filter is applied to obtain the *independently moving components (IMC)*. The second approach instead is based on *change motion masks* and it is more suitable for sequences where objects move fast. The *change motion masks* are based on the hypothesis that the difference of grey level between two consecutive frames indicate objects that are moving or changing their shape. So these masks are extracted by calculating the absolute difference between two frames and filtering it to remove the noise. The drawback of this last method is that it can be used only with static camera. The results obtained by the two methods are then merged with the edge pixels of the object detected by the Canny operator [38] to derive a two-dimensional binary model for the object of interest. Once the model is initialized, it is tracked along the sequence.

From each motion model the corresponding foreground object is extracted with a two-step method. First the closed object boundary is approximated by a filling-in technique: it starts by assigning for each row the pixels between the first and last model point to the object. This procedure is repeated for each column and one more for each row. After this first step each wrong boundary is corrected separately. The wrong boundary is removed and a gap is left in the otherwise closed contour. The correct boundary between the two points of the gap is then determined using the Dijkstra's algorithm [39], assigning a weight to the different type of pixels.

This algorithm needs some inputs from humans in choosing the method to extract the motion masks and requires hand-tuning for some critical parameters.

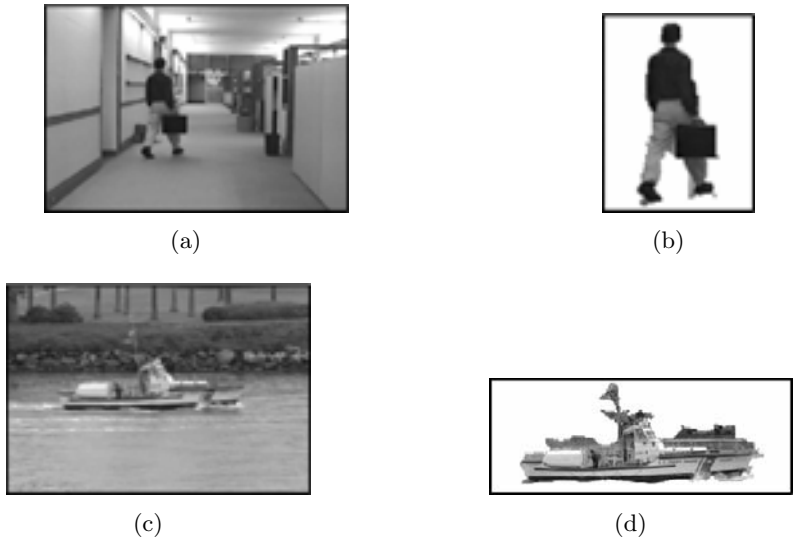
In Figure 2.7 some results from two head-and-shoulder sequences are presented; in a) and c) two frames extracted from the original video sequence are shown while



(a) (b)

(c) (d)

Figure 2.7: Some results obtained from two head-and-shoulder sequences [27]; a), c) original frames, b), d) segmentation result.



(a) (b)

(c) (d)

Figure 2.8: Some results obtained with the method of change motion masks presented in [27].

in b) and d) the obtained result is shown. In Figure 2.8 other results are presented; in this last case, the objects in the sequence are characterized by higher motion and so the method of *change motion masks* is used; in a) e c) the original frame and in b) and d) the extracted objects are shown.

In [40], [41] and [42] an object segmentation method based on region-based active contours is presented. This method gives very good result but is computationally very expensive. For this reason in [40], to improve numerical efficiency, a multiresolution approach is proposed; this approach is based on making the active contour evolve first on a low resolution image. The final contour obtained for this reduced image are used as an initial curve for the real size image.

In [29] a semantic object tracking system using mathematical morphology is proposed. In [31] the algorithm is based on luminance information and motion parameters: the luminance is filtered using a morphological operator as well and then clustered using *k-means* algorithm; at the end regions with similar motion are merged. In [43] spatio-temporal similarity is used as a merging criterion: spatial similarity is obtained from the test statistic of the gradient value along the boundary of the regions; the temporal similarity is derived from test statistic of the residual distribution and motion parameters.

We will see in the following of this work, how a morphological segmentation is used in our object extraction scheme.

2.4 Semi-automatic approaches

To get more reliable results, some authors propose some *semi-automatic approaches* that use human help to tune some parameters or to select regions of interest. One of the work presented above [27] may also be classified as a *semi-automatic approach*, in fact human help is needed to select the correct methodology for the video sequence to analyze.

Another *semi-automatic* approach is presented in [28]. Here the user identifies a semantic object. The input is a polygon whose vertices and edges are approximatively along the desired object boundary. User can also specify a set of parameters to start the tracking process or stop the tracking process at any frame, modify the object boundary that is being tracked, then restart the tracking process from the modified frame.

The segmentation process starts with the generation of three feature maps: the color map, obtained from the original image filtered to remove noise, the edge map,

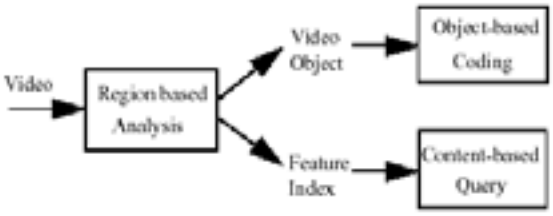


Figure 2.9: Object segmentation as described in [28].

generated by applying the Canny edge-detection algorithm [38] and the motion field generated by a hierarchical block matching algorithm [44]. Region segmentation is performed by a modified merge-and-split method where edge information is fused directly in the color merging process. It is an interactive spatial-constrained method based on color distances between adjacent regions and edge information.

First, a color-pixel labelling process is applied to the color map: one label is assigned to a group of neighboring pixels with the same (or very similar) color. To prevent assigning one label to two regions with the same color but separated by edge pixels, only non-edge pixels are labelled in the process. This process generates an initial group of regions. Then, color distances between two connected regions are computed, and they are merged if the color distance between them is smaller than a given threshold and if it is smaller than all the other distances between these two regions and their neighbors. Once a new region is generated from two adjacent regions, its mean color is computed by taking weighted average of the mean colors of the old regions. The merging is iterated until color distances between every two connected regions are above the color threshold.

Then the segmentation result is compared with the initial object boundaries from the user input and regions are labelled as *foreground* or *background* regions. Last, affine motion parameters of all regions, including both foreground and background, are estimated by a linear regression process over the dense optical flow inside each region.

Once the first frame has been segmented, the tracking process starts: segmented regions from the previous frame are first projected onto the current frame using their individual affine motion models. Even for the current frame the three feature maps are computed. Then the projected regions are used in the segmentation process as seeds in another color-based region growing process. Regions are then aggregated

(a) (b) (c)

Figure 2.10: The results obtained with the *semiautomatic* method proposed in [28].

to compose foreground objects. Region aggregation include two inputs: the homogeneous region and the estimated object boundary. The module implements an iterative region grouping and boundary alignment algorithm based on the estimation object boundary as well as the edge and motion features of the region. The aggregation and boundary alignment process is iterated multiple times to handle possible motion projection errors, especially for fast motion.

In Figure 2.10 the results obtained with this method are presented. We can notice that the objects of interest are detected very precisely, but the drawback of this method is that human help is strongly required to manually detect the object boundaries in the first frame of each sequence.

It is possible to find other *semi-automatic* algorithms in the literature: for example in [45] and [46] a user initially has to select objects in the scene by manual segmentation. Then the objects are tracked and updated along the sequence; in [47] instead user interaction is necessary to directly tune some crucial parameters or to select an area containing the objects of interest where these critical parameters can be more easily estimated by the proposed algorithm. In [48] a segmentation approach with minimal interaction is proposed; in this algorithm to extract object to be tracked a spatial color based segmentation of a video frame is applied. The spatial segmentation is the result of a morphological method based on modifies watershed. Once the frame has been segmented the regions must be semantically labelled and tracked. But automatic motion-based labelling is possible only for simple scenes, where a strong difference of the dynamic range and of the textural characteristic of objects and the background is observed. In general case, instead, an object can be partly static and thus, it cannot be distinguished from the background based on motion difference. Therefore user interaction is required to completely extract the object: the user creates an object masks on the first frame by encircling objects in

foreground.

The reason for interactive segmentation methods expansion in 1999-2001 can be explained very simply. None of automate methods is able to precisely estimate object borders with pixel accuracy. It may often result in a strong segmentation noise. This is also one of the reasons why today MPEG4 is not popular. It is impossible to precisely extract object for encoding without human interventions. In coding applications (VOP coding of MPEG4), the precision on the object borders is a must, as the boundary pixels yield the strongest motion error. On the contrary, in indexing context (MPEG7), segmentation errors are not so crucial. This is due to the fact that when retrieving objets by color, shape, motion criteria, some tolerance is admissible. Obviously, the similarity measure (or distance) have to be sufficiently robust against small segmentation errors.

Therefore, fully automatic methods can be employed for object extraction. This allows for processing of huge video data such as regular TV programs, sports or artistic video content.

2.5 Spatio-temporal video object segmentation

Interactive or automatic spatio-temporal segmentation has proven to give the best results with respect to video object segmentation because it can overcome the problems of purely spatial or motion-based approaches, nevertheless maintaining their advantages.

Nevertheless it is not possible to unambiguously define an objective function for a semantic video object, an object can be well characterized by its motion information.

Though much work has been done in the area of motion-based video object segmentation in the pixel domain [49] [50] [51] [52] [53] (see section 2.2 and 2.4 as well), little work has been carried out in compressed domain. Pixel domain motion segmentation is based on the motion information at each pixel location, obtained for example by optical flow estimation, which is computationally very demanding. The uncompressed pixel domain provides these algorithms with the potential to estimate object boundaries with pixel accuracy, but also requires that the processed sequence is fully available before segmentation can be performed. As a result, the usefulness of such approaches is usually restricted to non real-time applications; this is due to the high computational complexity resulting from the large number of pixels that have to be processed. Real-time pixel domain methods [54] are usually applicable only on head-and-shoulder sequences for video conference applications or are based

on the assumption that the background is uniformly colored, which is not always valid in practice.

On the other and, in a compressed domain, the motion information available in an MPEG stream is only one motion vector per macro-block, which is too sparse and may not be very reliable to perform motion segmentation. So most of the compressed domain methods have only been based on spatial information such as color or edge information [55].

2.6 Spatio-temporal object segmentation in compressed domain

As we stated before, the hope that indexing of multimedia content will systematically happen at the encoder and simultaneously with encoding processes has not been realized up today. The technology of coding due to regularity of color approaches (block-based) of actual standard, made easy a batch encoding of tremendous amount of video data. Therefore, the problem of indexing is just transferred to the compressed domain. It is obvious that compressed video content can be indexed after a full decoding has been done.

Nevertheless, since the decoding is a relatively expensive process, segmenting the objects and extracting their features directly from the compressed domain would be an effective way to achieve fast and efficient algorithm for searching a large database and indexing objects. In fact it is less expensive to directly process in the compressed domain rather than decoding the video in the spatial domain. The block structure of the compressed domain data also drastically condenses the amount of data to be processed.

In addition to reduced computational complexity, there are several other advantages of performing the analysis in the compressed domain such as, for example, that motion information is readily available without incurring in the cost of estimating the motion field. But compressed domain analysis have limitations as well. The Discrete Cosine Transform (DCT) removes the spatial correlation among the pixels within a block, thus the precision of the segmentation degrades with the block dimension. Moreover, since the goal of motion compensation is to provide a good prediction but not to find a physical motion field, the motion vectors are often contaminated with estimation errors.

Even if various segmentation approaches have been investigated, most of them are based on examining the images in pixel domain and only few researchers have

proposed some segmentation algorithms in the compressed domain. For example, in [56] JPEG documents are segmented into specific regions such as those containing text using the encoding cost map (ECM) based segmentation; in [57] a fast algorithm to automatically detect faces starting from MPEG compressed video is presented: the method is based on skin-tone statistics, shape constraints and energy distribution of the luminance DCT coefficients. In [58] p-adic distance is used on DCT coefficients of colour frames. It uses both DC and AC coefficients, thus exploring local textural components inside blocks. P-adic metric, which is a ultrametric, allows for a very fast segmentation by clustering.

Despite compressed segmentation methods have appeared since late 90's, the methods cannot be considered as a simple transposition of previous spatio-temporal segmentation methods onto compressed domain. The most recent compressed-domain methods for objet extraction use both color and motion information and propose solution to overcome noisiness of this information in a compressed stream.

One of the first methods that perform object segmentation starting directly from MPEG2 flow is the one proposed in [59]; in this case a system is proposed to incorporate the motion information corresponding to few frames on either sides of the current frame to enrich motion information, which would otherwise be too sparse in compressed domain to correctly detect objects with proper motion. The first step is a *motion accumulation* step which takes the compressed video sequence as input and decode the motion vectors from the intercoded (P and B) frames of an MPEG stream. In the process of accumulating motion vectors, the motion vector obtained from the current macro-block of the current frame n is assigned to the center pixel (k, l) of that macro-block. Let $m_x^{kl}(n-c)$ and $m_y^{kl}(n-c)$ represent the motion vectors along the horizontal and vertical directions for the macro-block centered at (k, l) in the frame $(n - c)$. Then, the new position for this macro-block in the current frame can be estimated as:

$$(\hat{k}, \hat{l}) = (k, l) + \sum_{f=n-1}^{n-c} (m_x^{kl}(f), m_y^{kl}(f)) \tag{2.10}$$

The motion vector $(m_x^{kl}(n-c), m_y^{kl}(n-c))$ in the $(n-c)^{th}$ frame is assigned to the new position $(\hat{k}, \hat{l})$ with respect to the current frame. Motion accumulation is also done by tracking frames in the forward direction from the current frame. In the forward tracking, the motion vectors are accumulated according to the following equation:

$$(\hat{k},\hat{l}) = (k,l) - \sum_{f=n+1}^{n+c} (m_x^{kl}(f), m_y^{kl}(f)) \tag{2.11}$$

where the motion vector $(m_x^{kl}(n+c), m_y^{kl}(n+c))$ in the $(n+c)^{th}$ frame is assigned to the new position $(\hat{k},\hat{l})$ with respect to the current frame. This motion accumulation is performed over few frames on either side of the current frame. The accumulated data is further processed to remove the noisy motion information. A two-dimensional median filter is used to remove the noise from the accumulated sparse motion vectors. This filter operates individually on nonzero elements of horizontal and vertical motion data. The set of motion vectors obtained by the above process is sparse and nonuniformly spaced. Thus a Delaunay triangle-based surface interpolation [60] scheme is used to get the dense motion field for the current frame. This interpolation technique fits the surface that passes through the given data points. The dense motion field obtained is further processed by a Gaussian filter to get a smoother dense motion field. At the end of these steps, a motion vector for each pixel is obtained.

The dense motion vectors and the number of motion models are given as input to the *segmentation module*. Since each video object can be characterized by the motion information, an *affine parametric motion model* is used to describe the corresponding object region. First, the static object (usually the background) is segmented by assigning the pixels with zero motion to a single layer. The remaining pixels with motion are segmented into different layers by applying the *EM* algorithm, which is an iterative technique that alternatively estimates and refines the segmentation. Given the number of motion models N_0, extracted from each nonoverlapping square region of the dense motion field whose variance is less then a predefined threshold, and the corresponding initial motion hypothesis expressed in terms of affine parameters vectors $\{a_1, a_2, \ldots a_{N_0}\}$ (each a_i is a six-dimensional vector), the *EM* algorithm alternates between the E-step and M-step until convergence. The E-step computes the probabilities associated with the classification of each pixel p as belonging to the k^{th} class with motion parameters a_k. Let

$$R_k^2(p) = (u_k(p,a) - v(p))^2 \tag{2.12}$$

be the square residual between the predicted motion $u_k(p,a)$ and the interpolated motion $v(p)$ at pixel location p, then the likelihood of p belonging to k^{th} class is given by:

$$L_k(p) = \frac{e^{-\frac{R_k^2(p)}{2\sigma^2}}}{\sum_{j=1}^{N_0} e^{-\frac{R_j^2(p)}{2\sigma^2}}} \tag{2.13}$$

where σ^2 controls the fidelity of the affine model fit to the dense motion vectors. Then the M-step refines the motion model estimates given the new classification arrived at E-step. The motion model parameters are refined by minimizing an error function using a weighted least-square estimation. The function to be minimized is:

$$J(a_k) = \sum_{p \in R} L_k(p) R_k^2(p) \tag{2.14}$$

where p represents the position of the pixel within the square region R with respect to a common origin. After few iterations between E-step and M-step (typically four to six), the final video object plane is obtained.

Once the initial segmentation of the objects is obtained, the following frames are further tracked temporally to generate a sequence of video objects. The same motion parameters obtained after the segmentation of the initial frame are used for tracking in the future frames. However, this tracking technique holds good only when no objects enter or leave the frame; in such a case the number of object has to be determined again, prior to segmentation. After the segmentation step, the obtained video objects are further processed to perform an *edge refinement phase*, where the pixels belonging to edge regions can be assigned to correct objects. In fact the results obtained from the EM algorithm suffer from poor edge localization, and whereas for MPEG4 video objects it is essential to get good edge localization. So the blocks belonging to the edges and their eight connected neighbors are completely decoded and each pixel is assigned to one object or to the other on the basis the motion direction.

Even if this method [59] starts working with motion vectors extracted from an MPEG2 stream, for the segmentation step it works with dense motion vectors and in the pixel domain; moreover partial decoding of the stream video is also needed to perform the edge refinement phase; so, even if it is faster and more efficient than pixel domain methods it cannot be implemented in real-time.

With this example we have seen that, if the "standard" goal, such as a precise segmentation of object from compressed stream is searched for, then an efficient use of motion and color information from compressed stream is not so advantageous as it could be expected. The segmentation method requires local decoding, heavy filtering

and tricky combination of compressed and pixel domain video processing. Hence, the real-time for such methods is still impossible. Nevertheless, the need to process a huge amount of video data, specifically on the decoder and with low power hardware (such as home multimedia devices or mobile devices) requires real-time methods.

2.7 Real-time segmentation of video object

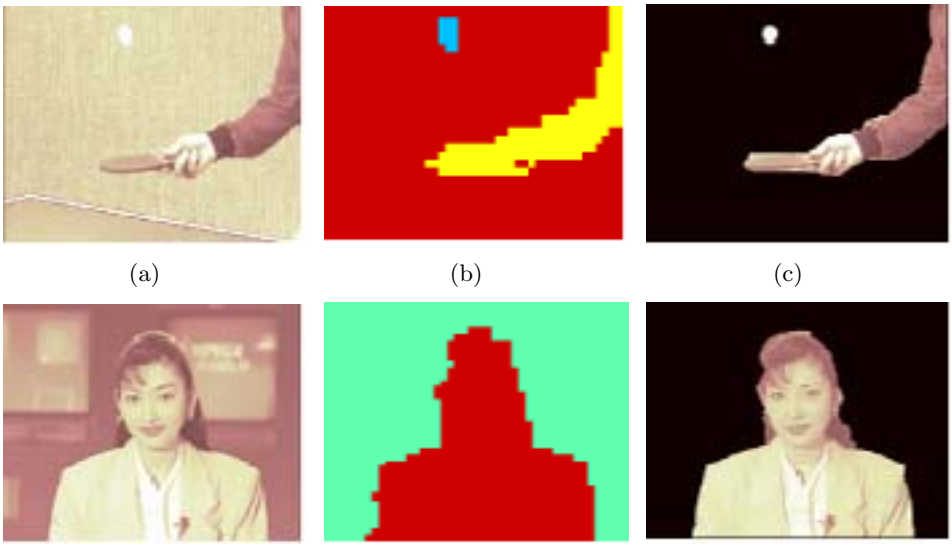
Using of already available motion information from compressed stream allows for reduce computational cost for the most heavy operations such as motion estimation. Therefore, real-time methods can be implemented.

A real-time algorithm to track moving object using macro-block motion vectors is presented in [61], but this is a semi-automatic algorithm because the region to be tracked has to be manually identified from the user.

In [55] a fast algorithm to detect and segment objects in MPEG compressed domain is proposed. The algorithm is organized in two distinct parts: the first part proposes a method to approximately extract the objects directly from compressed domain, while the second part extracts object details to better define object boundary. The main advantage of this algorithm is that for a fast low-level indexing only the first part can be used, obtaining thus some first results in **real-time**. If a precise detection of boundaries is needed, then second part of the algorithm can be performed; in this case small portions of the video frame are decoded.

The first step in this algorithm is an initial segmentation generated from 3D spatial information based on the DC image and AC energy information. Color and energy information is used to cluster the image using sequential leader clustering to form homogeneous regions without knowing the number of clusters in advance. The sequential leader clustering uses a threshold to decide whether the input data should be in the existing cluster or a new cluster should be created. After having obtained a number of clusters, adaptive k-means clustering is applied iteratively to the image until no more changes occur in each cluster. The clustered regions with areas less than a threshold are then merged into their neighbors using luminance and ac energy distance. The result of this process is the set of initial regions to be used in spatio-temporal segmentation.

Both spatial and temporal information are used as criteria of the region-merging algorithm and when two adjacent regions are combined into a single value, both spatial and temporal similarities are measured; in fact spatial information ensures the boundary of the objects, while temporal information provides the temporal change



(a) (b) (c)

(d) (e) (f)

Figure 2.11: The results of the method proposed in [55].

characteristics.

For spatial similarity, the entropy of the ac energy from the luminance information is calculated. For temporal similarity instead, a 3D Sobel filter is applied along the (x,y,t)-axes. The spatio-temporal similarities are calculated and used to create a similarity graph between regions. Then an iterative clustering process is performed until the minimum number of regions is reached. The result of spatio-temporal segmentation is then applied for foreground-background classification in order to separate the objects from the background. The classification decision is based on the average temporal changes of regions. Even though these features extracted in compressed domain are coarse, they can be useful for fast video indexing. If detailed features, such as contours, are required, it is possible to decode DCT coefficients around the boundary of objects and obtain the detailed edges in the pixel-domain using Canny edge detection [38] algorithm only in the neighborhood at the object borders.

In Figure 2.11 some results are presented. In figures 2.11 a) and d) two of the analyzed frames are presented; in b) and e) it is possible to see the extracted objects. We can notice that object borders are not well defined; in fact these first results have been obtained using DC coefficients without decoding the MPEG sequence, so the results are obtained at the DC resolution. Working with low spatial resolution allows

to obtain these results in real-time. In case more precise contours are needed, as in the case of VOP extraction for MPEG4 coding, it is possible to further decode the sequence and segment object boundaries with pixel accuracy. The results of this last process are shown in figures 2.11 c) and f).

This method [55] is very important because it introduces a new concept: a coarse result obtained in real-time, for a certain type of application can be more important than a detailed result obtained off-line. The real-time result is obtained thanks to the use of Sobel filtering applied in time to estimate similarity instead of a complex motion model calculation as precedent works. The use of this filter is less computationally intensive than a motion estimation algorithm, but it is also less reliable. However the proposed method have some limitations: it requires several preset thresholds and the value of the sequential clustering threshold is crucial to determine the number of objects. Besides, k-means clustering needs appropriate weights for the block coordinates, DCT coefficients and motion information.

The main drawback of the method is an inefficient use of motion information. Indeed, if the motion of object in video is strong, then the 3D (x,y,t) approach could not be efficient as object masks will be disconnected along t axis.

To address the shortcoming of the above approaches, [62] proposes a different real-time algorithm that blends motion and frequency information (in Figure 2.12 the flow diagram of the algorithm is presented). After parsing the MPEG flow into DCT coefficients and motion vectors, a frequency-temporal data structure is constructed for multiple GOPs between two scene-cuts. The DCT coefficients and motion vectors of the GOPs are assembled into a 3D data where each element corresponds to the attributes of an 8×8 block. The components of the feature vector include some DCT coefficients (*dc* coefficients of the Y,U and V channels, selected *ac* components of the Y channel), an energy term E and the forward-predicted motion vector.

Starting from this frequency-temporal feature representation, a volumetric region-growing algorithm is performed. In order to choose the seeds to start the growing process the gradient volume for the Y channel is calculated. Starting from the seed with minimum gradient value, the volume grows including all neighbor vectors that full within a certain *distance* from the seed. As these vectors contain different terms the following distance metric is used:

$$\delta(f^i, f) = \omega_{dc}\delta_{dc}(f^i, f) + \omega_{ac}\delta_{ac}(f^i, f) + \omega_{mv}\delta_{mv}(f^i, f) \tag{2.15}$$

where

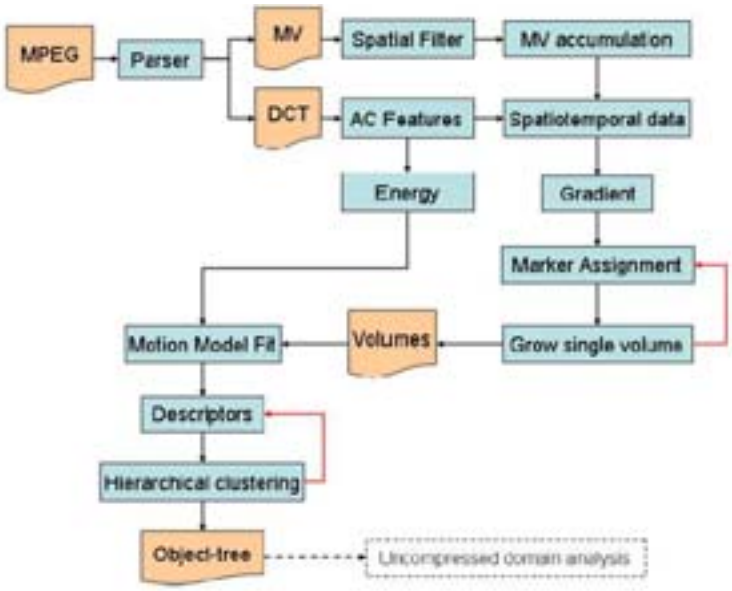


Figure 2.12: Flow diagram of the segmentation algorithm presented in [62].

$$\delta_{dc}(f^i, f) = |f^i(dc_y) - f(dc_y)| + |f^i(dc_u) - f(dc_u)| + |f^i(dc_v) - f(dc_v)| \quad (2.16)$$

$$\delta_{ac}(f^i, f) = \sum_{k=1}^{3} |f^i(ac, k) - f(ac, k)| \quad (2.17)$$

$$\delta_{mv}(f^i, f) = \sqrt{(f^i(mv_x) - f(mv_y))^2 + (f^i(mv_x) - f(mv_y))^2} \quad (2.18)$$

where ω_{dc}, ω_{ac}, ω_{mv} are the weights of the corresponding distances. These weights determine how much each attribute contributes to the distance metric. If the color distance is less than a predefined threshold the vector f is included in the volume.

Once a volumetric region stops to grow, another seed is chosen to determine the growth of another region. After volume growing, the obtained parts of the video are consistent in terms of their DCT coefficients and translational motion distribution. The next step is to fit a motion model to each volume. This is obtained by first estimating the affine motion parameters of the regions of a volume in the corresponding layers, then averaging the set of individual parameters over all the layers. The last step is to cluster the segmented volumes into objects using their

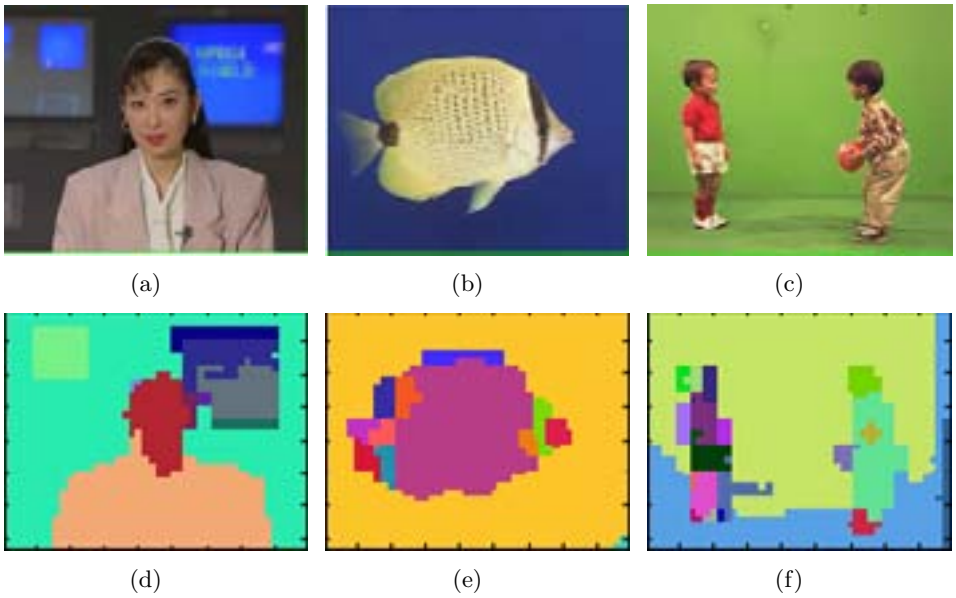


Figure 2.13: Some results of the real-time algorithm proposed in [62].

motion parameters; this is accomplished by merging the regions with the most similar parameters and then updating the motion parameters.

In Figure 2.13 some of the results of the method proposed in [62] and discussed above are shown. In figures 2.13 a), b) and c) three frames extracted from different sequences are presented. In figures 2.13 d), e) and f) the images are presented subdivided into homogeneous regions both in terms of color and motion characteristics.

We have to notice that the results are given on test sequences with very slow object motion on a static background. The paper rises an interesting problem of how to merge both spatial and motion information in a unique feature space in which classical segmentation approaches can be performed. This question still remains open, as the results presented analyze video scenes with limited typology.

To allow efficient indexing of large video databases, another algorithm for real-time, unsupervised spatio-temporal segmentation of video sequences in the compressed domain is proposed in [63]. In this method, only I and P frames are examined, since they contain all the information that is necessary for the algorithm; this is also the case for most other compressed-domain algorithm [62].

The first step of the algorithm proposed in [63] consists in information extraction from the compressed domain. Motion vectors are extracted from the P-frames and are used for foreground/background segmentation and for the subsequent identifi-

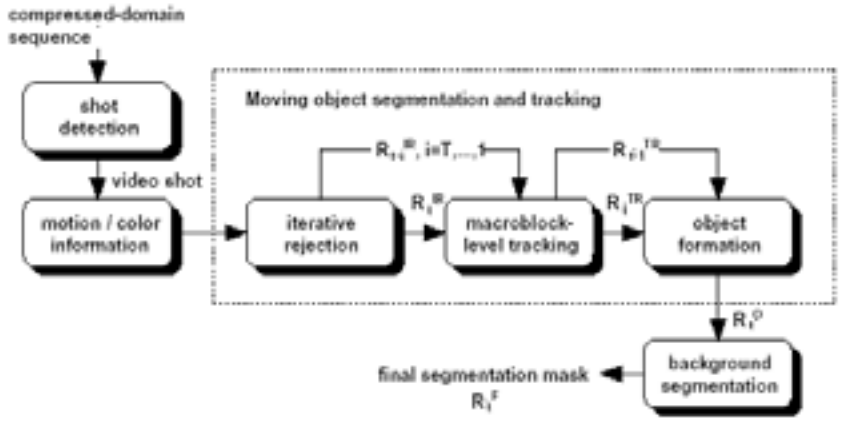
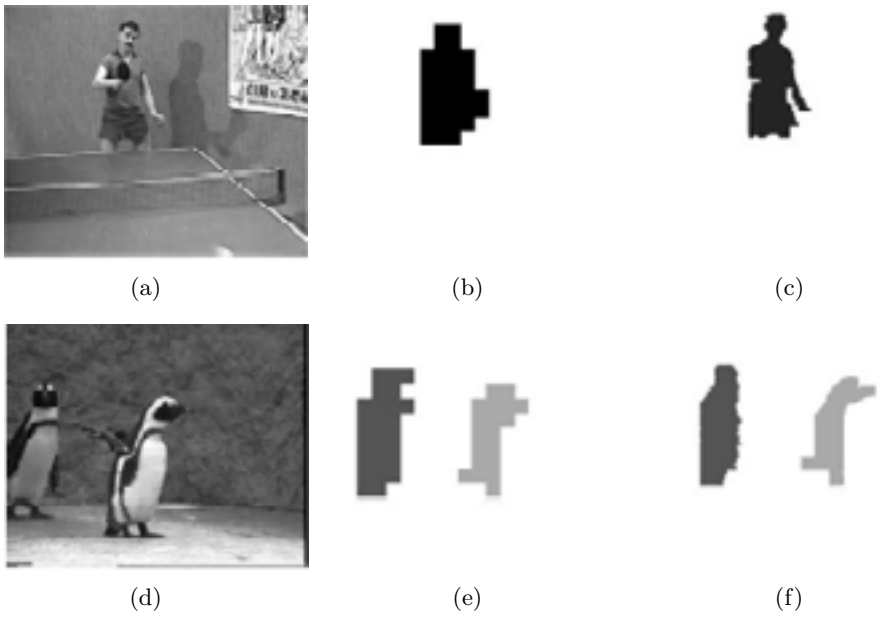


Figure 2.14: The segmentation algorithm presented in [63].

cation of different foreground objects. In order to derive motion information for I frames, averaging of the motion vectors of the P frames that are temporally adjacent to the I frame is performed. Then, in order to segment the background into its constituent objects the use of color information is essential. For this purpose, DC coefficients of the macro-block, corresponding to the Y, Cb, Cr components of the MPEG color space, are extracted.

The proposed algorithm for moving object extraction is based on exploiting the motion information of the macro-blocks. An iterative macro-block rejection is performed in a frame-wise basis to detect macro-blocks with motion vectors deviating from the single rigid plane assumption that are supposed to be part of a foreground object. Then the temporal consistency of the foreground object macro-blocks is examined by temporal tracking of foreground macro-blocks using their motion vectors. Foreground macro-blocks are later clustered to form connected regions and a filter is applied to detect and remove single-macro-block objects. The method exposed in [63] is based on iteratively estimating the global motion model and rejecting those blocks whose motion vectors result larger than average estimation errors. But in this case the underlying assumption is that the background is significantly larger than the area covered by the moving objects, and this can not always be true.

The iterative method to detect foreground macro-blocks is fast and simple, but it is not very reliable. In fact there may be inaccurate estimation of motion vectors from the compressed stream or inability of the motion model to accurately capture the undergoing global motion.



(a) (b) (c)

(d) (e) (f)

Figure 2.15: The results of the method proposed in [63]: in the first row, the original frames; in the second row, the result obtained at DC resolution, and in the third row the results presented after boundary refinement at pixel resolution.

Thus foreground objects at macro-block resolution are obtained. For certain applications this level of resolution may not be sufficient. In this case, pixel-domain processing of a partially decompressed sequence may be required to extract object mask with pixel accuracy. This can be achieved using the color features of pixels in the area of each moving object and a classifier for two-class separation (moving object/background) to reclassify all pixels in that area or a portion of them.

Background segmentation is also performed based on classifying the remaining macro-blocks (assigned to the background) to one of a number of background objects. The background segmentation starts from the first I frame applying the Euclidean distance to the DC color coefficients in the YCbCr color space to identify radically different colors; these indicate the presence of different background spatio-temporal objects. The number of estimated objects obtained and their corresponding colors are used to start the clustering process. The background segmentation is obtained thanks to the k-means algorithm, where k is set to the number of objects calculated in the previous step. In the case of P frames, since color components are not available temporal tracking is performed.

In Figure 2.15 some results are presented; in a) and d) the original frames are

shown and in b) and e) we can see the extracted objects at DC spatial resolution. In this case object boundaries are not clear and precise, but they can again be refined with a pixel domain process (see Figure 2.15 c) and f)).

2.8 Conclusions

In conclusion to this chapter we have presented different approaches to the problem of object extraction from a video sequence; we have started from first approaches to video segmentation, performed frame by frame (intraframe segmentation) to more complex algorithms which require to carry out spatial segmentation and optical flow estimation. Afterwards these algorithms, because of their computational complexity, were substituted with faster, but less precise, methods able to directly extract from compressed stream all color and motion information to perform object segmentation. In the last years, the obtained algorithms constitute a good compromise between computational complexity and quality of the results, reading to reliable object segmentation in real-time with a good level of accuracy. We also have seen that despite the opportunity to use motion and low resolution information from compressed stream, a precise object segmentation requires decoding of video frames to fulfill pixel-wise analysis. In this case, the complexity of methods grows very quickly and a real-time object segmentation is not possible. We also stated that several authors propose a coarse object segmentation, which to our mind is a promising issue for video indexing.

Thus, remaining in the indexing context, we will introduce in the next chapter our object extraction method and a new associated paradigm we call *"Rough indexing"*.

Chapter 3

Object-based segmentation of video streams for 'Rough Indexing'

3.1 Introduction

As we have seen in the previous chapter, the state of the art research in object-based analysis of video content is very rich, starting from very fine pixel-wise methods up to analysis at a coarse resolution in a real-time context. Our work is indeed devoted to the development of latter methods. Here we make an attempt non only to develop 'yet another method', but to introduce a new concept in a general video indexing domain. We call it 'rough indexing'. This chapter is organized as follows: in section 3.2 we will introduce this paradigm and in the next sections we will describe an object extraction method developed in the context of this paradigm operating on MPEG-compressed video streams.

Thus in section 3.3 the basic philosophy of the work is proposed. The first step is to extract from P frames, by using a robust camera motion estimation algorithm (section 3.4), the regions that present a motion model that does not follow the camera motion. In sections 3.5 and 3.6 the way to extract this motion information and build the object masks is presented. Then a morphological color segmentation algorithm is performed on I frames to refine the result of the mask segmentation (section 3.7). Finally in section 3.8 we will explain how the results of motion segmentation are combined with rough low-resolution color segmentation of I frames to refine object shape and capture meaningful objects at I frame temporal resolution.

3.2 Rough indexing paradigm

Recently, as also presented in chapter 2, a new trend in analysis methods for indexing multimedia content has appeared which can be qualified as 'rough indexing' paradigm. Many authors [64] [62] [63] (see also section 2.7) are interested in fast and approximate analysis of multimedia content at poor (or intentionally degraded) [64] resolution. Coded multimedia streams give a rich background for development of these methods, as low resolution data can be easily extracted from MPEG compressed streams without complete decoding. Thus many authors dealt with extraction of moving foreground objects from MPEG2 compressed video with still background [65], while numerous works [66] have been devoted to the estimation of global camera model from compressed video. Thus the 'rough data' - that is noisy motion vectors and DC images - have been used for fine indexing. Our 'rough indexing' paradigm can be expressed as 'the most complete model' on rough data and at rough resolution (both spatial and temporal).

Due to noisiness of input data and missing information, 'rough indexing' does not aim at a full recovering of objects in video. The idea here is to extract them at most salient time moments, that is when object differs very much from the background scene. The detection results will be used to build a rough spatio-temporal model of objects. This model will capture the most salient features of the object: its trajectory, its surface with regards to the frame surface, its dominant color. The lack of detection can be compensated by the model, which will help to interpolate object location along the time.

In some sense, 'rough indexing' is the concept adopted to a fast browsing of the video content, when the attention is focused only on the most salient features of video scenes.

In this paradigm we follow a spatio-temporal approach combining both motion information - the complete first order camera motion descriptor of MPEG7 standard - and region-based color segmentation to extract meaningful objects from compressed video with arbitrary camera and object motion.

3.3 Methodology for foreground object extraction

Figure 3.1 displays the global scheme of our approach. Our method is composed by a motion analysis of the sequence and color segmentation of the I frames. With respect to the motion analysis, the first step is camera motion estimation of P frames

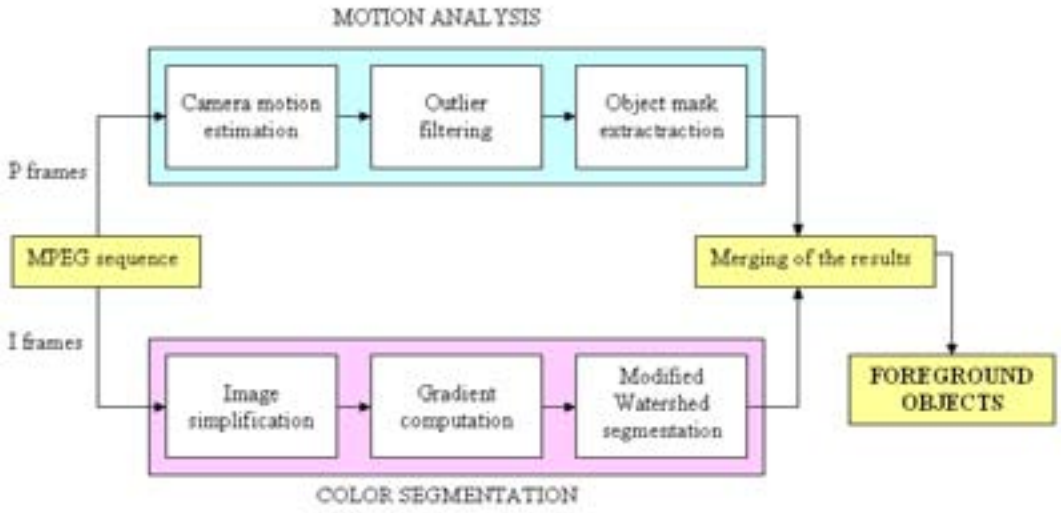


Figure 3.1: Global scheme of the proposed system.

using a robust estimator. In this first step we can separate all the 'outlier blocks' that is all the macro-blocks which do not follow the estimated camera motion. These blocks are not only part of the foreground objects but they are also due to noise, camera motion and 'flat zones', that is zones with very low gradient; so a filtering is necessary to separate the macro-blocks part of the the foreground objects from the others. So second step of our motion analysis consists in filtering of outliers. Then, the third step consists in the extraction, in correspondence of I frames, of the so called 'object masks' where we suppose that foreground objects can be located with high probability.

As the result of motion analysis can be quite approximative a color segmentation process is necessary to refine the motion results. So, from the I frame, we extract color information, applying a color segmentation algorithm to the DCT coefficients of the I frame to subdivide the image into homogeneous color regions. As we work with reduced spatial resolution, a preprocessing step is necessary to make the DC images more homogeneous. After that we can compute the gradient and apply a modified Watershed segmentation.

Once obtained, color and motion information are merged together in correspondence to I frames to extract the foreground objects.

3.4 Motion mask extraction

The idea here is to extract foreground blocks which do not follow global camera motion in each P frame and then to apply a 3D segmentation on a whole GOP in order to extract a 3D 'motion shape'. So, first of all, the camera motion is extracted and, using this information, foreground blocks are separated from background blocks which usually follows the camera motion, then 3D segmentation is performed and from the cross-section of this 3D motion shape the object masks related to each P frame are extracted. Starting from the obtained masks, foreground moving regions can be extracted for I frames for which we have no motion information in the MPEG2 flow. In the following sections all these steps are going to be described in detail.

3.4.1 Camera motion estimation

In order to detect 'foreground blocks' which do not follow the global camera motion, we have to estimate this motion first. Here we consider a parametric affine motion model with six parameters as admissible in MPEG7 'parametric motion' descriptor. It is defined as follows:

$$\begin{cases} dx_i = a_1 + a_2 x_i + a_3 y_i \\ dy_i = a_4 + a_5 x_i + a_6 y_i \end{cases} \tag{3.1}$$

where (x_i, y_i) is the position of the i^{th} macro-block center in the current image and $(dx_i, dy_i)^T$ is the motion vector pointing from the current position to corresponding macro-block in the previous image, as here only P frames are used for motion estimation. The obtained estimation vector is $\theta = (a_1, a_2, a_3, a_4, a_5, a_6)^T$ where a_i parameters allow to model different camera movements (pan, tilt, zoom, rotation). If the camera is static, all the parameters are supposed to be zero, but in reality, due to noise, they are not strictly equal to zero. To estimate the camera motion parameters from an MPEG2 macro-block resolution optical flow we have used a robust weighted least-square estimator [67], taking the MPEG2 macro-block motion vectors as measures. The robustness of the method is based on a good outlier rejection scheme.

The outlier rejection scheme we implement follows the approach proposed in [67].

First of all we explicitly use the architecture of MPEG2 standard and the particularities of block-based motion estimation block of the encoder. As we stated in chapter 1, in P frames two ways of encoding macro-block are used. For macro-blocks,

for which the quality Q of motion compensation is considered acceptable, a motion vector is encoded. In block-based estimators, MAD is one of the employed criteria, that is:

$$MAD(B^T) = \sum_{(x,y)\in B^T} |I(x,y,t) - I(x+dx, y+dy, t-dt)| \tag{3.2}$$

Here B^T denotes a block in video P frame at current moment of time t and $t-dt$ is a reference moment of time. The mean square error (MSE) is also very much popular:

$$MSE(B^T) = \frac{1}{N \times M} \sum_{(x,y)\in B^T} (I(x,y,t) - I(x+dx, y+dy, t-dt))^T \tag{3.3}$$

where $N \times M$ stands for the block size. Finally the PSNR criterion can also be met:

$$PSNR(B^T) = 10 \log_{10} \frac{MAX^2}{MSE} \tag{3.4}$$

where MAX is the maximum pixel range value. Mathematically in block-based motion estimations the acceptability of motion compensation quality is expressed by the inequality:

$$Q(B^T) < \text{Th.} \tag{3.5}$$

Here Th. is a threshold for the chosen criterion.

In case the condition 3.5 does not hold, the encoder selects intracode for macro-block encoding. In this case the original luminance and color signal is intra-frame encoded by blocks of 8×8 pixels. A specific flag is then set in the coded stream. Thus the intracoded macro-blocks have to be rejected as their motion is surely ill-estimated.

The macro-blocks situated on the borders of video frames represent the second source of outliers. Indeed, if there is a weak camera motion such as zoom-out or pan or tilt, then the blocks on the frame borders do not have matching areas in reference frame. In this case estimated motion vectors are erroneous.

Another source of outliers in video frames are moving objects animated with proper motion. These outliers represent a source of information for object extraction and thus should be properly labelled. Such a labelling of outliers is an inherent part of so-called robust motion estimators. These estimators, built with robust statistical methods were in the focus of recent research in motion estimation. Their principle

is based on limited influence functions, which are derivatives of a cost function to be minimized.

The general principle of this estimators consists in the following. Given a set of measures $D = d(x, y)$ and the model $\theta = (\theta_1, \ldots, \theta_n)$ the problem consists in estimating the model parameters $\hat{\theta}$ such that the function $\rho(r)$ of residuals

$$r = d(x, y) - d_{\hat{\theta}}(x, y) \tag{3.6}$$

is minimal. Here $d_{\hat{\theta}}(x, y)$ demotes a measure conform to a model. If we take a usual least-square estimator $\rho(r) = r^2$, then random distribution of r will make the estimator instable. The stability, or influence of errors, is characterized by the derivative of the estimator $\rho'(r)$. In the case of least square estimator:

$$\rho'(r) = r \tag{3.7}$$

It is therefore unlimited. Thus the family of robust estimators with limited derivatives has been proposed. In [68] and [69] various estimation methods are presenting starting from simple ones to more complex criteria. In the context of motion estimation with OFE, Bouthemy and Odobez [70] proposed to use Tuckey estimator.

The first step consists in explicit outlier rejection of macro-blocks on the borders, where MPEG2 motion compensation method does not give good results, and of intracoded macro-blocks; the second step is based on the use of the Tuckey bi-weight estimator [71] which uses the cost function $\rho(r)$ instead of classical cost function $\rho(r) = r^2$, where r is the residual between the measured values and those obtained by the model. The Tuckey's bi-weight function $\rho(r, C)$ and its derivative $\psi(r, C)$ are defined as follows:

$$\rho(r, C) = \begin{cases} \frac{r^6}{6} - 2\frac{C^2 r^4}{4} + \frac{C^4 r^2}{2} & \text{if } |r| < C \\ \frac{C^6}{6} & \text{otherwise} \end{cases} \tag{3.8}$$

$$\psi(r, C) = \begin{cases} r(r^2 - C^2)^2 & \text{if } |r| < C \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

where C is a constant value.

In [67], Durik and Benois follow the method [71] and pose the problem of motion estimation as a weighted least square method. The problem here is to minimize the sum of estimator values, which can be expressed as:

$$\sum_{r_i=d_i-d_{\theta_i}} \rho(r_i) = \frac{1}{2}\sum w_i r_i^2 \tag{3.10}$$

Here d_i and d_{θ_i} represent the i^{th} measure from input data and conform to the underlaid model.

Deriving (3.10) according to each parameter a_j with $j = 1, \ldots, 6$, we will have

$$\sum_i \psi(r_i)\frac{\partial r_i}{\partial a_j} = \sum w_i r_i \frac{\partial r_i}{\partial a_j} = 0 \tag{3.11}$$

Here $\psi(r_i)$ is the derivative of the estimator ρ, that is $\psi(r, C)$ in our case (3.8). Then from (3.11) we will have:

$$w_i = \frac{psi(r_i)}{r_i} \tag{3.12}$$

Finally from (3.8) and (3.12):

$$w_i = \begin{cases} (r^2 - C^2)^2 & \text{if } |r| < C \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

Bouthemy and Odobez [70] consider as residuals the residual in the OFE:

$$r = \left(\nabla I_x \frac{\partial u}{\partial x} + \nabla I_y \frac{\partial u}{\partial y} + \nabla I_t\right) \tag{3.14}$$

In this case r is a scalar value. In [67], the input measures for the model estimators are residual vectors: $\vec{r}(x, y) = (\vec{dx} - \vec{d_\theta x}, \vec{dy} - \vec{d_\theta y})$. Nevertheless, according to (3.13), only squared norm of vectors con be used. Therefore, this estimation scheme is consistent.

The authors of [67] propose an x, y rejection scheme with an independent choice of the constants C_{dx} and C_{dy}. Let us consider the square norm:

$$\|\vec{r}\|^2 = r_{dx}^2 + r_{dy}^2 \tag{3.15}$$

in (3.13) with $r_{dx} = dx - d_\theta x$ and $r_{dy} = dy - d_\theta y$. We then loose the information about which coordinate of the input motion vector yields strong errors with regards to the model. Therefore, the equation (3.13) is considered in a vector form, with two independent constants C_x and C_y as follows:

$$\vec{w}_i = \begin{pmatrix} w_i dx \\ w_i dy \end{pmatrix} = \begin{cases} (r_{dx}^2 - C_{dx}^2)^2 & \text{if } |r_{dx}| < C_{dx} \\ 0 & \text{otherwise} \\ (r_{dy}^2 - C_{dy}^2)^2 & \text{if } |r_{dy}| < C_{dy} \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

Furthermore, all weights are normalized in the interval $[0, 1]$ such as:

$$w'_{i_{dx}} = \frac{w_{i_{dx}}}{C_{dx}^4} \quad w'_{i_{dy}} = \frac{w_{i_{dy}}}{C_{dy}^4} \tag{3.17}$$

In the following we will always use normalize weights, thus we will omit $'$ in the notation.

The authors of [67] propose a close form solution for the weighted least square scheme, such that:

$$\hat{\theta} = (H^T W H)^{-1} H^T W Y \tag{3.18}$$

Here H is the *observation matrix*

$$H = \begin{pmatrix} 1 & x_i & y_i & 0 & 0 & 0 \\ \vdots & & & & & \vdots \\ 1 & x_N & y_N & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_i & y_i \\ \vdots & & & & & \vdots \\ 0 & 0 & 0 & 1 & x_N & y_N \end{pmatrix} \tag{3.19}$$

with macro-block centers conform to the affine motion model:

$$\begin{aligned} dx = &\quad a_0 + a_1 x + a_2 y \\ dy = &\quad a_3 + a_4 x + a_5 y \end{aligned} \tag{3.20}$$

where Y is the vector of measures, that is of N coordinates of macro-block motion vectors

$$Y = (dx_1, \ldots, dx_N, dy_1, \ldots, dy_N) \tag{3.21}$$

and W is the weight matrix:

$$W = \begin{pmatrix} w_{1dx} & 0 & \cdots & & 0 \\ 0 & \ddots & & & 0 \\ \vdots & & w_{Ndx} & & \vdots \\ & & & w_{1dy} & \\ & & & & 0 \\ 0 & & \cdots & 0 & w_{Ndy} \end{pmatrix} \tag{3.22}$$

The authors of [67] propose a multi-resolution estimation scheme, which allows to tune the constants C_{dx} and C_{dy} between resolution levels proportionally to standard deviation of residuals σ_{dx} and σ_{dy}. Thus, for a given resolution level l, the constants C are calculated as:

$$C_l = \frac{(k \cdot \sigma_l + C_{l-1})}{2} \qquad l \geq 3 \tag{3.23}$$

For the lowest resolution level all measures are accepted, that is $c_1 = \infty$, for the second layer $C_2 = k \cdot \sigma_2$.

In Figure 3.2 an example of camera motion extraction is shown; in Figure 3.2 a) a P frame and in 3.2 b) the corresponding MPEG2 motion vectors are presented; in Figure 3.2 c) the camera motion vectors extracted with the algorithm explained are shown.

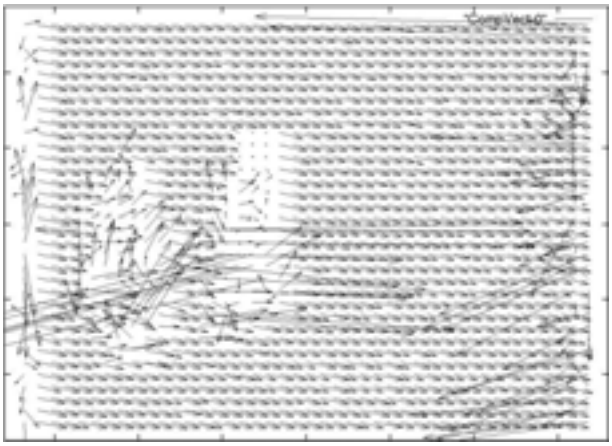
3.5 Motion mask extraction from a single P frame

Once the estimation of camera motion model is fulfilled, the problem of object extraction can be formulated as the separation of the macro-blocks with motion irrelevant to the estimated model. Hence macro-blocks belonging to objects endowed with own motion can be detected. They are characterized by weights equal or close to zero according to the motion estimation method we presented above. The object extraction step is organized in three steps:

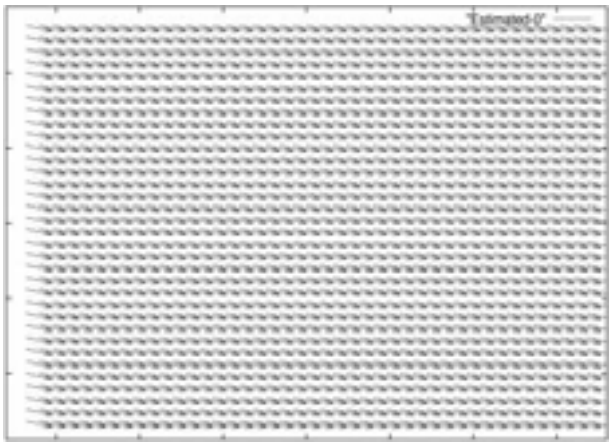
- conversion of the estimated model weights into a grey level image and thresholding;
- camera motion filtering;
- 3D segmentation.

The aim of the first step is to convert the measure of weights extracted from the camera motion estimation into a grey level image and to apply a threshold to distinguish low weights caused by outliers from high weights probably caused by noise effects. The second step filters the outliers due to camera motion and then the third step applies a 3D morphological segmentation to filter the image along the time axis. In the following paragraphs these steps will be described in details.

(a)



(b)



(c)

Figure 3.2: En example of camera motion extraction from a P frame.

(a) (b) (c)

Figure 3.3: Creation of a grey level map and of macro-block weights and thresholding operation to remove noise.

Creation of a grey level image

The weights extracted from camera motion estimation are used to separate macro-blocks that are part of the background from macro-blocks that are supposed to be part of a foreground object. The number of weight values may not correspond exactly to the total macro-block number. As we presented in our explicit first step outlier rejection scheme, a P frame, even if it is coded in inter-frame mode, may have some macro-blocks coded in intra-frame mode when the encoder is not able to attain satisfactory quality of motion compensation; in this case, for the intra-coded macro-blocks, no weight values are produced. But these intra-coded macro-blocks can also be included in the set of "irrelevant" measures as they correspond to the failure of MPEG block-based motion estimator: so we can assign them a value of weight w_i close to 0 because we can suppose that the motion vectors of these macro-blocks do not follow the camera model. Let us consider a grey-level image $I_{x,y}$ of resolution N/MacroBlockSize*M/MacroBlockSize defined as follows:

$$I_{x,y} = [(1 - \max(w_{dx}, w_{dy})) \cdot 255] \tag{3.24}$$

Here the brighter pixels correspond to macro-blocks with low weights and thus would belong to the objects that do not follow the global movement. Pixels of $I_{x,y}$ with low values correspond to low weights scattered throughout the P frame due to texture and local motion deformation.

In Figure 3.3 the grey level map (3.3 b) obtained from a P frame (3.3 a) is shown; it is possible to see that foreground objects are not clear because of the noise due to motion vectors errors. Thus in order to get relevant pixels well representing objects

(a) (b) (c)

Figure 3.4: Extraction of a binary image from a P frame.

with proper motion, a binary image $I^b_{x,y}$ will be now computed by thresholding:

$$I^b_{x,y} = \begin{cases} 1 & \text{if } I^b_{x,y} < s \\ 0 & \text{otherwise} \end{cases} \tag{3.25}$$

The threshold s is based on typical "low value" of weights and was tested on various sequences in the range [7,10]. In fact a threshold lower than 7 still leaves a lot of isolated macro-blocks that are probably due to noisy motion vectors, while a threshold higher than 10 can delete macro-blocks that are part of some connected region (so probably part of an object).

The result is a binary image as in Figure 3.4 where the relation between the motion vectors and the region where the objects are located is shown. Figure 3.4 a) shows a P frame taken from a video sequence where the objects of interest are constituted by two women walking tracked by the camera whereas Figure 3.4 b) indicates its associated motion vectors. It is possible to see in the middle of the motion vector field two regions with motion vectors completely different from the others due to the object presence and in an area on the right border some other "outliers" due in this case to camera motion (a right pan in this case). In Figure 3.4 c) the obtained binary image $I^b_{x,y}$ is shown.

3.5.1 Filtering of outliers due to camera motion

As we can see from 3.4 some outliers appear in the binary image in the opposite direction of camera movement. These outliers are due to new macro-blocks entered in the frame. In fact, the pixels of original video frame in these macro-blocks do not have their antecedent in the reference frame. Therefore, motion vectors resulting

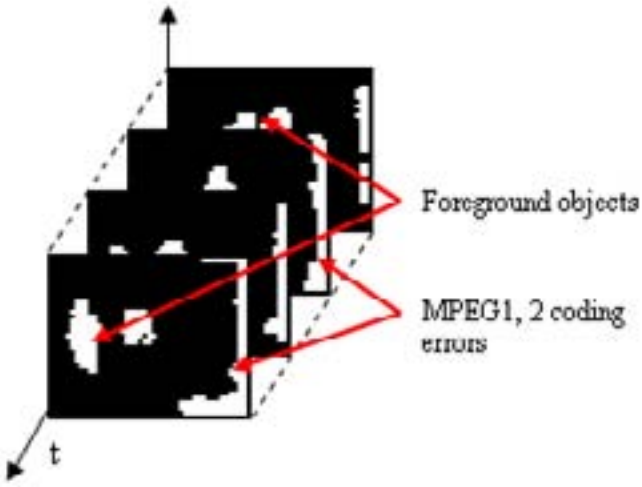


Figure 3.5: Outliers due to camera motion in a video sequence.

from the MPEG2 decoding process are erroneous and do not follow the camera motion in most cases. In this case we have high irrelevance weight on these zones even if no object is present (see Figure 3.3 c), 3.4 c) and Figure 3.5).

In order to filter the outliers on the frame border the estimated camera motion model is used (see Figure 3.6). In fact with forward prediction motion coding, the displacement vector $\overline{d} = (dx, dy)^T$ of a macro-block in the current frame is related to the coordinates of pixel $(x_c, y_c)^T$ in a current frame and its reference pixel $(x_p, y_p)^T$ in the reference frame as follows:

$$\begin{cases} dx = x_p - x_c \\ dy = y_p - y_c \end{cases} \tag{3.26}$$

Now using the model (3.1) we can solve it for x_c and y_c taking as reference pixels the corners of the reference frame. Thus the reference frame result to be warped to the current frame. Thus we can obtain the geometry of the zone entered in the frame: if some "outliers" are present in that zone we can suppose that they are due to camera motion and we do not consider them when searching for object masks.

Repeating the method described above for all P frames inside a single video shot we obtain motion masks for foreground objects in the shot. This method requires preliminary segmentation of video content into shots, since at shot boundaries, MPEG

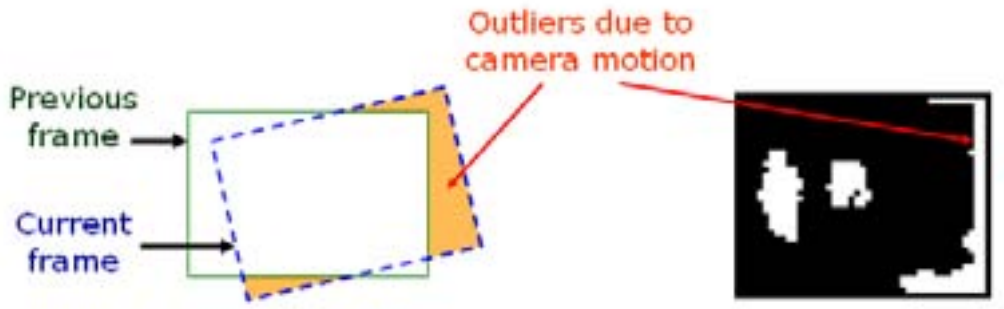


Figure 3.6: High weight values caused by outliers on the frame borders.

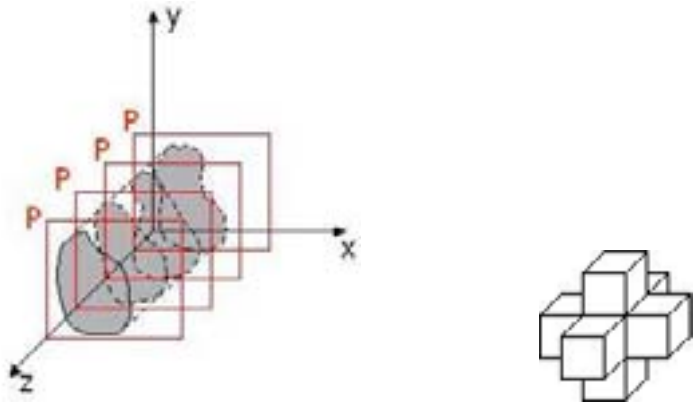
motion vectors are likely to be massively erroneous and the estimated camera motion does not correspond to the reality.

We have described how to obtain a first guess of objects at reduced temporal resolution which constitutes our rough indexing paradigm. Nevertheless masks in each pair of P frames were obtained independently from each other. This is why they remain noisy in time.

3.5.2 3D filtering of the motion masks

In order to improve the detection we take advantage of the temporal coherence of moving objects and thus smooth the detection along the time. To do this we model a video segmentation as a conjunction of 3D volumes in (x, y, t) space. Here, the indexing function of the various objects $f(x, y, t)$ is assumed to be known for values of t corresponding to P frames. Let us consider two consecutive GOPs from an MPEG2 stream both belonging to the same shot. To smooth $f(x, y, t)$ along the time we apply a 3D segmentation algorithm to such pairs of GOPs (see Figure 3.7 a)). The result of this segmentation is a 3D volumetric mask that highlights the region inside where a foreground object is probably located and moves. In this work we use a 3D morphological segmentation algorithm developed in [72]. The algorithm follows a usual morphological scheme, applying these operations in the following order:

- a 3D morphological filtering;
- a modified gradient computation;



(a) (b)

Figure 3.7: a) 3D segmentation along the volume composed by P frames; b) the 6-connected element used for morphological filtering and segmentation.

- a region growing by watershed in a 3D space.

In all applied morphological operations, the 3D 6-connected structuring element as shown in Figure 3.7b) has been used.

Filtering of the noise

Although we apply a threshold on each image, we notice that in some parts of the volume there are still some isolated macro-blocks that, even if not following the global motion, don't represent any foreground objects. So we filter the volume to eliminate these isolated blocks using a 8-connected structuring element to investigate macro-blocks: if we find a white macro-block that has all his neighbors of black color we turn it into a black one (see Figure 3.8). In the same way, if we find a black macro-block with all white neighbors, we turn it into a white one. This allows to eliminate isolated noise whereas it avoids holes in the foreground objects.

Computation of the modified gradient

Once the noise and the outliers due to camera movement have been filtered out, we calculate a modified morphological gradient. The morphologic gradient is defined in the literature as the difference between the pixel dilation applied on the image and its erosion, i.e:

$$G_{\delta\epsilon}(x) = \delta(x) - \epsilon(x) \tag{3.27}$$

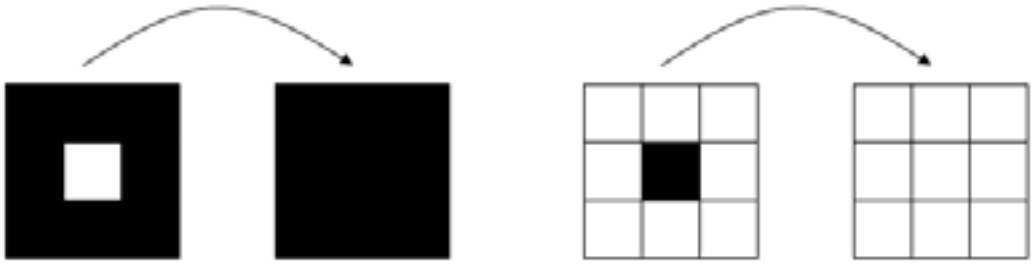


Figure 3.8: The filter used to remove noisy isolated macro-blocks.

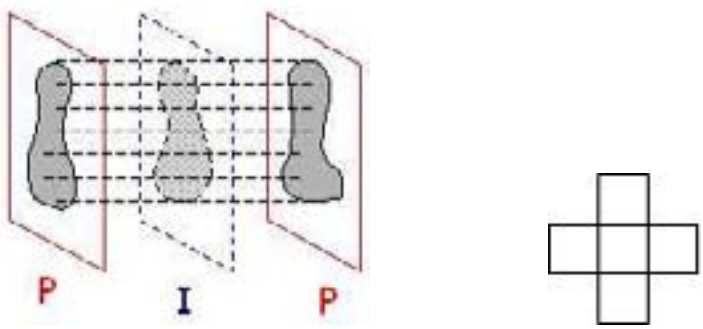
where the dilation $\delta(x)$ is the maximum value of difference between a pixel and its neighbors while the erosion $\epsilon(x)$ is the minimum difference. In our work, we have used a modified gradient, as explained in the following: after having applied the threshold and after the noise filtering process, we obtain binary images, where background pixels present zero value and the foreground ones have been assigned a 255 value. In this case all pixels inside the objects have the same dilation and erosion values since both the maximum value ($\delta(x)$) and the minimum one ($\epsilon(x)$) are 255. For the pixels on the boundaries instead, the maximum value is represented by the pixels in the internal part of the object (255) while the minimum is given by the pixels outside the baundaries (0 value). So we decide to calculate the gradient as the maximum difference between a pixel and its neighbors:

$$G(x) = \max_i |I_{xy} - I_{x_i y_i}| \qquad i = 1, \ldots, 6 \tag{3.28}$$

where (x_i, y_i) are the coordinate of the pixel neighbors.

Region growing algorithm

Once we have calculated the gradient volume we search for the connected 3D regions inside the volume. For this purpose we employ a 3D region growing algorithm. Foreground objects usually exhibit low internal gradient and high gradient values on their boundaries. Due to the fact that a foreground object doesn't follow the global movement, it is usually characterized by low weight values in the macro-block matrix of camera movements because these blocks do not contribute in camera motion estimation (see section 3.4.1). So the strategy foresees to start the region growing algorithm from the regions with both low gradient value and low weights



(a) (b)

Figure 3.9: a) Creation of the mask for the I-frame by interpolation of two P-frames; b) the 4-connected structuring element used in morphological 2D operation.

and expanding them to reach a given gradient barrier. At the end, we obtain a mask for each P frame of the sequence we have analyzed; This 3D segmentation process allows for smoothing of initial noisy characteristic function of objects. Considering the volume slices $\tilde{I}^b_{x,y}$ we obtain a 2D mask for each P-frame we have analyzed.

3.6 Motion mask extraction in I frame

The extracted motion masks represent a good guess to the object shape in P frames. Nevertheless, the sole motion information is not sufficient for a robust object extraction. Thus we propose to merge the motion masks with the results of color-based intra-frame segmentation of I frames. In the framework of the rough indexing paradigm we only use DC images of original I frames. Since motion masks have been obtained only for P frames, we have to build the corresponding mask for the I frame in order to overlap it to a color-based segmentation. As the MPEG2 decoder does not give motion vectors for the I frame we cannot build the mask starting from MPEG2 motion information directly. Looking to the structure of the MPEG2 compression standard it can be seen that, considering two consecutive groups of pictures (GOP), the I frame is located between two B frames, or, if we consider only P frames, as in the case of this work, between two P frames. Therefore, in order to calculate the mask for the I frame, we can consider the two P frames surrounding the I frame of interest and then to interpolate the two images (3.9 a)).

Usually the interpolation process can be fulfilled by two different approaches, i.e.:

- a motion based approach [73];

Figure 3.10: On the first row a sequence of I frames and on the second row the extracted object masks.

- a spatio-temporal segmentation without use of motion.

For the sake of low computational cost we decide to use a spatial interpolation, using a morphological filter. As a result, the binary mask in I frame $\tilde{I}^b_{x,y}(t)$ is computed as:

$$\tilde{I}^b_{x,y} = \min(\delta\tilde{I}^b_{x,y}(t-1), \delta\tilde{I}^b_{x,y}(t+1)) \tag{3.29}$$

Here δ denotes the morphological dilation with 4-connected structural elements of radius 1 (as shown in Figure 3.9 b)). In this way we obtain the mask for the I frame that exhibits the approximate position of the objects. Figure 3.10 depicts some I frames extracted from a MPEG2 video stream and the corresponding masks obtained using the motion-based approach proposed before.

3.7 Object mask refinement by color segmentation

As we described in the previous section motion masks interpolated for I frames indicate likely the position of objects having their own motion. Now, using the color information of I frames inside the masks, we can refine the object shapes and

estimate their textural and color parameters. Hence, in order to index the video content by spatial features at I frame temporal resolution, we implement a color segmentation process to subdivide the I frame into regions. Then regions belonging to moving objects are selected by overlapping the I frame with the motion mask we have calculated before. The resulting set of the overlapped regions form the objects of interest. Remaining in the rough indexing paradigm framework, we use only DC coefficients of I frames shaped into "DC images" [74], since DC coefficients are easy to extract from the MPEG2 stream without a complete decoding.

We apply a morphological framework for the color segmentation. The implemented segmentation method is organized in these steps:

- first a pre-processing filtering is applied;
- then the gradient of the DC image is extracted to obtain the objects boundaries;
- finally, a region growing algorithm by modified watershed is performed in the YUV space.

3.7.1 Pre-processing filtering

The aim of the filtering is to delete some information in the image that is not useful for segmentation and makes the image too noisy. Our goal is to obtain an image with reduced noise so as to keep only the useful information for segmentation, in order to apply the gradient extraction algorithm. For this purpose we take the three components Y, U and V and first we apply to them a morphologic filter called *partial reconstruction opening filter (OF)* and then a *partial reconstruction closing filter (CF)* using the erosion (ϵ) and dilation (δ) operators. These operations are performed with a 4-connected structuring element as shown in Figure 3.9.

- Erosion

The morphologic erosion of a function f with a structuring element M is defined as:

$$\epsilon(f(x)) = \min\{f(x+k), k \in M\} \tag{3.30}$$

where f is the DC image and M is the 4-connected element. Thus this operator modifies each pixel value as a function of itself and its neighbors identified by the structuring element M.

- *Dilation*

The morphologic dilation of a function f with a structuring element M is defined as:

$$\delta(f(x)) = \max\{f(x-k), k \in M\} \tag{3.31}$$

Composing the two operators it is possible to obtain the following morphologic filters:

- *Morphologic closing*

The morphologic closing is the application of a dilation followed by an erosion with the same structuring element:

$$\phi = \epsilon(\delta) \tag{3.32}$$

- *Morphologic opening*

The morphologic opening is the application of an erosion followed by a dilation with the same structuring element:

$$\gamma = \delta(\epsilon) \tag{3.33}$$

The *partial reconstruction filter* implemented in this work is obtained by applying to the YUV component an opening filter followed by a closing filter. The used opening filter is:

$$\gamma_{m,n} = \delta_m(\epsilon_n(I)) = \underbrace{\delta(\ldots\delta(\delta(\epsilon_n(I))))}_{m} \tag{3.34}$$

where ϵ_n means that the operator ϵ is applied n times

In our case the choice of $n = 2$ and $m = 3$, means that we apply three times a dilation filter on the twice eroded image. In Figure 3.11 we want to show the effects of the opening filter: after having applied the operator described in (3.34), we obtain a Y component as shown in Figure 3.11 b); all the small objects that present high luminance values have disappeared because of the effect of the erosion filter, leading to a more homogeneous image.

Starting from the obtained result we then apply a closing filter to delete all isolated blocks that present low luminance value, i.e.:

(a) (b)

Figure 3.11: a) Y component of a DC image; b) the same DC image after the application of the opening filter.

$$\phi_{m,n} = \epsilon_m(\delta_n(I)) = \underbrace{\epsilon(\ldots\epsilon(\epsilon(\delta_n(I))))}_{m} \tag{3.35}$$

3.7.2 Gradient extraction

The extraction of the morphologic gradient is the next step in 2D color segmentation process. We compute the gradient for all the three YUV components of the image we have obtained from the DC image, so obtaining the vector:

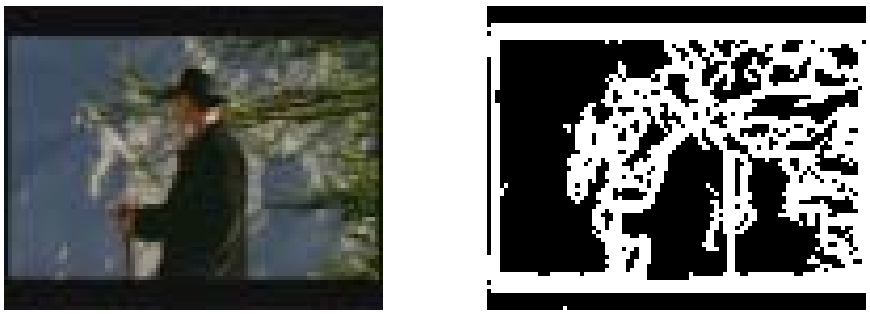
$$\overline{G_{\delta\epsilon}} = \begin{pmatrix} \delta(Y) - \epsilon(Y) \\ \delta(U) - \epsilon(U) \\ \delta(V) - \epsilon(V) \end{pmatrix} \tag{3.36}$$

After this we build the gradient image using a scalar gradient $G_{\delta\epsilon}$:

$$G_{\delta\epsilon} = \max(\overline{G_{\delta\epsilon}}(Y), \overline{G_{\delta\epsilon}}(U), \overline{G_{\delta\epsilon}}(V)) \tag{3.37}$$

In this image the homogenous regions, typically the regions inside the objects, appear with low values of gradient because their neighboring pixels have close values, while boundary pixels of a region have high gradient values because since luminance and chrominance values of pixels across boundaries have large differences. Morphologic filters like opening and closing filters (presented in equations (3.34) and (3.35)) lead to more homogenous regions so that gradient information becomes more reliable for the objective at hand.

The next step is to discard all gradient values below a given threshold so as to



(a) (b)

Figure 3.12: a) An I frame at DC resolution and b) the gradient image $G_{\delta\epsilon}$ with a threshold of 20.

get a binary image, i.e.:

$$I^b(x,y) = \begin{cases} 0 & \text{if } \|G_{x,y}(x,y)\| \leq \text{threshold} \\ 255 & \text{otherwise} \end{cases} \tag{3.38}$$

In such binary images regions with homogenous color present pixels with zero value inside, whereas boundary pixels are set to white. In Figure 3.12 b) the gradient calculated for the DC image in 3.12 a) is shown. In this case a threshold equal to 20 has been applied to the gradient image. From our tests performed on different video genres we have seen that the best threshold ranges between 15 and 20 for generic video material, while it is usually lower for cartoons (around 10), since they are naturally less textured.

3.7.3 Region growing algorithm

After the threshold has been applied to the morphological gradient, all the black connected regions present in the image are filled and labelled separately. For each region the average color value is calculated. A color region map is obtained with some uncertainty zones corresponding to the zones with high gradient, typically in the neighborhood of objects boundaries. In order to assign these zones to the corresponding connected regions, an iterative region growing algorithm with a region-adaptive threshold is used [75].

The threshold is calculated as a function of the average luminance of the region $\bar{m}$ and of a parameter Δ^i that grows with the iteration i:

$$\text{threshold} = F(\bar{m})g(\Delta^i) \tag{3.39}$$

where $F(\bar{m}) = |\bar{m} - 127| + 128$ and $\Delta^i = \Delta^{i-1} + 0.01$.

The function shows that the threshold depends on the mean grey level of the considered region. This function is derived from the *"function of visual sensitivity"* which indicates that the difference between two grey level values is perceived less when the values are at the extremities of the range.

The function $g(\Delta^i)$ is an incremental term used to progressively relax the thresholds for merging boundary pixels with the adjacent region. The initial value Δ^0 is computed as $1/F(127)$ and it is used to compute the first threshold value; so for regions in the middle of luminance range the threshold corresponds to only one level of luminance difference. In this first step the macro-blocks that differ from the adjacent region by a value smaller than this threshold are included in the region. When no macro-blocks can be added to any region the value of the threshold is recalculated using the incremented parameter Δ^i. The threshold is relaxed until all the uncertainty zones are assigned to a region.

Until now we have only talked of luminance values. But the algorithm we have explained can be also applied to color components. In such case we process the three components (luminance Y, chrominance U and chrominance V). In this case the average color is used, namely:

$$m_j^{COL} = \frac{m_j^Y + m_j^U + m_j^V}{3} \tag{3.40}$$

where m_j^Y, m_j^U and m_j^V are the average values of respectively luminance and chrominance components of the region with label j. The adaptive growing procedure is changed as follows in this case. A pixel is included in region j at the i^{th} iteration, if the following relationship is satisfied:

$$|f^Y(x,y) - m_j^Y| + |f^U(x,y) - m_j^U| + |f^V(x,y) - m_j^V| < \alpha \tag{3.41}$$

where:

$$\alpha = 3 \times F(m_j^{COL})g(\Delta^i) \tag{3.42}$$

and $f^Y(x,y)$, $f^U(x,y)$ and $f^V(x,y)$ are respectively the luminance and chrominance values of the pixel we want to include in the region.

In Figure 3.13 the result of this segmentation process is presented: the DC image

(a) (b)

Figure 3.13: in a) a DC frame and in b) the spatial segmentation obtained. Each region has been represented with its average color.

has been subdivided into homogeneous regions, where each region has been associated the average color.

3.8 Merging of motion masks and color segmentation results

What we have obtained until now is:

- a mask for each P frame of the analyzed video sequence using a 3D segmentation;
- a mask for each I frame by interpolating the results from the P frames;
- a 2D color segmentation for the I frame of the sequence.

Now if we superpose the motion mask to the color segmented image we can see which regions obtained from the segmentation process are included in the mask. Actually, as long as it is possible that a region is not completely included in the mask, we consider that, if:

$$\frac{\text{pixels of a region included in the mask}}{\text{total number of the pixels in the same region}} \geq \text{threshold} \tag{3.43}$$

The threshold is typically set to 80%: this means that the majority of the pixels of the region are included in the mask, the whole region is considered to be part of the foreground objects. In Figure 3.14 the result of a foreground object extraction is presented. In 3.14 a) and b) the original DC image and the corresponding motion

mask are shown. In 3.14 c) the motion mask has been superposed to the DC image. In d) the objects is extracted according to the method exposed in equation (3.43). Note that some regions that are not part of the foreground object are still present. Since the background contain little regions some of them have been included in the motion mask.

(a) (b) (c) (d)

Figure 3.14: a) the original DC image; b) the extracted motion mask; c) merging of the two results and d) the foreground object obtained with this method.

3.8.1 Flat region removal

Once the objects have been extracted, it is usually necessary to apply a gradient-based filtering to delete possible flat zones. A flat zone is a region in the color image characterized by very low color gradient value (for example the sky is a typical flat region). In no-gradient regions, motion estimation typically fails since a lot of macro-blocks share the same color. Thus macro-blocks belonging to a flat region may be associated with motion vectors that do not follow the camera motion so that they are erroneously detected as moving objects.

In Figure 3.15 a) an I frame with a flat zone is presented. In this case the flat zone is represented by the sky. As it is shown in Figure 3.15 b) the presence of constant color zones can lead to motion vectors with incoherent direction and magnitudes. Consequently the motion mask may incorporate them even when no objects are present (3.15 c)). After the superposition of the motion mask with a color based segmentation of the DC image, the flat zone is recognized as a foreground object (3.15 d)). The approach we adopt to remove it consists in computing the average gradient energy $\bar{E}(R)$ within a region R per macro-blocks: (3.44)

$$\bar{E}(R) = \frac{1}{\text{card}(R)} \sum \|\bar{G}(R)\|^2 \tag{3.44}$$

where $card(R)$ represents the number of macro-blocks contained in the region. All

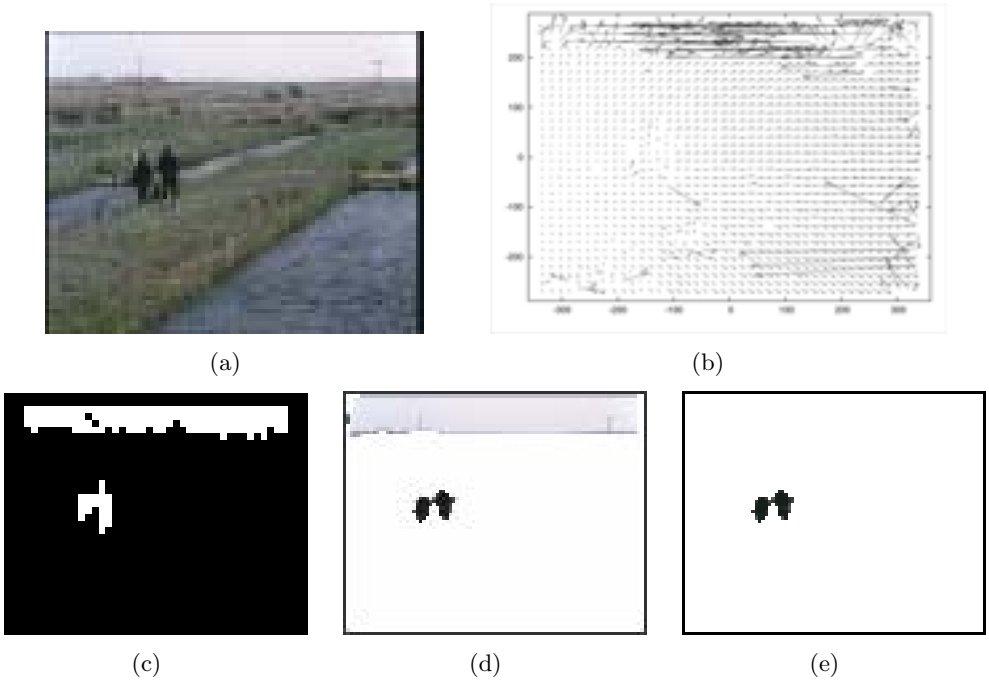


Figure 3.15: a) I frame at DC resolution with a flat zone (the sky); b) the corresponding map of motion vectors; c) the object mask for the frame; d) objects obtained after the superposition of the mask and color the segmented I-frame; e) the final result after flat zone removal.

connected components which have an average gradient energy below a predefined threshold are declared "flat zone" and can thus be excluded. As a result only objects with a heterogeneous color content are preserved. In fact, an object is usually composed of more than one region with constant color and so its mean gradient energy value is higher that the one of a region composed by a single homogeneous region.

The Figure 3.15 e) shows the benefits of the filtering process on the final results. It can be noticed that even if, in this case, the objects of interest present a quite homogeneous color, its gradient values are not as low as the one of a flat region and so they are preserved.

3.9 Conclusion

Thus in this chapter we introduced a new concept, such as 'rough indexing paradigm' and proposed a spatio-temporal foreground object segmentation method. We fol-

lowed the leading edge research methodology by combining both motion information, $2D + t$ information and color information in a combined spatio-temporal segmentation approach. Its key parts were:

- robust estimation of global (camera) affine motion;
- morphological segmentation applied to binary outlier masks and to color frames.

These techniques rather popular for video analysis in uncompressed domain, are very seldom combined for compressed streams with partial decoding. In the 'rough indexing' context we showed how these approaches complete each other in a difficult condition of motion directly recovered from coded streams. Nevertheless, the method is not stable in the sense that it requires the presence of proper motion of objects captured by any industrial encoder. Now to complete object extraction in case of lack of motion, how to index the behavior of objects? This question will find an answer in the next chapter where we will propose a new rough spatio-temporal model of foreground objects.

Chapter 4

Feature extraction and spatio-temporal filtering

4.1 Introduction

In chapter 3 we have explained how we extract foreground objects from a sequence at I frame temporal resolution and DC spatial resolution. It can be useful to extract also some information about the objects, such as for example their approximative shape, trajectory or color, to produce a sort of "rough indexing" of the sequence. Moreover sometimes due to the low resolution of original frames and lack of relative motion of camera and objects, some detection errors may be present in isolated frames. Due to the final goal of our work, such object-based indexing of video content, miss-detection or partial detection of objects is more annoying then over-detection in frames. These kinds of error, specially the miss-detection, can be avoided taking into account the results of the neighbor frames with a sort of filtering of the results along the time.

In this chapter we are going to explain how we have extracted the shape and the trajectory of the foreground objects and we are going to explain how we have realized our spatio-temporal filtering to smooth the detection in time. Taking into account the rough indexing context the core of two approaches is a spatial and spatio-temporal filtering of detection results.

4.2 Extraction of object characteristics

Once foreground objects are obtained, the objective is to describe each object. So for each object we calculate its area, its perimeter and its bounding box, that is the

Figure 4.1: A bounding box of an extracted object.

smallest rectangular box that totally include the object, built using the coordinates (x_{min}, y_{min}) and (x_{max}, y_{max}) as in Figure 4.1.

The bounding box is a very rough approximation of object shape. Nevertheless we cannot use MPEG7 shape descriptors, which is a standard way to index context today. The reason here is that the segmentation results are very noisy. Thus we propose to index the extracted objects by their approximative shape. We use an approximation with an elliptic function. Hence our purpose now is to find the parameters of a generic cartesianly oriented ellipse that can best approximate the contours of the objects in the frame we are considering.

We consider $z(x, y)$, a normalized gaussian function centered in the object barycenter (x_c, y_c) and with variance values set accordingly to object bounding box:

$$z = \exp\left(-\frac{1}{2}\left(\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2}\right)\right) \tag{4.1}$$

with:

$$\mu_x = x_c$$
$$\mu_y = y_c$$

and

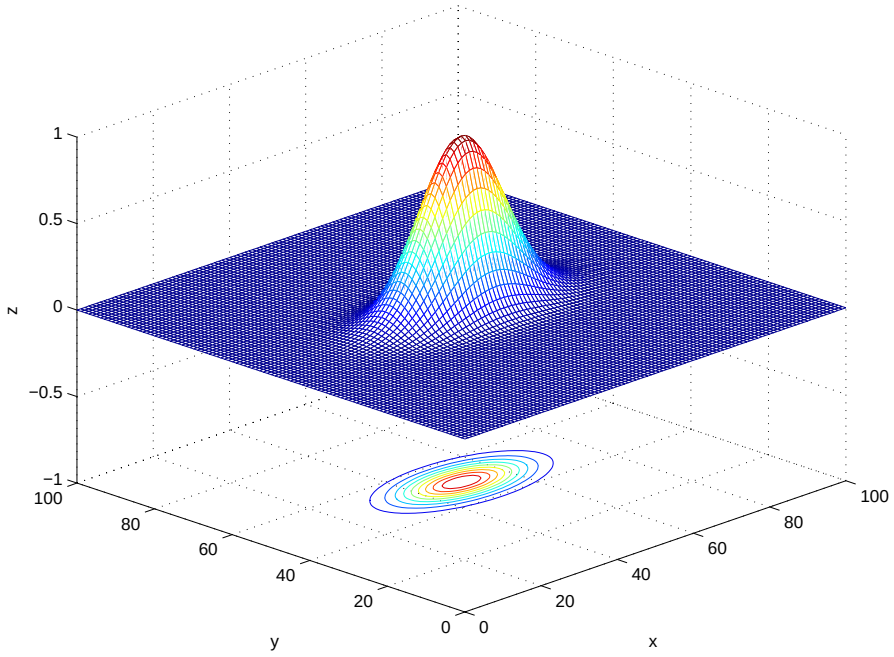


Figure 4.2: A gaussian function and different ellipses obtained cutting the gaussian function along the axis z.

$$\begin{aligned} \sigma_x &= x_c - x_{min} \\ \sigma_y &= y_c - y_{min} \end{aligned}$$

In Figure 4.2 a gaussian function is shown. The values used to build the function are $\mu_x = \mu_y = 50$, $\sigma_x = 10$ and $\sigma_y = 5$. At the bottom of the image the section of the image at different values of axis t are shown.

In fact, by fixing the value of z, the gaussian function provides a set of concentric ellipses with different extent. This operation is equivalent to cut the gaussian function with a plane of equation $z = z_0$. So, to choose the value of z that gives the best approximation of the shape for the selected object we cut the gaussian function in 4.1 for different values of z and we minimize the function:

$$S(z) = \min_z \sum_{(x,y)} \delta(x, y, z) \qquad z \in [0, 1] \tag{4.2}$$

where

$$\delta(x,y) = \begin{cases} 1 & \text{if } (x,y) \in ((\text{ellipse} \cup \text{mask}) - (\text{ellipse} \cap \text{mask})) \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

(a) (b) (c)

Figure 4.3: Object shape approximation with an ellipse.

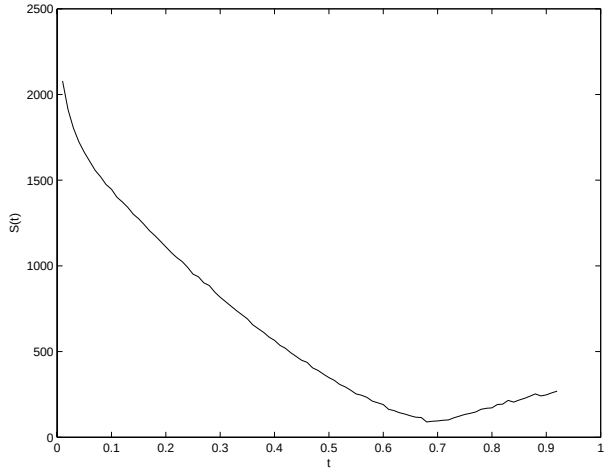


Figure 4.4: An example of the function S(z).

In Figure 4.3 en example of this shape extraction is proposed. In Figure 4.3a) a DC frame is proposed and in b) the foreground object extracted with the algorithm proposed in chapter 3 is shown. In Figure 4.3c) we have approximated the object shape with an ellipse.

In Figure 4.4 the function $S(z)$ for the object in Figure 4.3 is shown.

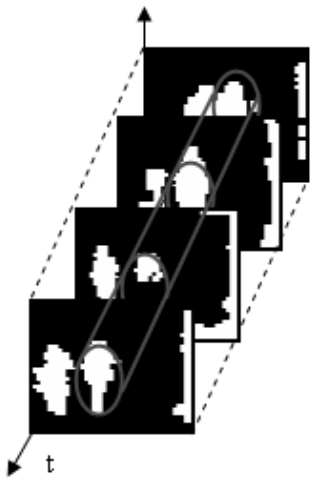


Figure 4.5: Construction of a tube that follows the objects along the sequence.

4.3 Spatio-temporal filtering

Once color and motion information have been merged, foreground objects at I frame temporal resolution are obtained. As I frames have been processed independently one from the others, no information about object variations in time is provided. Furthermore it may happen that if the object movement does not differ a lot from camera motion or the object is still, no moving objects are detected. But, as we can suppose that an object cannot reasonably appear and disappear along a short sequence of frames, we can filter the sequence of the objects at I frame temporal resolution in the direction of time and try to find it even if the motion based detection failed. The objective here is to build the object trajectory along the sequence of I frames where it was detected and then to approximate the shape with a conic function in those frames where objects have not been detected. It means building a sort of tube where each section provides the object position in the frame along the time (see Figure 4.5).

In fact, it is difficult to avoid annoying effects such as blinking of segmentation [76], especially when low resolution measures are used as in our work. It means that some part of objects can disappear or that some objects fail to be detected for several I frames. We can see for example as in Figure 4.5 that in the second frame the bottom half of the object is missing because of a segmentation error.

When building conic tubes for objects, we remain in the rough indexing paradigm.

In any case we will not seek for an exact matching of extracted object borders to the tube. On the contrary, the tube sections at each moment of time will approximate the object shape and can be used in correspondence to the I frames to become part of the whole segmentation process. Thus the object shape can be extracted by merging of color and the cross-section. The visualization of the tube along the time gives an idea of the temporal behavior of objects.

As a natural video sequence can contain several objects sufficiently closed one to an other, the preliminary step for tube construction consists in the identification of the scene object from the detected masks in the sequence at I frame resolution. The whole scheme of spatial-temporal filtering can be described as follows:

- identification of the same object in the sequence of object masks from the I frames;
- spatio-temporal tube construction;
- merging of cross section of the tube and color-based segmentation of I frames with poor detection.

In the following sections we are going to explain each one of these steps in details.

4.3.1 Identification of objects at I frame resolution along the time

The objective here is to track the objects extracted independently in each I frame (see Figure 4.6); that means to follow each object along the sequence. To do this, we estimate motion of each object detected and forward project the object mask from the I frame at the time t to the I frame at $t + \Delta t$. Then if the result of projection overlaps with the result of the detection in I frame at time $t + \Delta t$, the object is identified for the considered pair of frames.

To realize the object projection, first the object motion has to be estimated. Given the context, a global, "rough", object motion is considered with a rigid object assumption. Therefore we suppose that the motion of each object O_k can be sufficiently well described by the affine model for a pair of I frames at t and $t + \Delta t$:

$$\begin{aligned} dx_i &= a_1 + a_2 x_i + a_3 y_i \\ dy_i &= a_4 + a_5 x_i + a_6 y_i \end{aligned} \tag{4.4}$$

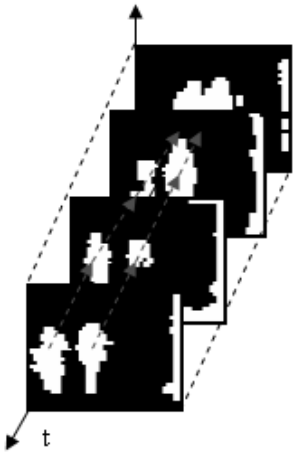


Figure 4.6: Projection of object masks along the sequence.

Basically we work with the same hypothesis as for the camera model estimation we presented in chapter 2. The observations $(dx, dy)_i$ are motion vectors of MPEG macro-blocks which are covered by extracted object mask. The difference from the approach we used for camera model estimation is that for the objects there is no use to apply robust motion scheme with outliers rejections. Macro-blocks in object are very few, therefore we will use all these motion vectors and apply classical least square scheme we will briefly resume below.

The object motion vectors $\bar{d}_{i,k}$ will be used by a least square estimator [77] as an initial guess to estimate the global object motion model θ_k.

Here we describe the least-square method [77] for affine model parameters estimation. Suppose to have a function $y(t)$:

$$y(t) = a_1 y_1(t) + a_2 y_2(t) + \ldots + a_n y_n(t) \tag{4.5}$$

where y_i are different measures effectuated and $y(t)$ is a linear combination of these measures weighted by a_i.

If a vector notation is used we can write the equation (4.5) as follow:

$$\mathcal{Y}(t) = \mathcal{H}^T(t) \cdot \theta \tag{4.6}$$

where $\mathcal{H}(t)$ represents the measure matrix while θ represents the parameters vector. The solution of the system represented in (4.6) is given by:

$$\theta = \mathcal{H}^{-1}(t)\mathcal{Y}(t) \tag{4.7}$$

if $\mathcal{H}$ is invertible. But, in general, the output is affected by a noise $\mathcal{V}(t)$ so that (4.6) has to be written as:

$$\mathcal{Y}(t) = \mathcal{H}(t) \cdot \theta + \mathcal{V}(t) \tag{4.8}$$

so the problem is to estimate the parameters of the system function starting from noisy observations of the output. This means that the solution system (4.7) becomes:

$$\theta = \mathcal{H}^{-1}(t)\mathcal{Y}(t) - \mathcal{H}^{-1}(t)\mathcal{V}(t) \tag{4.9}$$

The real vector θ cannot be estimated because the noise value is unknown so the problem of determining the parameter vector becomes a problem of estimation of the vector $\tilde{\theta}$ that can constitutes the best approximation of θ following a given optimization criterion. Usually the used criterion is the least square one, namely:

$$\mathcal{J}(\theta) = \sum(\text{single errors})^2 \tag{4.10}$$

If $\epsilon(t+i)$ is the error present on the observation $y(t+i)$ and if we consider N measures on the function y, the criterion $\mathcal{J}(\theta)$:

$$\mathcal{J}(\theta) = \sum_{i=1}^{N} \epsilon^2(t+i) = \epsilon^2(t+1) + \epsilon^2(t+2) + \ldots + \epsilon^2(t+N) \tag{4.11}$$

We can write the function $\epsilon(t+i)$ as a vector E so that the error function $\mathcal{J}(\theta)$ becomes:

$$\mathcal{J}(\theta) = \mathcal{E}^T(t)\mathcal{E}(t) = (\mathcal{Y}(t) - \hat{\mathcal{Y}}(t))^T(\mathcal{Y}(t) - \hat{\mathcal{Y}}(t)) \tag{4.12}$$

where $\hat{\mathcal{Y}}(t)$ are the estimated values while $\mathcal{Y}(t)$ are the measured values.

The optimality condition is reached when:

$$\left.\frac{\partial \mathcal{J}(\theta)}{\partial \theta}\right|_{\theta=\theta_0} = 0 \tag{4.13}$$

$$\begin{aligned}
&\tfrac{\partial J}{\partial \hat{\theta}} = -2H^T(Y - H\hat{\theta}) = 0 \\
&H^TY - H^TH\hat{\theta} = 0 \\
&\hat{\theta} = (H^TH)^{-1}H^TY
\end{aligned}$$

This method can be applied only if the matrix $(H^T H)$ is not singular.

In our case the vector Y is formed by (dx_i, dy_i) of the motion vectors of the macro-blocks which are part of the foreground object, while H is build from the coordinates (x, y) of the corresponding macro-blocks. The estimated vector given by $\tilde{\theta} = (a_0, a_1, a_2, a_3, a_4, a_5)^T$ describes the object movement.

Once we have calculated the object motion for all the P frames we have to calculate the one for the I frame. To determine motion model for I frames where we have no motion vectors in MPEG stream, we interpolate the motion models of closer P frames. Here we use linear interpolation in the parameter space. An interpolated parameter θ_i in intermediate moment can thus be computed as:

$$\theta_{P1}(1-\alpha) + \theta_{P2}\alpha \quad \text{where} \quad \alpha \in [0,1] \text{ when } \; t \in [t(P1), t(P2)] \tag{4.14}$$

where $\alpha \in [0,1]$ when $t \in [t(P1), t(P2)]$. Taking into account that the temporal distances between P1 and I frame and I frame and P2 are the same, this interpolation is a simple mean.

If we call $\tilde{\theta}_{P1}$ the estimated vector of the P frame before the I frame and $\tilde{\theta}_{P2}$ the estimated vector of the P frame after the I frame, the motion vector $\tilde{\theta}_I$ calculated for the I frame, is:

$$\tilde{\theta}_I[i] = \frac{1}{2}\tilde{\theta}_{P1}[i] + \frac{1}{2}\tilde{\theta}_{P2}[i] \qquad \forall i \in [0, ..., 5] \tag{4.15}$$

Motion models θ we obtain from MPEG2 correspond to the "forward prediction mode" of the standard. This means that the model-based motion vector for each pixel in frame at t gives its position in the past reference frame at $t - \Delta t_{ref}$. In our case we have to project the object mask in the future.

To realize this forward projection we have to invert the backward motion model, in fact in our case we have to obtain the object macro-blocks (x_c, y_c) of the current frame knowing the position of the object macro-blocks (x_p, y_p) in the precedent frame and the object global motion $\theta = (a_0, \ldots, a_5)$. The backward motion model between current frame C and precedent frame P is the following:

$$\begin{aligned} dx &= x_p - x_c = a_0 + a_1 x_c + a_2 y_c \\ dy &= y_p - y_c = a_3 + a_4 x_c + a_5 y_c \end{aligned} \tag{4.16}$$

Therefore the coordinates of pixels of object macro-blocks in the current frame

(x_c, y_c) can be calculated as:

$$\begin{aligned} y_c &= \left(y_p - a_3 - \frac{a_4 x_p}{1+a_1} + \frac{a_4 a_0}{1+a_1} \right) \cdot \frac{1+a_1}{(1+a_1)(1+a_5) - a_4 a_2} \\ x_c &= \frac{x_p - a_0 - a_2 y_c}{1+a_1} \end{aligned} \tag{4.17}$$

If the object projection from the precedent I frame and the object of the current I frame have a non empty intersection, then we can suppose that the object is the same. In this way, the object label L_O is propagated from the starting I frame to the following I frame in the sequence and we establish a correspondence between the same object in the different frames along the sequence.

It is clear that this reasoning which we illustrate in Figure 4.6 is hold in case where there is no occlusion between objects. That is object masks do not overlap along the time. To cope with occlusions we will need more sophisticated methods to determine which of objects is closer to the camera. This analysis can be found on the method proposed in [73] and is in the perspective of our work.

In Figure 4.7 we shaw the result of the matching processing. In the first column five consecutive I frames at DC resolution are presented. In the second column the extracted foreground objects are shown. In the third column, we show the result of the mask matching process: the object masks have different color corresponding to the associated labels.

4.3.2 Spatio-temporal tube construction

The construction of the spatio-temporal tube has the objective to smooth objects shape along the time. We propose to center the tube on the trajectory of the center of mass of object mask in the sequence. In order to use a reasonably complex model for the tube construction remaining in the framework of *"rough indexing"* paradigm we will suppose that this trajectory is linear. We note that a piece-wise linear approximation can be proposed in a more complex case. Based on this assumption we assume that the tube model is adequately represented by a quadric function in a three dimensional $(2D + t)$ space.

Generally speaking, a quadric equation in an n-dimensional space can be written as follows [78], [79]:

$$\sum_{1 \le i \le j \le n} a_{ij} x_i x_j + \sum_{1 \le i \le n} b_i x_i + c = 0 \tag{4.18}$$

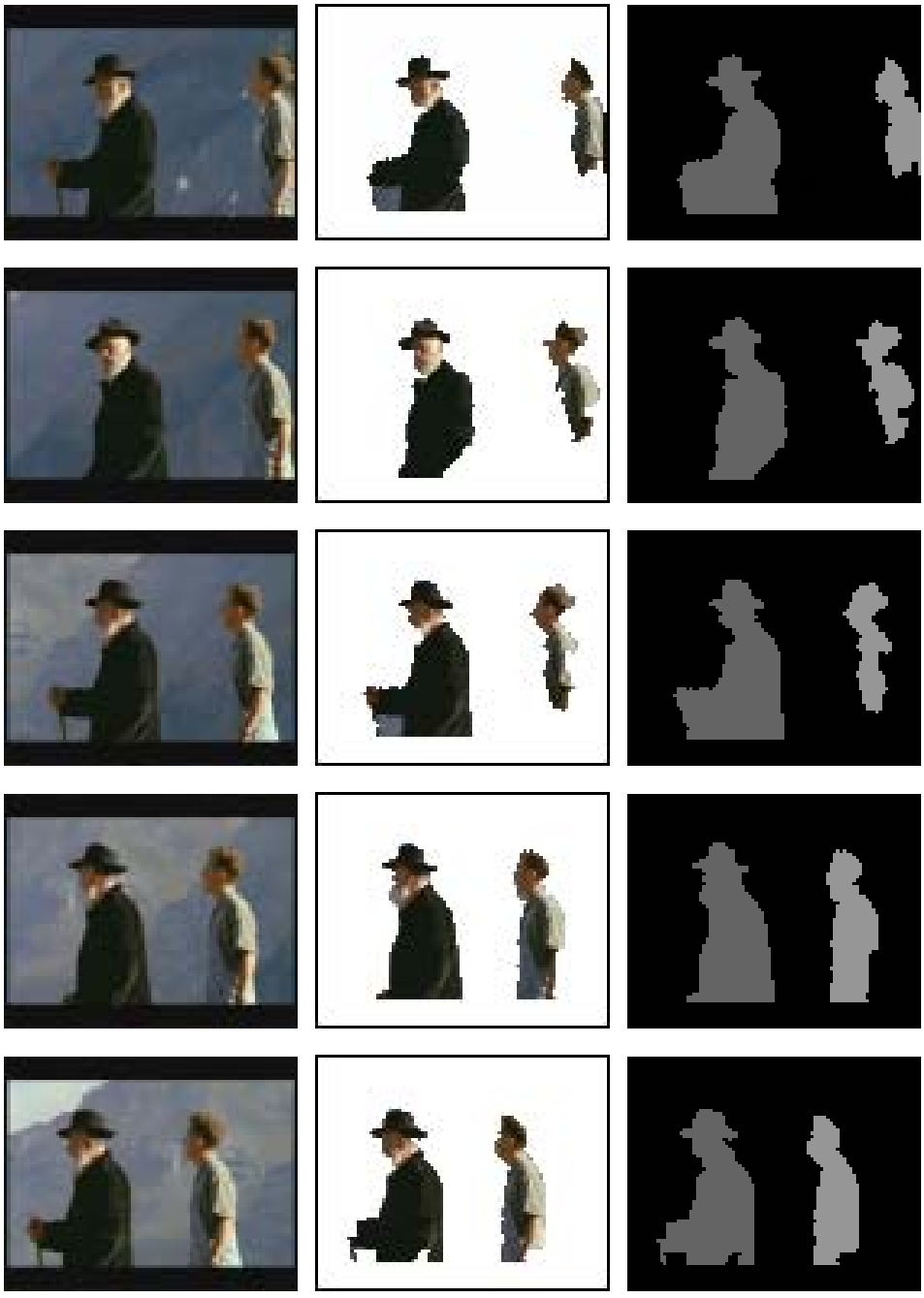
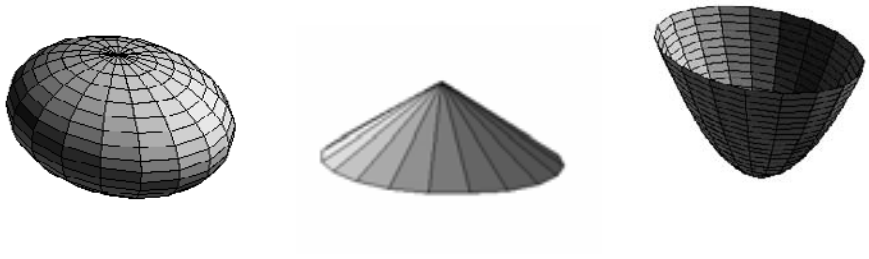
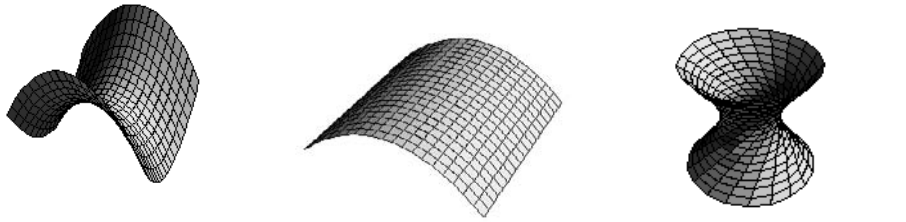


Figure 4.7: Matching of object masks along the time for isolated objects. Sequence "De l'arbre à l'ouvrage", SFRS®.

where a_{ij}, b_i, c are coefficients and at least one of the a_{ij} must be different from zero; in the particular case of n=3 the function is called *quadratic surface*. Quadratic surfaces are also called *quadrics*, and there are 17 standard-form types. In Figure 4.8 some quadrics are plotted.



(a) ellipsoid (b) elliptic cone (c) elliptic paraboloid



(d) hyperbolic paraboloid (e) parabolic cylinder (f) hyperboloid of one sheet

Figure 4.8: Some examples of quadratic functions.

Our purpose now is to find the coefficient a_{ij} of a generic quadric as in (4.19) that can best approximate the contours of the objects in the sequence we are considering.

$$a_{11}x^2 + a_{22}y^2 + a_{33}t^2 + a_{12}xy + a_{13}xt + a_{23}yt + a_{14}x + a_{24}y + a_{34}t + a_{44} = 0 \quad (4.19)$$

As it was shown from construction method of the object masks from rough data, we cannot expect very precise detections and therefore we cannot use the exact (x, y) coordinates to estimate the coefficients in equation 4.19.

Therefore we will smooth the object shape by introducing a probabilistic characteristic function of the object shape z as we described in 4.2. We define the function $z(x, y)$ as a gaussian function centered on the object center of mass. In the section 4.2) the center of mass was the object center, while in this case, we compute the trajectory and we will take this line as the center of mass line as we will explain next

(see equation (4.20)).

$$z = \exp\left(-\frac{1}{2}\left(\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2}\right)\right) \tag{4.20}$$

Now the quadric equation can be written as:

$$z = a_{11}x^2 + a_{22}y^2 + a_{33}t^2 + a_{12}xy + a_{13}xt + a_{23}yt + a_{14}x + a_{24}y + a_{34}t + a_{44} \tag{4.21}$$

We note that in such formulation we have a linear model for the unknown parameters $(a_{11} \dots a_{44})$. Knowing the sequence of coordinates (x, y, t) and the values of z, we can therefore obtain the estimation of parameters by least square approximation. It is clear that the resulting parameters will strongly depend on the quality of the observation z. In practice relative to equation 4.20 we have to identify (μ_x, μ_y) and (σ_x, σ_y).

As it may happen that the objects are not correctly detected or occluded, the object center of mass can be different from the mask one. In Figure 4.9 a) and b) two DC frames are shown; in the second one the object at the left is partially covered by the background and for this reason it is only partially detected (see Figure 4.9 c)) where the object masks are presented). In this case using the object center of mass to compute the object trajectory may create some bias.

We suppose that the object motion does not change along the sequence and we suppose that the object center of mass follows a straight line trajectory. In most cases this is not true because of the detection errors caused by working with *"rough data"*, so an approximation of trajectory is needed to reduce the influence of the object detection errors. To obtain the approximation of the object center of mass line we use again a least-square estimation , and the problem becomes the approximation of a set of points (the object centers of mass) with a line in a 3D space (x, y, t).

In the 3D space a straight line is the solution of a linear system representing the intersection of two different planes:

$$\begin{cases} a_1x + a_2y + a_3t + a_4 &= 0 \\ b_1x + b_2y + b_3t + b_4 &= 0 \end{cases} \tag{4.22}$$

As it is always possible to write one equation in an explicit form and then operate a substitution in the second one, we can obtain a system where one of the two

(a) (b) (c)

Figure 4.9: Example of wrong object center of mass caused by a partial object detection.

equations depends only on two variables:

$$\begin{cases} y &= k_1 x + k_2 \\ t &= k_3 x + k_4 y + k_5 \end{cases} \tag{4.23}$$

It means that from all the possible planes we choose one of the two as a plane that is parallel to one of the space axes (4.23).

In Figure 4.10 we show an example of approximation of object barycenters by a straight line for the sequence 'Hiragasy' (SFRS®). Here the projections on ty plane is shown.

We note that such an approach helps in case the object mask is not detected in a given frame, such as when there is no relative motion between object and camera in P frames or some errors occurred.

Once the straight-line parameters $(k_1 \dots k_5)$ have been estimated, the new optimal coordinates (x_c, y_c) of the object center of mass for any frame along the sequence can be obtained. In Figure 4.9 c) the difference between the object center of mass and the one given by least square estimation. The little circle represents the centroid of the object mask, while the cross represents the estimated center of mass. The frames before the image shown in 4.9 b) lead all to a whole object detection and for this reason the estimated center of mass and the object extent indicate that the real object is in reality larger than the detected one.

This coordinates are also used in (4.20) as $\mu_x = x_c$, $\mu_y = y_c$ for a given moment of time t. The standard deviations (σ_x, σ_y) are represented by the maximum distance between the optimal center of mass (x_c, y_c) and the bounding box in the two directions x, y as shown in Figure 4.11.

Once the function $z(x, y)$ has been computed for all the images along the time

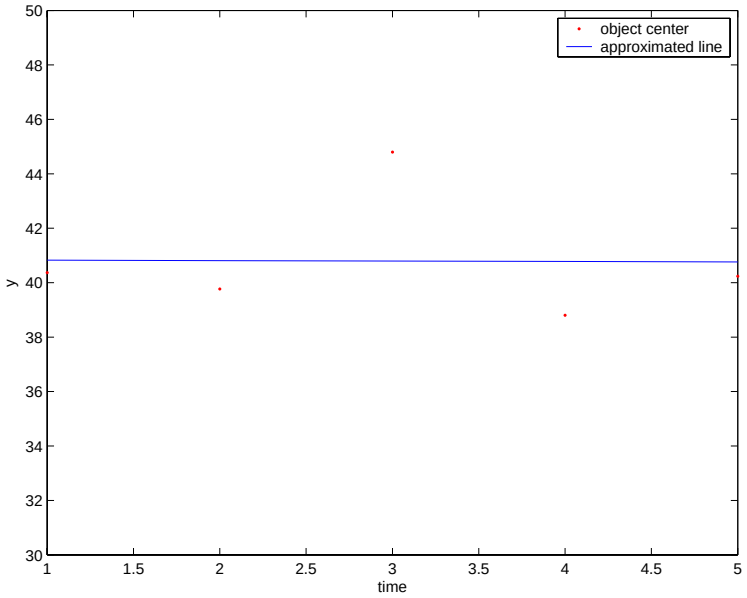
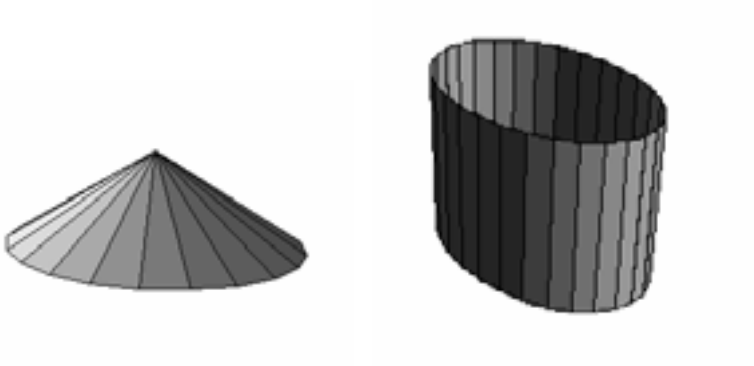


Figure 4.10: Least square approximation of the trajectory of object barycenters

axis, we can approximate the sequence of such functions with a quadric given by (4.19). Our purpose is to obtain a set of quadric functions with the same axis and to choose the one that better approximates the sequence of object shapes. For example, in case of a zoom sequence where the object change its dimension because the camera goes closer to it, an elliptic cone can be the quadric function that best approximates object changes, while in the case of an horizontal camera movement (called *pan*) an elliptic cylinder may be more adequate.

Equation (4.19) represents a generic quadric function, but for our aim we can

Figure 4.11: Sigma values used to compute the gaussian function in (4.20).



(a) (b)

Figure 4.12: A cone and a cylinder can approximate object shape changes in case of zoom and pan sequences respectively.

simplify it and consider only some constrained cases. As we are not interested in an arbitrary 3D volume but only in the volume slices taken along the t axis, to simplify the computation we can locate all the centers of mass along an axis parallel to the t axis (see Figure 4.13). In this way we will avoid terms in xy, xt or yt in the final equation.

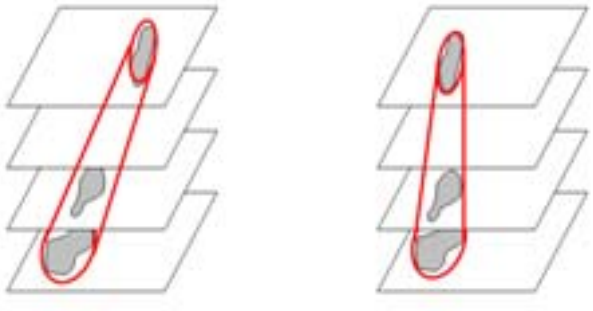
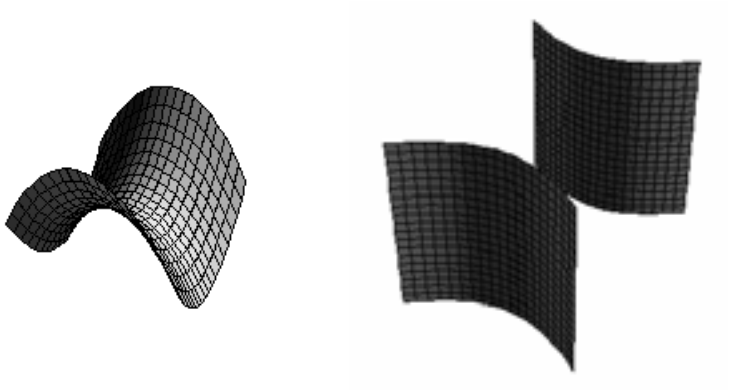


Figure 4.13: Translation of the barycenter line to an axis parallel to the t axis.

Furthermore we can suppose that as the object motion is uniform along the sequence, we can add to (4.21) some constraints on the parameters to obtain at the end a volume with shape like a cone, a cylinder or an hyperboloid. These constraints permit to reduce the total number of parameters we have to compute and to avoid a degeneration of the quadric function (such as for examples couple of planes) imaginary quadrics (for example imaginary cones or cylinders) or other quadrics, real and not degenerate (see for example Figure 4.14), that are impossible



(a) hyperbolic paraboloid (b) hyperbolic cylinder

Figure 4.14: Example of real and not degenerate quadrics that are not useful for our purposes.

in our case.

Such constraints applied to (4.21) lead to:

$$z = a_{11}x^2 + a_{22}y^2 + a_{33}t^2 + a_{14}x + a_{24}y + a_{34}t + a_{44} \tag{4.24}$$

For convenience we center the function in the center of the image (x_0, y_0). As we consider only a limited set of possible quadrics, we can also add some additional constraints to (4.24). In particular from the canonic form of a conic centered in (x_0, y_0, t_0) and in case of positive z we get:

$$z = a_{11}x^2 + a_{22}y^2 + a_{33}t^2 - 2a_{11}x_0x - 2a_{22}y_0y + a_{34}t + a_{44} \tag{4.25}$$

with the following constraints:

$$\begin{cases} a_{11} & > 0 \\ a_{22} & > 0 \\ a_{33} & \leq 0 \end{cases} \tag{4.26}$$

The next step is then to estimate the five parameters in (4.25) to obtain the function which best approximates the evolution of object shape and dimensions along the sequence.

Given a vector of measures $\mathbf{z} = [z_1, \ldots, z_N]^T$ and given the set of bounding coordinates $(x_1, y_1, t_1), \ldots, (x_N, y_N, t_N)$, we can write the (4.25) in a matrix form as:

$$\mathbf{z} = \mathcal{H}\beta \tag{4.27}$$

under the constraint:

$$\mathcal{A}^T\beta \geq 0 \tag{4.28}$$

Here

$$\mathcal{H} = \begin{bmatrix} x_1^2 - 2x_0x_1 & y_1^2 - 2y_0y_1 & t_1^2 & t_1 & 1 \\ \vdots & & & & \vdots \\ x_N^2 - 2x_0x_N & y_N^2 - 2y_0y_N & t_N^2 & t_N & 1 \end{bmatrix} \tag{4.29}$$

and

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.30}$$

Let us denote $\mathbf{e}(\beta) = \mathbf{z} - \mathcal{H}\beta$ the error with respect to the model 4.27. We will solve the following optimization problem:

$$\begin{array}{ll} \min & \frac{1}{2}\mathbf{e}^T\mathbf{e} \\ \text{under constraint} & \mathcal{A}^T\beta \geq 0 \end{array} \tag{4.31}$$

This is a *quadratic programming* problem.

Generally speaking, if a problem of quadratic programming it can be rewritten in the form:

$$\begin{array}{ll} \min & \frac{1}{2}\mathbf{x}^T\mathcal{G}\mathbf{x} + \mathbf{g}^T\mathbf{x} \\ \text{under constraint} & \mathcal{A}^T\mathbf{x} - \mathbf{b} \geq 0 \end{array} \tag{4.32}$$

then it possible to define the dual problem ([80]):

$$\begin{array}{ll} \max & \frac{1}{2}\mathbf{x}^T\mathcal{G}\mathbf{x} + \mathbf{g}^T\mathbf{x} - \lambda^T(\mathcal{A}^T\mathbf{x} - \mathbf{b}) \\ \text{under constraint} & \mathcal{G}\mathbf{x} + \mathbf{g} - \mathcal{A}\lambda = 0 \text{ with } \lambda \geq 0 \end{array} \tag{4.33}$$

where λ is a *Lagrange multiplier*. Equation (4.33) can then be simplified:

$$\begin{array}{ll} \max & -\frac{1}{2}\lambda^T(\mathcal{A}^T\mathcal{G}^{-1}\mathcal{A})\lambda + \lambda^T(\mathbf{b} + \mathcal{A}^T\mathcal{G}^{-1}\mathbf{g}) - \frac{1}{2}\mathbf{g}^T\mathcal{G}^{-1}\mathbf{g} \\ \text{under constraint} & \lambda \geq 0 \end{array} \tag{4.34}$$

This is still a quadratic programming problem in λ, but it is easier to solve. Once the value of λ has been found, the value of $\mathbf{x}$ is obtained by (4.33).

In our case, developing (4.31) for $\mathbf{e} = \mathbf{z} - \mathcal{H}\beta$, we obtain:

$$\begin{array}{ll} \min & -\frac{1}{2}\beta^T(\mathcal{H}^T\mathcal{H})\beta - \mathbf{z}^T\mathcal{H}\beta + \frac{1}{2}\mathbf{z}^T\mathbf{z} \\ \text{under constraint} & \mathcal{A}^T\beta \geq 0 \end{array} \tag{4.35}$$

This problem is of the same form of (4.33) that can be solved in the form of (4.34) where $\mathcal{G} = \mathcal{H}^T\mathcal{H}$ and $\mathbf{g} = -\mathcal{H}^T\mathbf{z}$. The vector λ is obtained by deriving with respect to (4.34):

$$\lambda = -\left[\mathcal{A}^T(\mathcal{H}^T\mathcal{H})^{-1}\mathcal{A}\right]\left[\mathcal{A}^T(\mathcal{H}^T\mathcal{H})^{-1}\mathcal{H}^T\mathbf{z}\right] \tag{4.36}$$

and then the vector β can be obtained from (4.33) setting β to $\mathbf{x}$, $\mathcal{G}$ to $\mathcal{H}^T\mathcal{H}$ and $\mathbf{g}$ to $-\mathcal{H}^T\mathbf{z}$:

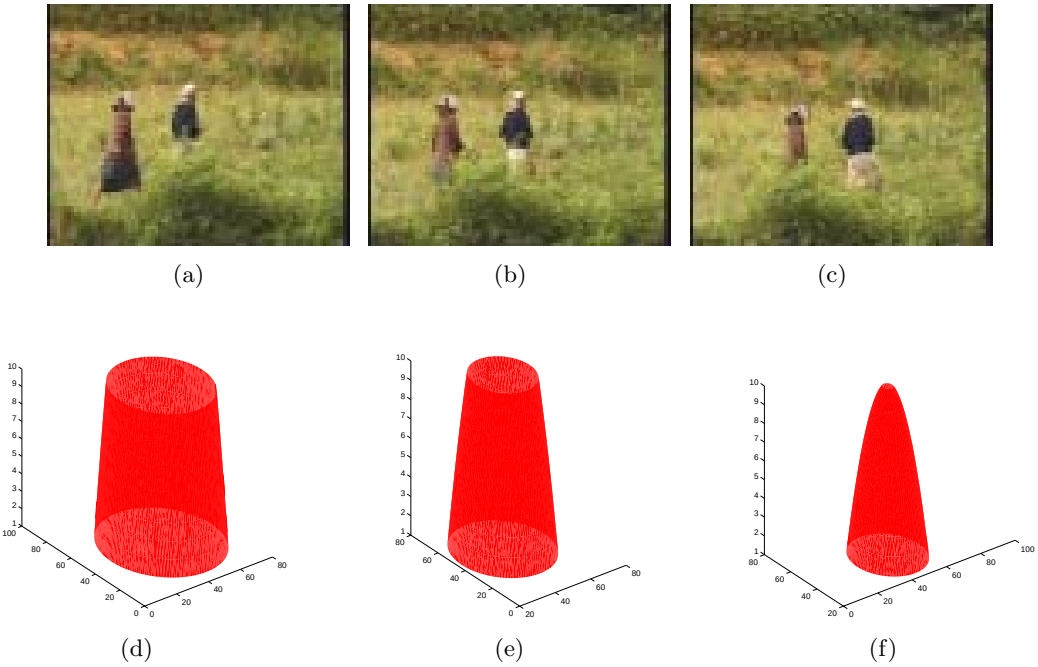
$$\beta = \left(\mathcal{H}^T\mathcal{H}\right)^{-1}\left(\mathcal{A}\lambda + \mathcal{H}^T\mathbf{z}\right) \tag{4.37}$$

Now with these optimal parameters, varying z, we can obtain a family of quadrics with the same central axis. To compute the conic function that best fits all the object masks we have to fix the value of z. In Figure 4.15 we see above some DC frames extracted from a video sequence; we can observe that the object on the left, changes its dimension because at the end of the sequence it is covered by the background. In Figure 4.15 below we can observe the quadric calculated for this sequence of frames. The three quadrics are described by the same equation, but they vary for the value of z.

In Figure 4.16 another example is presented: a quadric of different shape is calculated with different values of z, while the other five parameters remain the same.

To choose the value of z that gives the best approximation of the quadrics for the object masks of a sequence we minimize the following function for each quadric section (in all the case an ellipse or a circle) for different values of $z = z_0$:

$$\min \sum_{(x,y)} \delta(x,y)\ \forall t = 1, \ldots, N \tag{4.38}$$



(a) (b) (c)

(d) (e) (f)

Figure 4.15: Above three DC frames of a sequence. Below the quadric that approximates the shape changes for the object at the left.

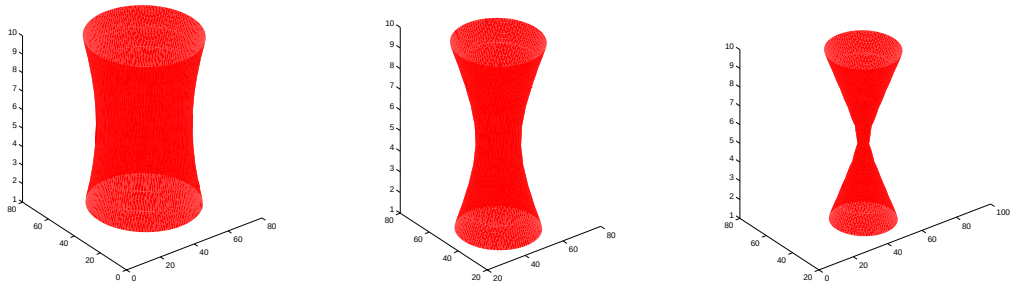
$$\delta(x,y) = \begin{cases} 1 & \text{if } (x,y) \in ((\text{ellipse} \cup \text{mask}\ ) - (\text{ellipse} \cap \text{mask}\ )) \\ 0 & \text{otherwise} \end{cases} \tag{4.39}$$

To minimize this function we have first to translate each section of the quadric to the line of the center of masses computed earlier. In Figure 4.17 an example of the obtained result is shown.

4.4 Conclusions

In conclusion, in this chapter we have presented how we extract foreground object characteristics after have extracted the object from the video sequence.

At the same time, we proposed a filtering of the detection results. Taking into account rough data and noisy segmentation results, we proposed a smoothed solution for object-based indexing. First the object shape in each I frame is smoothed by Gaussian minimizing miss-matching criterion and resulting in optimal elliptic shape in I-frame. Then the trajectory of the center of mass of the ellipses have been approximated by optimal straight lines by least square linear regression. Finally,



(a) (b) (c)

Figure 4.16: Another example of three quadrics built with the same five parameters but different z values.

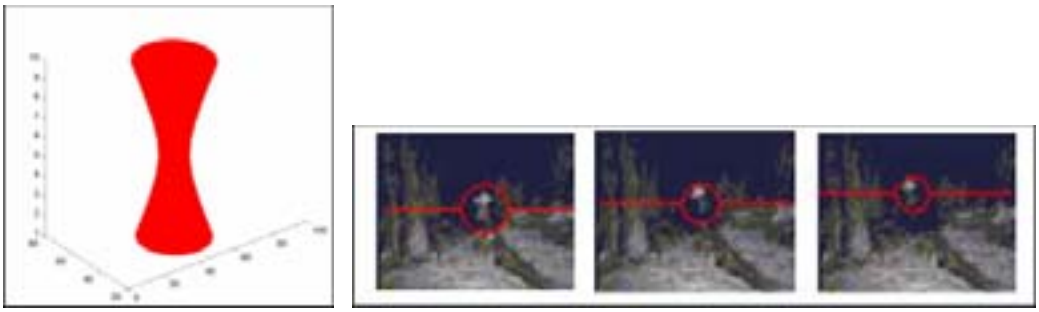


Figure 4.17: a) The quadric calculated and b) the section of the function that approximate the object shape and position

based on the ellipse shape and the approximated trajectory of the centers of mass, we proposed a spatio-temporal tube for each object. This tube captures object motion along the time and also gives a feed-back to the segmentation method. In case the object has not been detected at intermediate moment of time, this tube sections will approximately show the *locus* of the object in a video frame.

All these three models can serve as object descriptors in the context of rough indexing paradigm. They give a user an overview of approximate shape and object behaviors.

In the next chapter more results will be presented summarizing in this way all the work done.

Chapter 5

Results

5.1 Introduction

In chapter 3 and chapter 4 we have presented the method we propose to extract the foreground objects from a video sequence. In this chapter the results tested on different sequences will be presented. First of all, we will present a general discussion on the results and our evaluation method. Then, in section 5.3 we will present a sequence of a generic MPEG2 video and we will show step by step how we extract the foreground objects; then we will summarize the results for a set of generic videos. In section 5.4 we will do the same with cartoons: we will present the results step by step for a single cartoon and then we will summarize the results of the test on a set of cartoons.

5.2 Results

The motion and color based approach we have presented in this thesis work has been tested on different sequences from a set of natural video content and cartoons. We have considered feature documentaries "De l'Arbre à l'Ouvrage", "Hiragasy", "Aquaculture", "Chancre" (SFRS ®), and the cartoons "Au bout du monde", "Ferrailles", "La bouche", "Le moine et le poisson" (Channel+ ®), "Casa", "François le vaillant", "Le trop petit prince", "Petite escapade" (Folimage ®), "Le chat d'appartement" (Canal+ and Channel4 ®) and "Le roman de mon ame" (Channel4 and La Sept-Arte ®). We suppose that segmentation of the videos into shots has already been performed, so we randomly select shots amongst those containing foreground objects. As far as camera motion is concerned for the set of processed content, pan,

tilt, zoom and hand carried camera motion artefacts have been observed. In section 5.3.1 an entire sequence extracted from "De l'arbre à l'ouvrage" is presented and all the steps necessary to obtain the foreground objects are presented. The same has been done for the cartoon "Ferrailles" in section 5.4.

The reason for the choice of these two types of content are the follows. In this work we tackle a difficult problem of processing of artistic content. No model for such a content is available on the contrary to the soccer games. No assumption on the color of the background (such as play field in soccer) can be done. The only assumption is that the objects are animated with their proper motion with regard to the background motion (camera). From this point of view, the method will work if proper motion areas can be isolated by the method. In natural content it is mainly the case. Due to the natural texture and high frequency details in natural shootings, the input of our method, i.e. MPEG motion vectors, carry sufficient information thus allowing for satisfactory performance of the method. The failure of the method could be expected when objects are too small, i.e. less then the macro-block size. In this case the proper motion cannot be captured by MPEG motion estimators. Another situation when several objects can be mixed together is when their projection into the image plane are close, that is their relative distance is less then a macro-block size. In this case the method will be able to separate only the global objects areas from the background.

Another form of artistic content, cartoons, represent a challenging content for the method. The images present in cartoons are very different form natural ones. In fact the colors are often uniform both for foreground object and for the background. In addiction there is not the color diversity present in nature. This means that each frame has many flat zones with motion vectors that are independent from the camera motion. If this flat zones cover most part of the image, the estimated camera motion is not correct. The consequence is that all extracted motion masks are often inadequate, so that no object can be extracted. We will show some example extracted from some cartoons we have analyzed.

In the next section we will explain the method we have used to evaluate the results.

5.2.1 Evaluation method

The results have been validated by visual comparison with manually segmented images. Usually, in segmentation methods, the validation methods include the comparison of automatic with respect to manual segmentation in terms of detected pixels,

missed pixels and wrongly detected pixels. In our case it is not possible to adopt such approach. In fact, as we work with rough data and reduced spatial resolution, our results will be rough as well. Even manual segmentation may be difficult when working at macro-block resolution.

Thus we propose a visual comparison between the detected foreground objects and the original I frame. When we will show the results we will show a table with the percentage of detected objects with respect to objects present in the sequence; we will also show the cases of miss-detections and over-detections. By "miss-detection" we mean objects that have not been detected, while by "over-detection" we refer to the objects that have been detected but include a large portion of the background. Therefore as far as foreground objects are concerned, the percentage of detected objects, missed objects or over-detected objects will be listed.

We will show the same results at the frame level: that is, we will consider the percentage of correctly processed frames with respect to the total number of frame in the sequence. This ratio will be the same as the number of correctly detected foreground objects only if the frames contain a single object; otherwise they will differ. We will also describe in such cases the number of miss-detections (all object not detected or one object not detected) and over detections (background detected as moving object).

Performance evaluation as a visualization platform

To accelerate the evaluation process, we propose to use a tool able to visualize one or more characteristics of the extracted objects while displaying at the same time the original video sequence. We have designed for this purpose a simple visual interface that is shown in Figure 5.1.

In Figure 5.2 a simple use of the visualization tool is shown. In this case at the top of the visualization tool the graph of the desired feature (the object number) is presented. On the graph it is possible to see a vertical line; this line identifies the time stamp: while the video is playing, the line shifts from left to right. It is very simple to establish this way a correspondence between the algorithm result and what happens in the video.

Thanks to the vertical line, it is possible to navigate along the video and instantaneously view the result at any given location. In Figure 5.3 we have found an error, in fact in the video no object is present while in the visualization tool an object has been found.

Another example is presented in Figure 5.4 where two objects have been detected

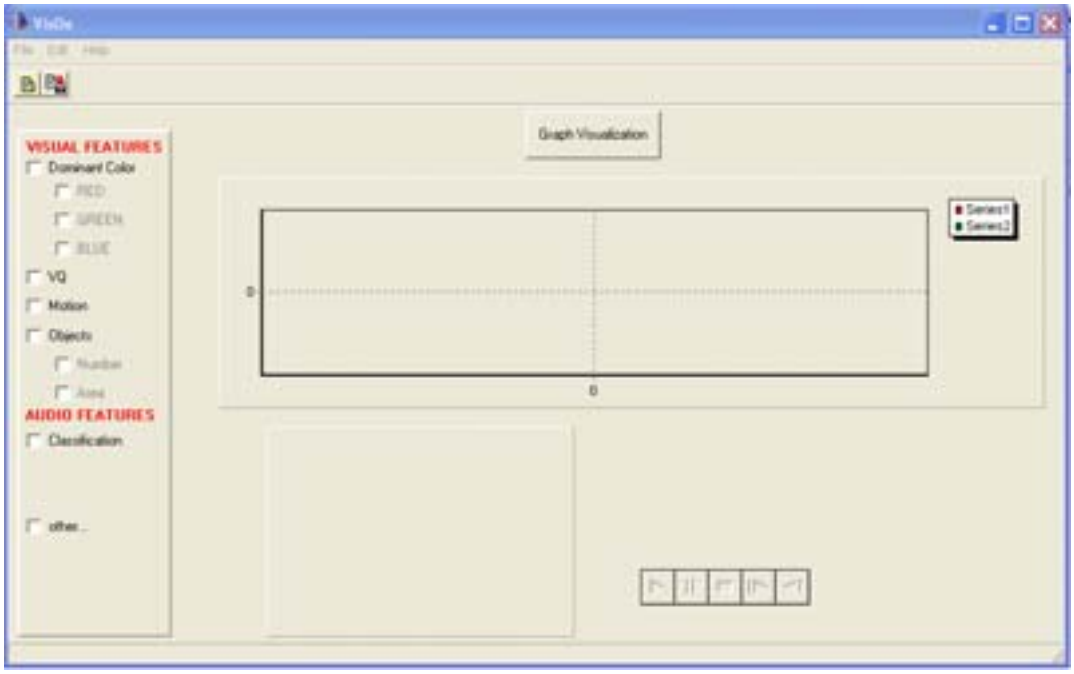


Figure 5.1: The tool used to visualize the results.

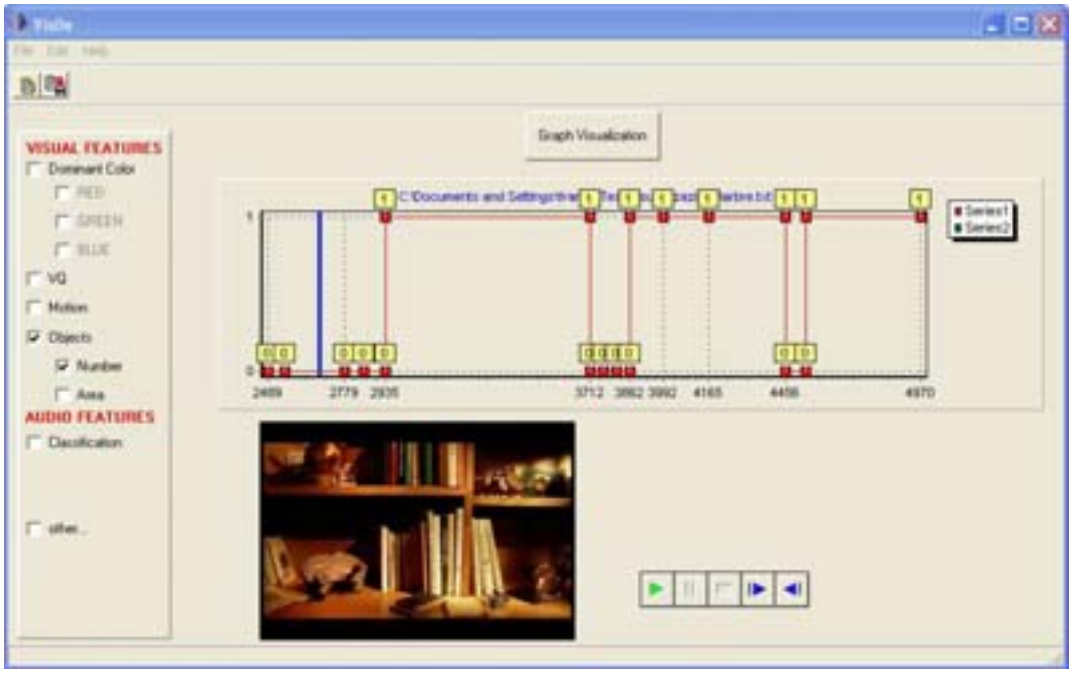


Figure 5.2: The visualization tool.

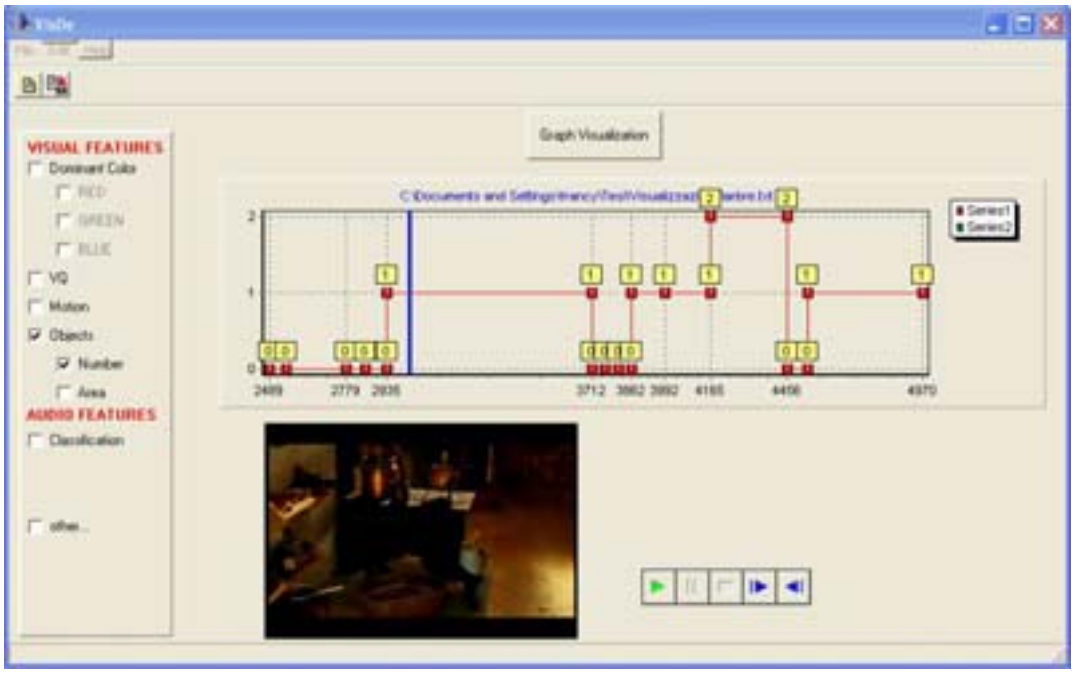


Figure 5.3: An error on the object extraction method occurred: in the graph, where the results of the extraction method are represented, it is possible to see that the extraction method has found an object; this is not true as it appears in the corresponding frame of the video sequence.

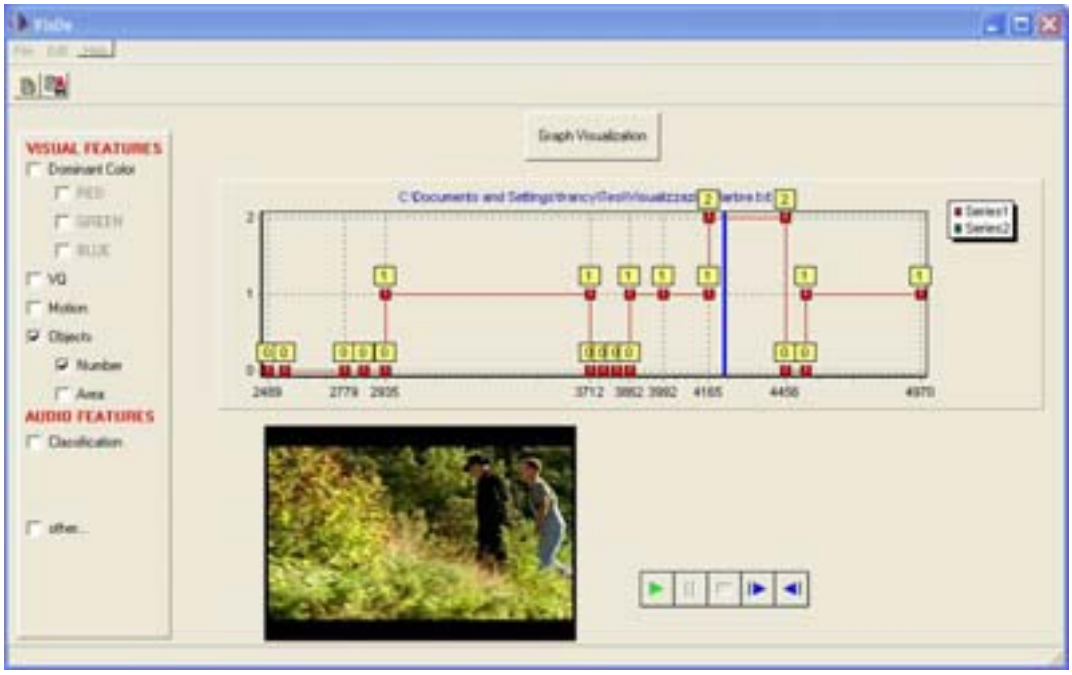


Figure 5.4: The graph shows that our detection algorithm has found two objects in the video sequence. The result is correct in fact in the video sequence we can see two men walking.

from our extraction method and two objects are indeed present in the video.

As a future perspective, we plan to extend this visualization tool so as to be able to visualize two or more properties of the video sequence (e.g. areas of the detected objects) at the same time. This visualization can further help to extract other characteristics of the object behavior that cannot be automatically extracted.

5.3 Analysis of a generic video sequence

In this section we will present an example of a generic video sequence that we have analyzed to extract the foreground objects. We will show all the steps of our method with some intermediate results; we want to detail this way the problems that have been found during the sequence analysis and how they could be overcome. A table will summarize the results. Some images will also be displayed to show the extracted foreground objects for some of the analyzed sequences.

5.3.1 "De l'arbre à l'ouvrage"

"De l'arbre à l'ouvrage" is a video sequence that presents indoor and outdoor scenes whit different types of illumination and color; it presents also different types of camera movements.

Now we present a sequence chosen from this video. The length of the sequence is 40 GOP. Since in this case each GOP is composed by 15 frames, its length is of 600 frames, that is 24 seconds. The I frames of this sequence are displayed in Figure 5.5.

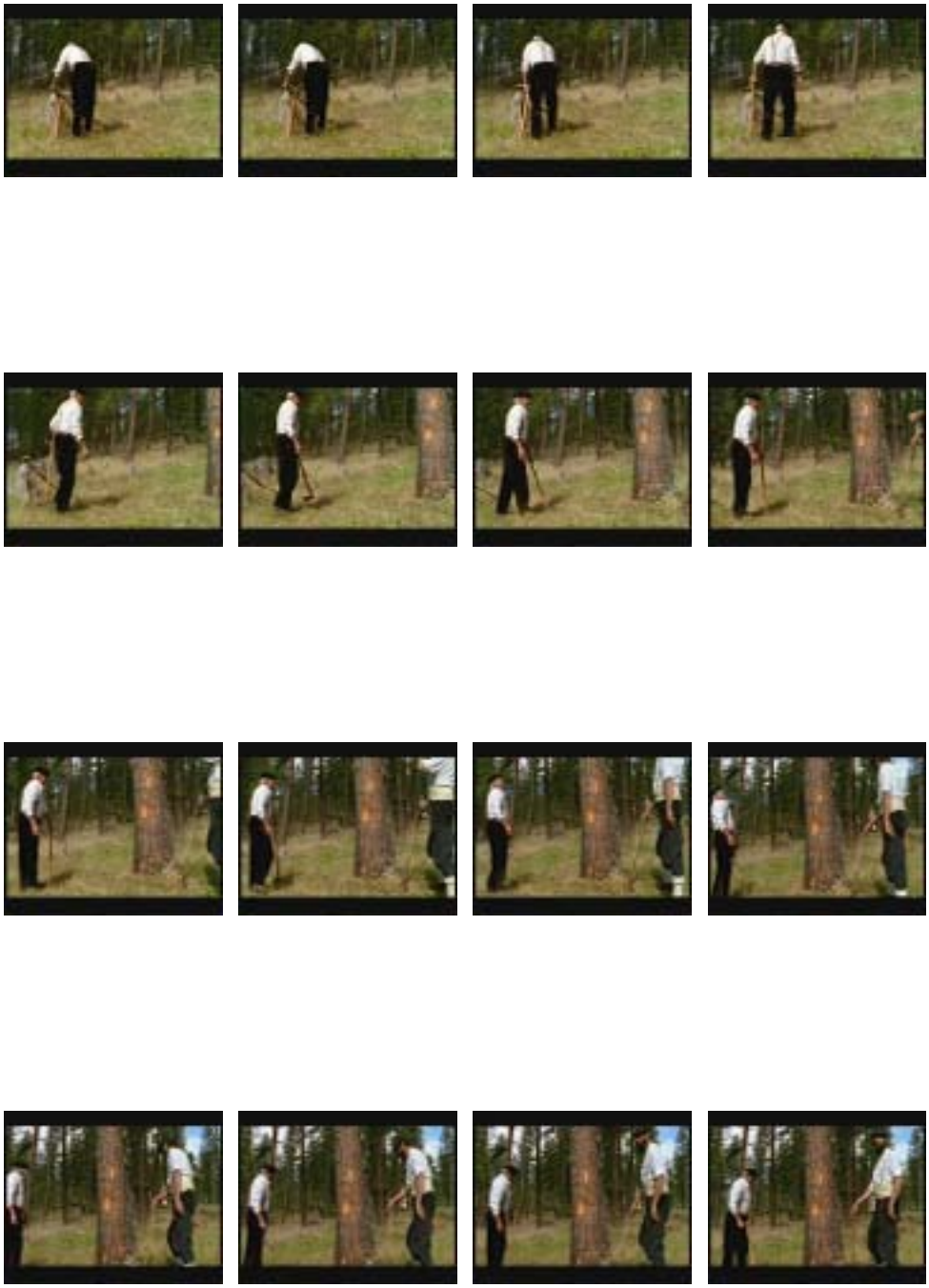


Figure 5.5: I frames from the analyzed sequence of “De l’arbre à l’ouvrage”.

Figure 5.6: Extraction of motion masks in P frames.

As we have explained in chapter 2 the first step of our rough object extraction method requires a camera motion estimation from P frame motion information. Incoherent motion vectors with respect to the camera motion model provide an estimate of moving objects through motion masks.

In Figure 5.6 we show some DC images and the motion masks extracted from the P frames that precede the I frames. In the first row the extracted masks correspond quite well to the corresponding DC image, but this is not always the case. Even in a sequence where there are not the so called "flat zones" it is very common to have detection errors and motion masks that do not correspond to the DC images, as we can see in the second row of Figure 5.6. For this reason we cannot directly use these masks to identify the foreground objects but it is necessary to filter them with the methods we have presented in chapter 2. These methods of filtering and interpolation lead us to obtain more accurate motion masks for I frames, as we show in Figure 5.7; in fact it is possible to see that even in those case where the P frames masks were very noisy the filtering operations have led to good results.

Once object motion masks interpolated on the I frames, we have to combine them with the color segmentation of such I frames. So the first step is to process the I frame with a morphological filter as explained in chapter 2. In Figure 5.8 we show the result of this filtering process. In 5.8 a) a DC image is presented and in b) the sole Y component of the image. In the 5.8 c) the filtered image is presented, and it is possible to see that the filtering process has produced a much more homogeneous image and consequently the segmentation process will be more accurate. In Figure 5.8 d) we present the result of color segmentation process where each region has been

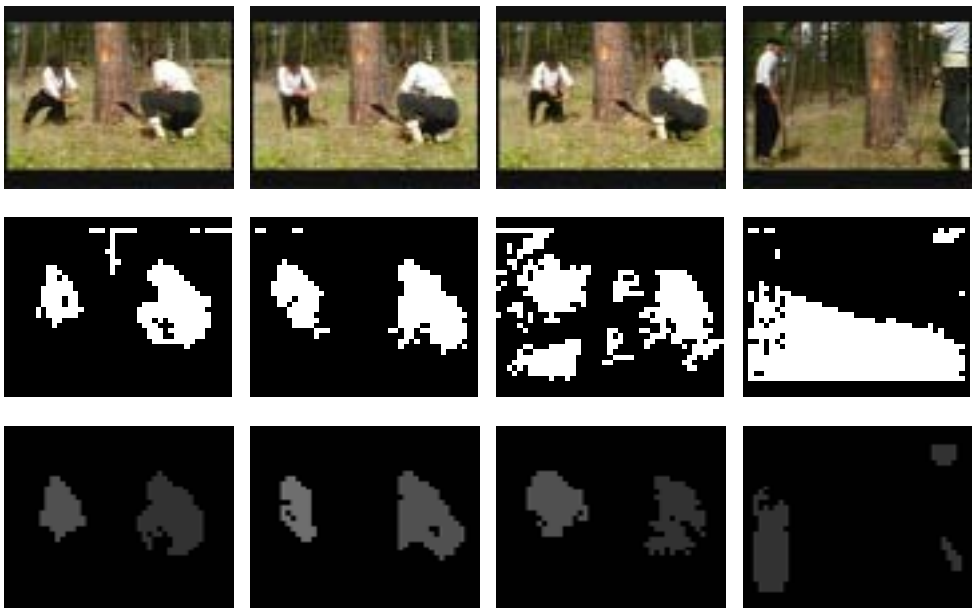


Figure 5.7: In the first row some DC images are presented, in the second row the motion mask extracted from a P frame and in the third row the motion masks obtained for the I frames.

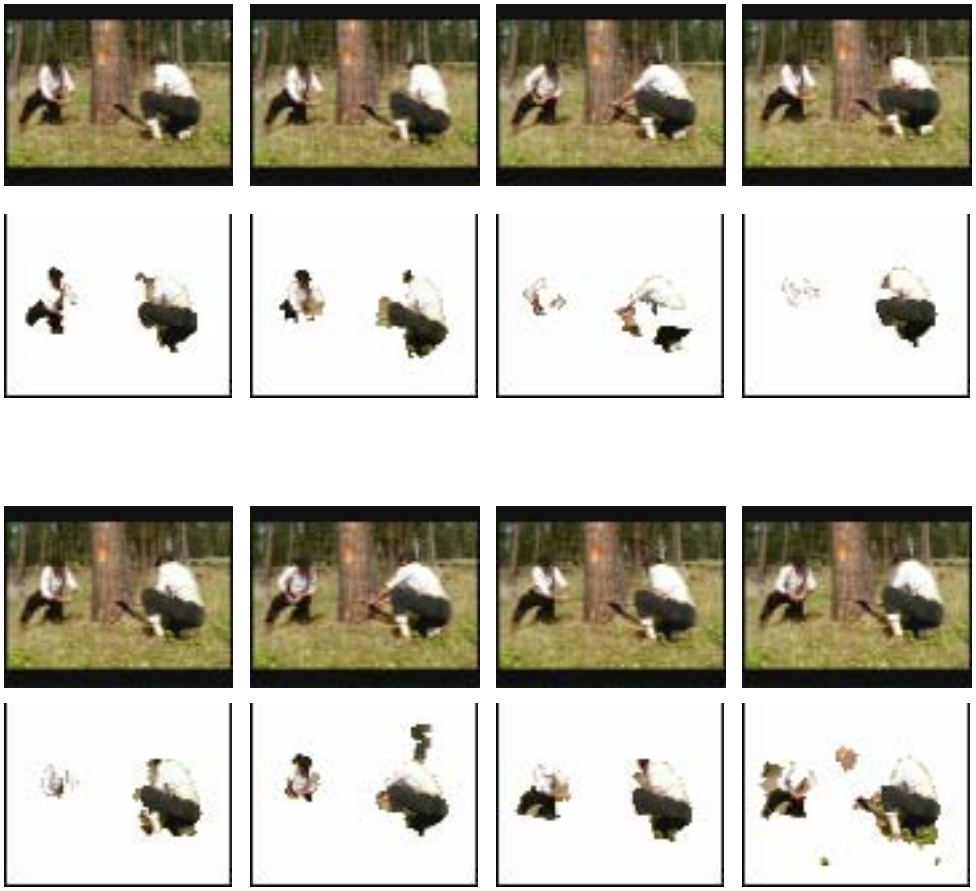
substituted with its average color value.

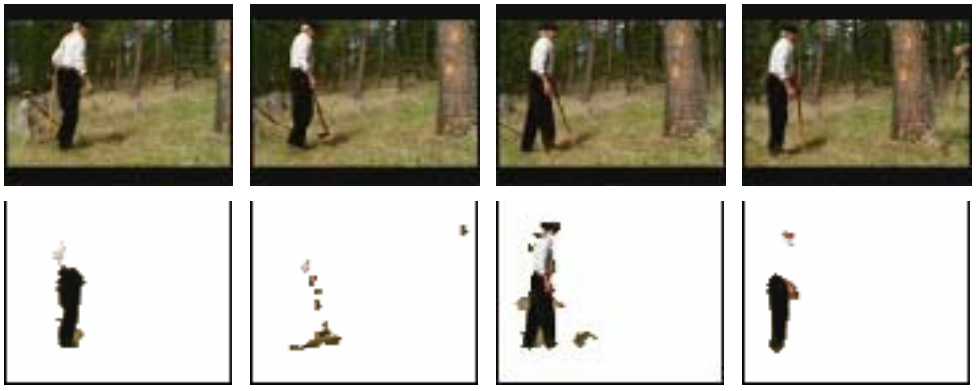
Once we have extracted both motion and color information we can merge them to obtain foreground objects. In Figure 5.9 we show the extracted objects. The result is good, in fact if we compare the extracted objects with the sequence of DC images we can see that the moving objects are completely detected. Some imprecisions at the object boundaries exist but this is a consequence of working at reduced resolution. In the first rows of Figure 5.9 we can see that the objects are partially detected, in this case often only the top part of the body is detected. The reason is that the people

(a) (b) (c) (d)

Figure 5.8: The result of the process of morphological filtering and color segmentation.

are mainly still since they only move their arms so that only these are detected. So, in this case, even if the people have been partially detected there is no error in the algorithm since the moving parts of the objects have been correctly detected.





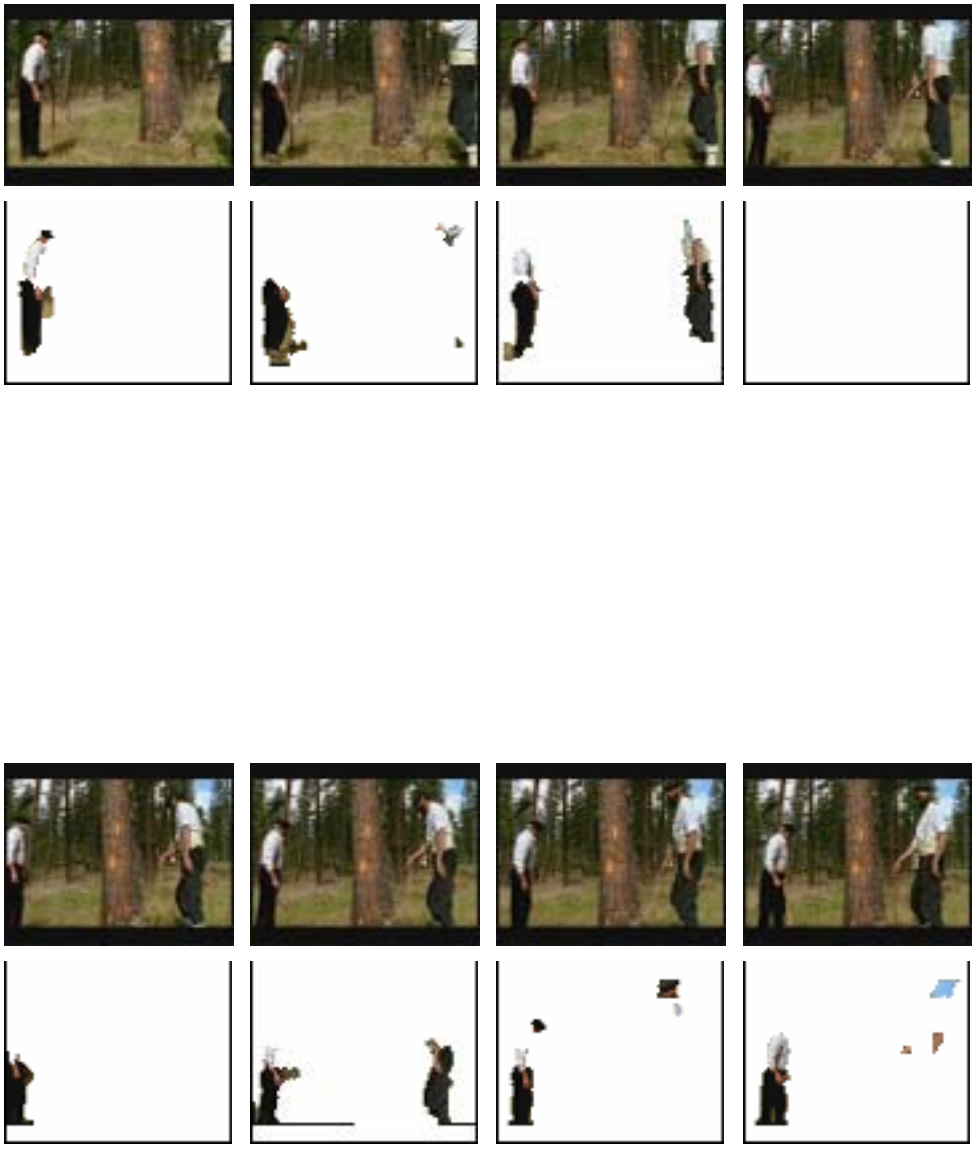


Figure 5.9: The extracted foreground objects for the sequence presented in Figure 5.5.

In Figure 5.10 we show a sequence where an object is partially detected. In this case the object motion was well detected and the motion mask was perfect, but an error in the segmentation process occurred so, combining the results of the motion and static segmentation processes we obtained the result indicated in the second row of Figure 5.10.

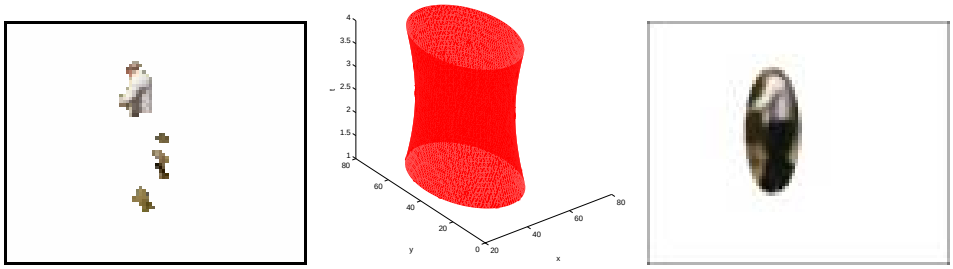
Figure 5.10: A sequence where an object is not correctly detected due to an inconsistency occurred in the segmentation process.

In this case the temporal filtering with the quadric function can help to correct the result in the case of wrong detection. So we compute the center of mass of the foreground objects and we approximate them with a straight line, then we approximate the masks with a quadric elliptic section that is function of object dimensions. In Figure 5.11 the result is shown. In Figure 5.11 a) the foreground objects are shown before the temporal filtering; Figure 5.11 b) shows the quadric that approximates the object along time and Figure 5.11 c) indicates the section of the quadric at the moment of time corresponding to the frame displayed in a). In this way it is not possible to have precise object boundaries but it is possible to have an approximation of object position in time and a better coverage of it.

5.3.2 Generic video results

We have analyzed other video sequences and in this section we propose a table to summarize the results (see table 5.1).

In the sequence 'arbre ♯3' two different situations are present. In the first part of the sequence two men are walking in the forest. In this case we have no problem



(a) (b) (c)

Figure 5.11: Approximation of object position and shape using the quadric based spatio-temporal interpolation.

Sequence	% detected objects	missd.	overd.	% correct frames	missd.	overd.
arbre ♯1	48/59 (**81**%)	10/59	1/59	32/40 (**80**%)	8/40	0
arbre ♯2	21/26 (**81**%)	4/26	0	16/19 (**84**%)	3/19	0
arbre ♯3	47/71 (**66**%)	5/71	19/71	42/63 (**67**%)	1/63	20/63
arbre ♯4	6/10 (**60**%)	2/10	2/10	6/11 (**54**%)	0	5/11
arbre ♯5	10/16 (**63**%)	6/16	0	23/29 (**79**%)	6/29	0
arbre ♯6	15/22 (**68**%)	5/22	2/22	15/22 (**68**%)	5/22	2/22
arbre ♯7	22/64 (**34**%)	42/64	0	15/32 (**47**%)	17/32	0
arbre ♯8	53/64 (**83**%)	11/64	0	22/24 (**92**%)	2/24	0
hiragasy ♯1	3/30 (**10**%)	0	27/30	3/30 (**10**%)	0	27/30
hiragasy ♯2	26/26 (**100**%)	0	0	13/13 (**100**%)	0	0
hiragasy ♯3	7/11 (**64**%)	0	4/11	7/11 (**64**%)	0	4/11
hiragasy ♯4	6/8 (**75**%)	2/8	0	6/8 (**75**%)	2/8	0
hiragasy ♯5	8/10 (**80**%)	0	2/10	4/5 (**80**%)	0	2/10
chancre	18/18 (**100**%)	0	0	9/9 (**100**%)	0	0
aqua	36/60 (**60**%)	24/60	0	14/29 (**48**%)	15/29	0

Table 5.1: Results of the tests on generic video sequences.

to detect them and we had a perfect extraction of the two objects. In the second part instead, the men go out of the forest and they walk with the sky in the background. In this case, the motion vectors of the background are not uniform and it is not possible to correctly estimate the camera motion and consequently it is not possible to correctly extract the motion masks and the foreground objects. In the sequence 'hiragasy ♯1' an error occurred in the camera motion estimation; in addition the object present is very small and the outliers generated by its movement in the sequence cannot be distinguished from the normal noise associated to the sequence motion vectors. In the sequence 'aqua' the detected objects have been found at the beginning of the sequence: in fact, in this sequence the camera zoom out and the objects become very small, so in this sequence, the objects at the beginning of the sequence were correctly detected but when their dimensions are the same of the ones of few macro-blocks they could not be detected.

We have noticed, by evaluating the obtained results that our algorithm gives very good results in general, but if, for any reason the motion vectors of the background are too noisy, it becomes impossible to correctly estimate the camera motion and consequently extract the foreground objects.

In the following pages other results extracted from the analyzed video sequences are presented. In each group of images the total number of processed frames for each sequence is presented; the computation time is given as well: to compute these times we have considered the entire process including partially decoding, camera motion estimation and all the foreground object extraction process except for the filtering with quadric surfaces. In fact, this last step cannot be executed in real time because it needs all the sequence computed before.

For one of the sequences proposed, we compute also the 3D shape with the quadric function to show that it is possible to use it even when all the objects have been detected to obtain the object approximate elliptic shape (see Figure 5.16).

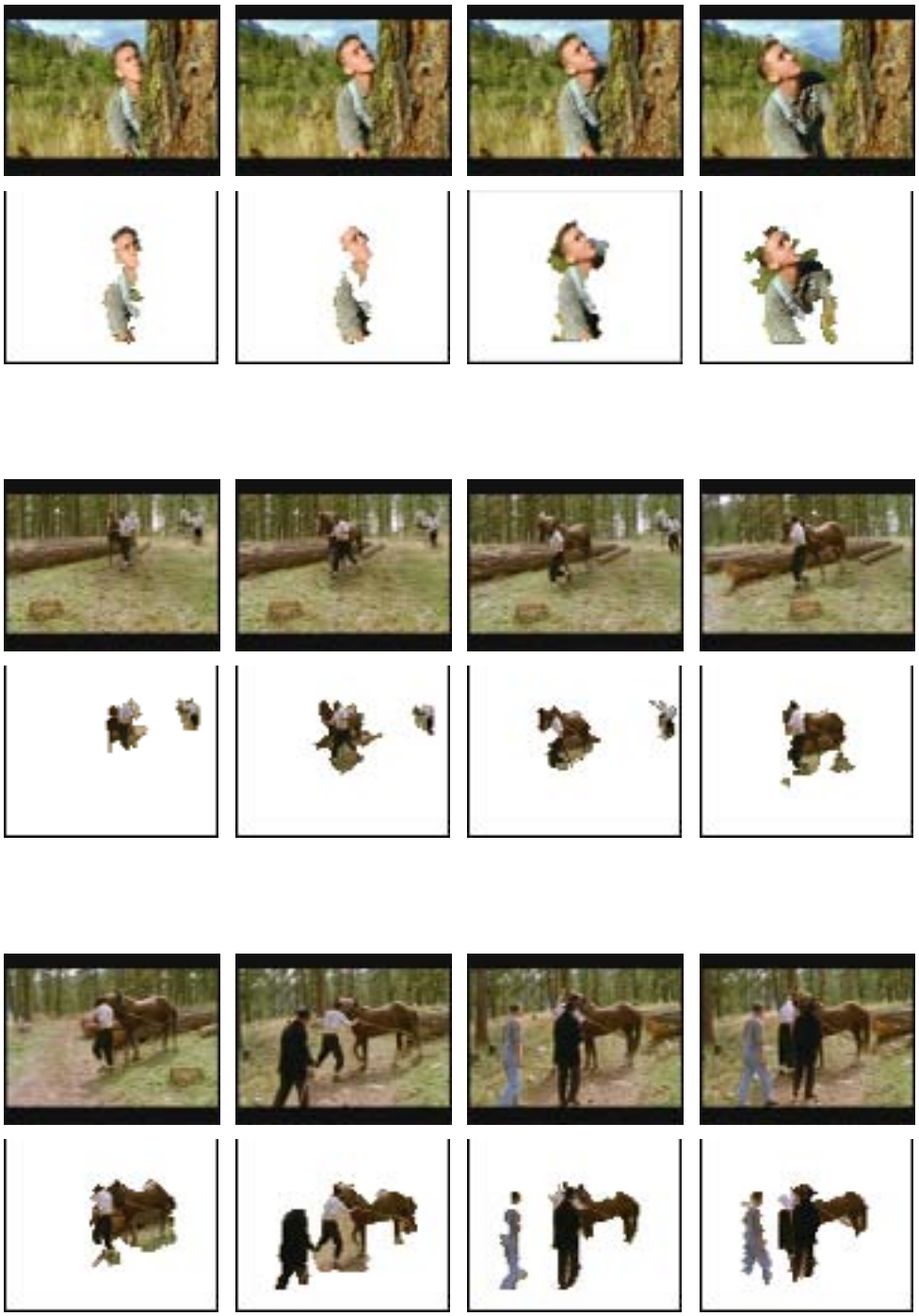


Figure 5.12: Some results from the video 'De l'arbre à l'ouvrage' (3600 frames, 140 seconds of processing time).

Figure 5.13: Some results from the video 'aquaculture' (435 frames, 16.9 seconds of processing time).

Figure 5.14: Some results from the video 'chancre' (135 frames, 5.25 seconds of processing time).

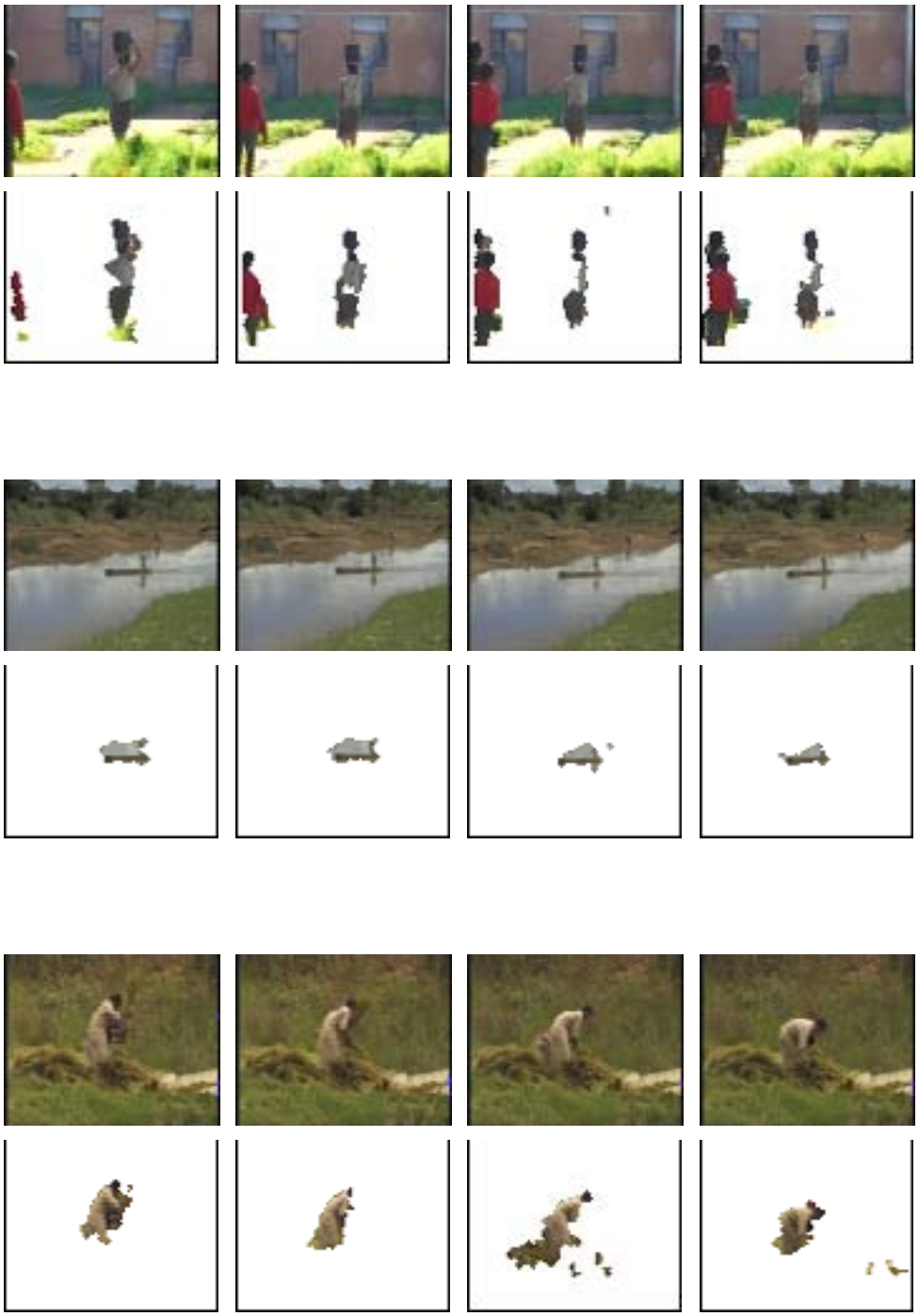
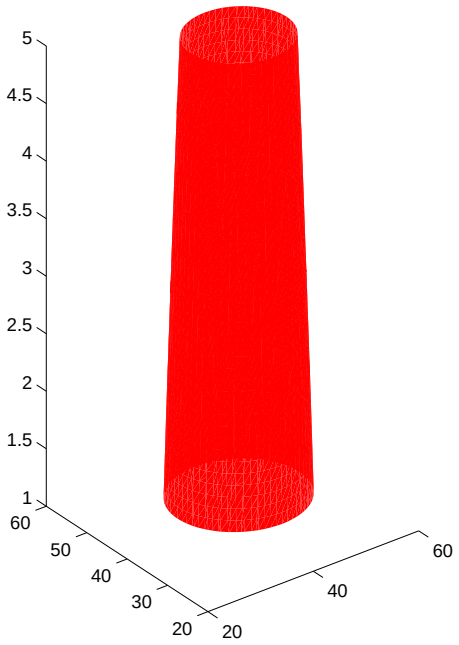


Figure 5.15: Some results from the video 'hiragasy' (1575 frames, 61.3 seconds of processing time).

(a) (b) (c) (d)



(e)

Figure 5.16: An example of tube construction for one of the sequences 'hiragasy'. In a)-d) the section of the computed quadric is overlapped to the sequence I frames; in e) the 3D shape is presented.

5.4 Cartoons

To test our algorithm we have also applied our method to some cartoons. This kind of video is very different from what we have called generic video sequences. In fact cartoons and most of the artificial video sequences present many flat zones and so too many motion vectors are erroneously estimated and sometimes the estimated camera motion becomes very different from the real one. In Figure 5.17 we propose an example of these noisy motion vectors: in a) the motion vectors corresponding to the P frames in b) are shown; in c) the macro-blocks that differ from estimated camera motion are represented; it is possible to see that, starting from these macro-blocks, it is not possible to obtain reliable motion masks, and the consequence is that we are not able to separate the foreground from the background.

In the cartoons that we have analyzed, we have found some cartoons that are textured like a generic video but for the most part they are not. We will present the results of our analysis as we have done for generic video sequences.

In the following section we will show the result from a cartoon sequence.

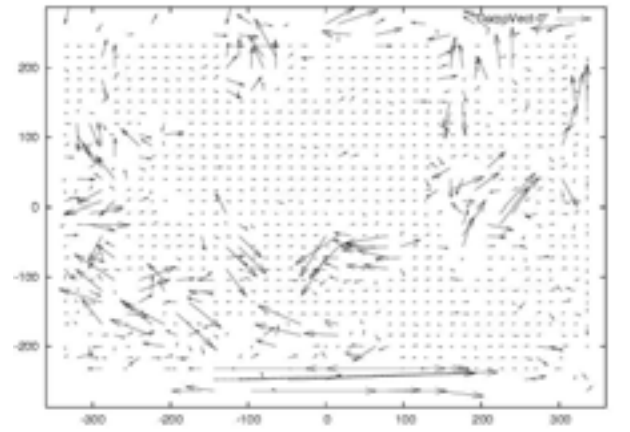
5.4.1 Ferrailles

In this section we will analyze a sequence extracted from the cartoon "ferrailles" (Canal+ ®). In Figures 5.18 the I frame of the sequence are presented at DC resolution.

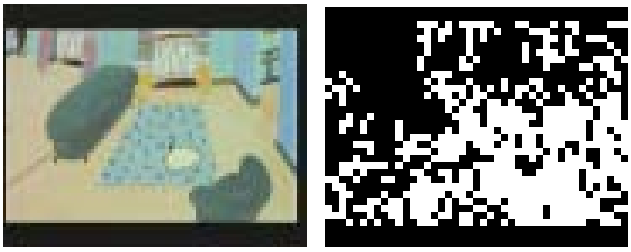
This is an indoor sequence and it is well textured, many details are present and there are no flat zones, so we can imagine that it will not be difficult to correctly estimate camera motion and obtain a performance of the algorithm similar to the one shown for the videos presented in table 5.1. In Figure 5.19 we show the outliers found in some of the P frames of this sequence. We can see that even if there is noise, it is possible to identify the foreground objects. The subsequent process of filtering will contribute to eliminate this noise.

After all the processing of filtering and interpolation we extract the object masks for the I frames. In Figure 5.20 some of these object masks are presented. Even if in the outlier masks of the P frames some noise was present, our filtering processes allow to correctly eliminate this noise without affecting the foreground object masks. In Figure 5.21 the DC images with the motion masks overlapped in transparency are proposed to show that moving objects have been correctly detected.

Now that we have extracted object masks for the sequence, we have to segment DC images to refine the result as we have already shown previously. In Figure 5.22



(a)



(b) (c)

Figure 5.17: in a) the motion vectors extracted from a P frame of the cartoon 'Le chat d'appartement', in b) the P frame and in c) the outliers corresponding to the motion vectors in a).

Figure 5.18: A sequence extracted from the cartoon ‘ferrailles’.

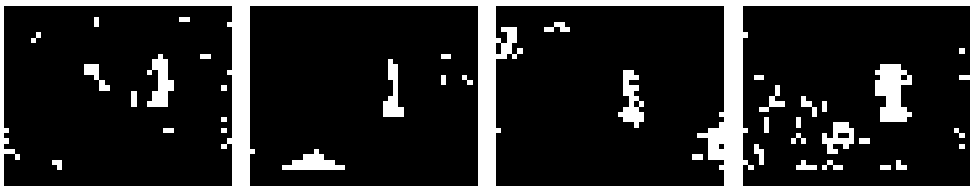


Figure 5.19: The extracted outliers from some P frames.

Figure 5.20: Extracted object masks from some I frames.

Figure 5.21: The sequence of DC images with object masks in transparency.

we show, this time for the cartoon sequence, all steps of proposed color morphological segmentation algorithm. In Figure 5.22 a) an I frame from the cartoon sequence is presented; in 5.22 b) the luminance component of the DC image is shown. In Figure 5.22 c) the same luminance component after the preprocessing filtering is shown; it is possible to see that the processed image is more homogeneous and so it will be easier to segment. Finally in 5.22 d) the segmentation result is proposed.

After having segmented all the DC frames, the foreground objects are obtained by merging this segmentation result with the motion masks. In Figure 5.23 we present the foreground objects extracted from the sequence.

5.4.2 Cartoon results

We can see that in the case of the cartoon 'ferrailles', the objects have been correctly detected, but this is in general not true for all type of cartoons. In fact we have analyzed many other cartoons and we have seen that in general our method does not give good results for this kind of videos. In table 5.2 we summarize the results

Figure 5.22: Color segmentation process for an I frame of the sequence.

Figure 5.23: Extracted foreground objects from the sequence 'ferrailles'.

for a set of cartoons.

Analyzing the table we can conclude that our method does not perform well in the case of cartoons; the reason is that many flat zones are present and in these cases many motion vectors are erroneous. Moreover, as our spatiotemporal filtering can correct isolated errors, it cannot be used in this case because of the large number of initial wrong detection.

Even if objects are detected, there often is a high number of imprecise object boundaries due to the presence of large flat zones.

In the following pages we propose further results extracted from the set of cartoons.

Sequence	% detected objects	missd.	overd.	% correct frames	missd.	overd.
ferrailles	9/11 (**82**%)	2/11	0	9/11 (**82**%)	2/11	0
escapade	8/10 (**80**%)	1/10	1/10	8/10 (**80**%)	1/10	1/10
boutdumonde	0/12 (**0**%)	10/12	2/12	0/12 (**0**%)	10/12	2/12
casa	11/50 (**22**%)	35/50	4/50	0/24 (**0**%)	24/24	0
françois	7/26 (**27**%)	17/26	2/26	7/26 (**27**%)	17/26	2/26
bouche	13/20 (**65**%)	7/20	0	13/20 (**65**%)	7/20	0
chat	2/12 (**17**%)	6/12	4/12	2/12 (**17**%)	6/12	4/12
moine	7/26 (**27**%)	17/26	2/26	2/14 (**14**%)	10/14	2/14
roman ♯1	15/50 (**30**%)	6/50	29/50	15/50 (**30**%)	6/50	29/50
roman ♯2	13/24 (**54**%)	6/24	5/24	13/24 (**54**%)	6/24	5/24

Table 5.2: Result of the application of the method on cartoon sequences.

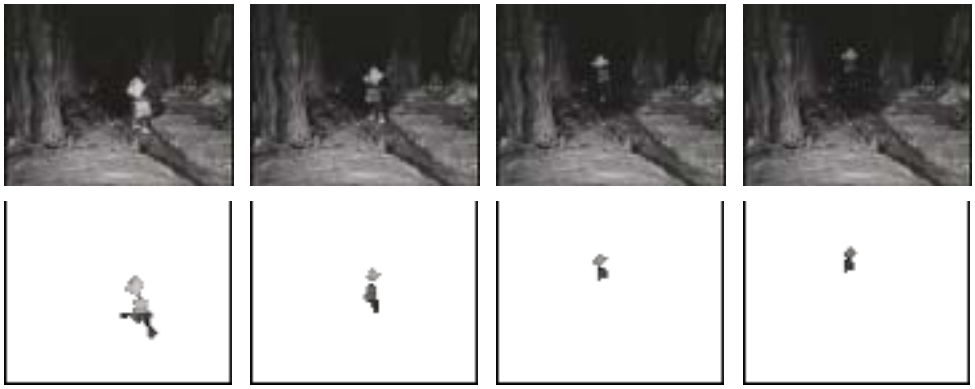


Figure 5.24: Some results from the cartoon 'Petite escapade' (150 frames, 5.8 seconds of processing time).

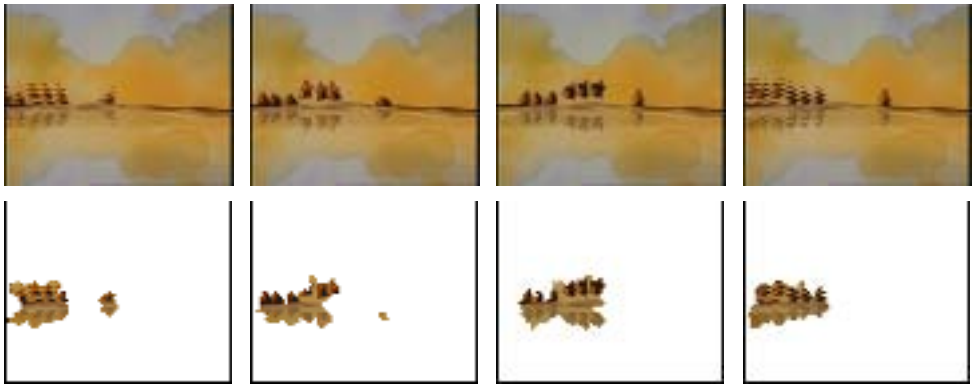


Figure 5.25: Some results from the cartoon 'Le moine et le poisson' (210 frames, 8.2 seconds of processing time).

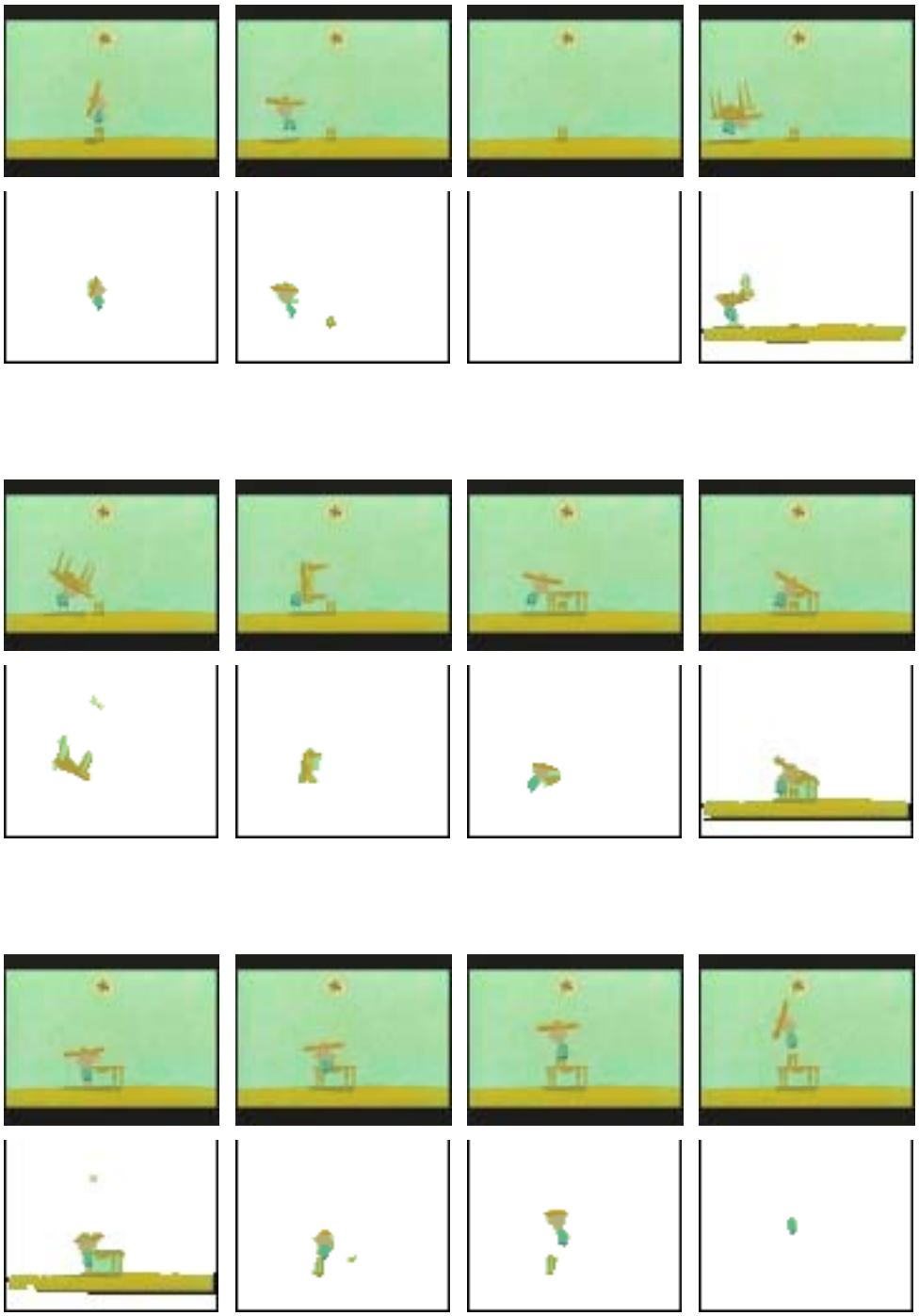


Figure 5.26: Some results from the cartoon 'Le trop petit prince' (255 frames, 9.9 seconds of processing time).

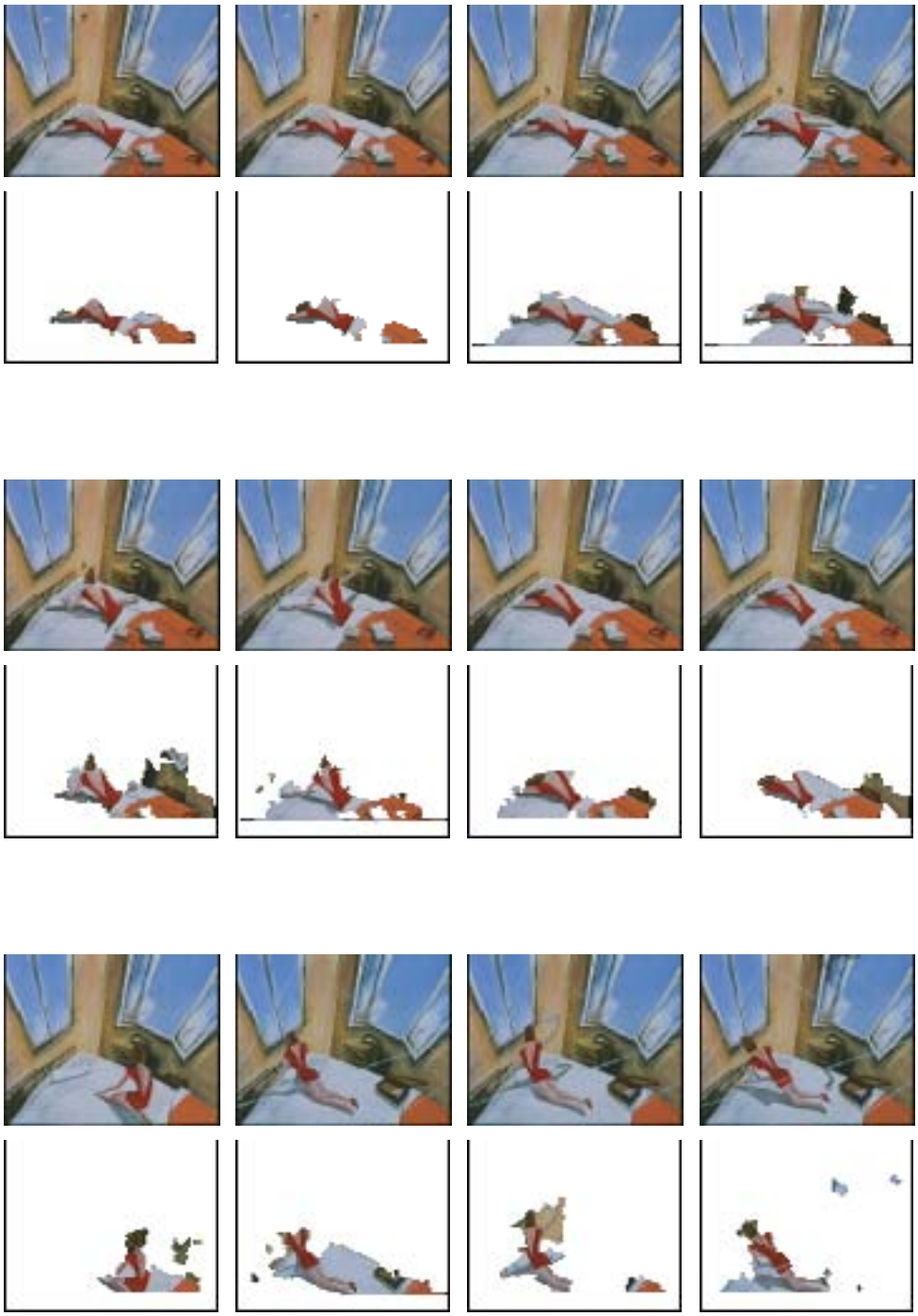


Figure 5.27: Some results from the cartoon 'le roman de mon ame' (1110 frames, 43.2 seconds of processing time).

5.5 Conclusions

In this chapter we have presented the results obtained with our method. The sequences used for the tests are of two types: generic video sequences, that is natural videos or films, and cartoons. In the first type of videos the method performs very well. Generally speaking, for a generic video sequence, our goal of detection of moving objects in I-frames is attained. The method shows some imprecision in the object boundaries due to the rough data used and the macro-block resolution. A slight over-detection can be observed due to the presence of false MPEG2 vectors in the case of strongly textured image. In all I and P frames, the outliers due to camera motion on the frame borders were correctly removed even if the camera motion was not a pure translation. The algorithm has also been tested under limit conditions, that is to say with objects so near to the camera that they cover a large part of the background (30%) and in the absence of foreground objects. In the first case the robustness of the motion detection algorithm has led to a correct extraction of the camera motion parameters and consequently to a correct detection of the objects in the foreground. In the second case, even if the noise present on MPEG2 motion vectors has caused the presence of some outliers, the flat zone filtering has permitted to classify these outliers as pure noise.

We obtain good results without the spatio-temporal filtering, that is in real time, and in many cases the spatiotemporal filtering can help to improve the performance further. In case of cartoons instead, our method does not perform well; the reason is that the motion vectors extracted from the MPEG2 decoder are very noisy due to numerous flat zones. In this case the estimated camera motion is not reliable and so the computed object motion mask. In this case the spatio-temporal filtering cannot help to obtain better results since to use it efficiently, it is necessary to start from some good results. In this case, to have better results it is necessary to change the processing method and to use a segmentation algorithm that is not based on motion characteristics as in our method. Nevertheless, while the compressed streams in this case do not supply a reliable motion information, in the perspective of this work, a specific motion re-estimation on uncompressed stream can be performed. As we proposed a method for filtering of flat areas, then we can re-estimate in a frame adaptive manner on frame containing a lot of flat zones.

Conclusions

In this thesis work we have presented our approach to foreground moving object extraction. The problem of object extraction, background/foreground separation and region characteristic extraction has been faced by many researchers in the last 10 years. In fact the new standards MPEG4 and MPEG7 proposed new concepts for compressing and indexing the multimedia content. MPEG4 for example, besides the traditional segmentation methodology that compresses information by subdividing the image into blocks, introduces new standardized methods to compress the images by subdividing them into objects they are part of. Moreover it allows different degrees of compression between foreground objects and background.

MPEG7, instead, is not a compression standard but it defines among other things the way to describe regions and objects in a video or image. For example it standardizes how to describe the shape of an object, its motion, its trajectory, its color and many other things.

As both standards do not give any indication on how extract the foreground objects or their characteristics, many works have developed different methods to achieve the result. Many of these works have the objective to extract the foreground objects to obtain foreground/background separation necessary to use some of the compression techniques proposed by the MPEG4 standard. In this case precise object boundaries are needed so these works perform object segmentation in the pixel domain with high computational cost. Other works instead, oriented towards indexing and retrieval rather than coding, try to extract foreground objects characteristics useful to be included for example in a platform dedicated to indexing and retrieval tools. In this case, if the object extraction algorithm is included in the retrieval platform, it becomes necessary to extract object information quite rapidly. So these algorithms are real-time oriented, that means that a reduced execution time become one of the most important things, in particular with respect to a very accurate boundary detection.

Our work belongs to these second family of algorithms. Our objective was to extract foreground objects and their characteristics from an MPEG2 video sequence and to reach the objective in real-time. For this purpose, we have decided to work in a "rough indexing" context. This means that we have used rough data, in our case MPEG2 motion vectors and reduced resolution images, to reduce the computational cost. This has reduced consequently the total processing time since our rough data can be directly extracted from the compressed bit-stream without decoding the sequence.

The segmentation algorithm can be classified as a combined motion and color based approach. In fact, we have first performed a foreground/background separation based on motion information and then applied a color refinement to extract foreground objects, at I frame resolution.

We have used motion vectors from an MPEG2 compressed stream to compute the global camera motion. It has been possible to estimate the camera motion without a dense vector field thanks to a robust estimation algorithm. Once the camera motion model has been estimated, we have separated the macro-block that presented motion vectors inconsistent with the camera motion model. But many regions did not reflect the correct object position and dimension; this may not necessarily be due to errors in the camera motion estimation algorithm. It may be a consequence of inadequate motion estimation by the MPEG2 encoder. We have reduced the problem by an appropriate filtering process and a segmentation of the motion masks.

After having obtained the post-processed motion masks, motion masks have been interpolated for I frames, since only color information is available for such frames. By spatial interpolation we have enabled motion masks to be merged with the results of a color segmentation applied to the I frames. For the color segmentation process, we had to perform a preprocessing as well, since given the low resolution data, the image quality was not so good and many regions exhibited high gradient values so that appropriate object boundaries detection using the color gradient information was difficult to achieve. After the preprocessing stage, a modified watershed algorithm enabled to subdivide the image into homogeneous color regions. This result was then used to refine the interpolated motion masks given by the motion analysis.

Thanks to the used rough data and the low resolution, the algorithm we have described can be executed in real-time, independently from the number of present objects. This part of the algorithm has been tested on two types of video, classified as general video sequences and cartoons.

In the case of general video sequences the results are promising because more

than 80% of objets are detected. The motion masks are, in the majority of cases, very precise thanks to the robustness of the camera motion estimation. At the same time the preprocessing performed on the DC images before the color segmentation stage guarantees a good level of accuracy in color segmentation. Sometimes, errors in color segmentation may appear but these represent isolated cases. The cases of mismatch we have found were due, in the majority of cases, to the lack of motion. In this case in fact the foreground object presented the same motion model of the camera one and thus would not be detected. Another cause of error in detection could also be a sudden luminance change. In this case the motion vectors of MPEG2 coding stream are erroneous and consequently also the result we obtain. In both cases no changes in the algorithm can prevent this type of errors, because it is due to erroneous input data.

In the case of cartoons instead, the results are very different because it is very difficult to have correct motion masks. In fact, in cartoons all the objects lack textured areas, so that the motion vectors extracted from compressed streams are very often erroneous. Consequently, as the MPEG2 motion vectors are not reliable, even if the camera motion estimation algorithm is very robust, it fails to provide the corrected results. This applies in general also to computer generated scenes. In all these cases no post processing can fix the problem.

On the contrary, in the case of general video sequences where errors appear most often in isolated frames, we have developed a spatio-temporal filtering that can correct the object considering the neighbouring results in time. This filter is a sort of tube, for example a cylinder or a cone or another generic quadric function, whose sections correspond to the ellipses that approximate the object shapes. In this case we are not able to exactly extract the missing object, but we can get the ellipse that best approximates it, by observing the tube section at the moment of time corresponding to the frame of interest. This kind of filtering technique is very reliable if isolated errors are present in a sequence where the majority of objects have been successfully detected. It is clear that if, for any reason, the MPEG2 motion vectors are very noisy and all the sequence is affected with errors this processing technique will not be not sufficient to fix the problem.

This kind of process cannot be executed in real-time since it is necessary to have all the extracted objects. Moreover as it builds a quadric function for every object, the processing time depends on the number of objects.

In conclusion the proposed method is very promising since, thanks to the robustness of motion camera estimation and of the combined motion and color segmenta-

tion, it has a high success rate. Without considering the spatio-temporal filtering, working in compressed domain and using rough data allows to obtain very good results in real-time.

To obtain better results and to complete the method making it appropriate for MPEG7 oriented applications, we suggest some possible improvements or some applications of the proposed algorithm:

- The presented algorithm produces results for only I frames, but all the motion masks are calculated for P frames as well, so it can be reasonable to think to extend this algorithm also to P frames. Moreover by increasing the number of results, more objects will be available to perform better spatio-temporal filtering, thus making it more reliable;

- this work did not consider the case of overlapping objects, but the information necessary to establish which object is closer to the camera is available, and this information can also make the spatio-temporal filtering process more reliable.

- a visualization platform can be implemented developing the instrument that we have used to validate the results. Many descriptors can be visualized along time and it could be interesting to find a few descriptors which, when visualized jointly, may bring some further semantic understanding of the context.

- as the temporal filtering is expensive in terms of execution time, a solution can be to execute the filtering only for the sequences where we need it. So it can be useful to integrate the filtering processing in the visualization system that we used as a validation method in chapter 5. In this case during the evaluation of the result if an error is detected the spatio-temporal filtering may be used to try to fix the problem.

References

[1] "Information technology: generic coding of moving pictures and associated audio information," 1995. MPEG Requirements Group, ISO/IEC 13818.

[2] I. Richardson, *H.264 and MPEG-4 Video Compression. Video Coding for next-generation code*. Wiley, 2003.

[3] B. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG-7. Multimedia Content Description Interface*. Wiley, 2002.

[4] "Information technology: coding of audio-visual objects- part 2: Visual," 2001.

[5] N. Brady, "Mpeg-4 standardized methods for the compression of arbitrarily shaped video objects," *IEEE Trans. Circuits and Systems for Video Technology*, pp. 1170–189, 1999.

[6] "Introduction to mpeg7." MPEG Requirements Group, Doc. ISO/MPEG N4325, July 2001. MPEG Meeting, Sidney.

[7] "Mpeg7 overview." MPEG Requirements Group, Doc. ISO/MPEG N4317, July 2001. MPEG Meeting, Sidney.

[8] "Mpeg4 overview." MPEG Requirements Group, Doc. ISO/MPEG N4316, July 2001. MPEG Meeting, Sidney.

[9] F. Pereira, "Mpeg4: why, what, how and when? tutorial issue on the mpeg4 standard," *Signal Processing: Image Communication*, vol. 15, pp. 271–279, 1999.

[10] "Mpeg7 requirements." MPEG Requirements Group, Doc. ISO/MPEG N4320, July 2001. MPEG Meeting, Sidney.

[11] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1190–1203, June 1997. San Juan, Puerto Rico.

[12] P. Salembier and F. Marques, "Region-based representations of image and video: segmentation tools for multimedia services," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 9, pp. 1147–1169, December 1999.

[13] E. Saber, A. Takalp, and G. Bozdagi, "Fusion of color and edge information for improved segmentation an edge linking," in *Proc. of International Conference on Acoustics, Speech and Signal Processing*, (Atlanta, GA), pp. 2176–2179, May 1996.

[14] M. Tabb and N. Ahuja, "Multiscale image segmentation by integrated edge and region detection," *IEEE Transaction on Image Processing*, vol. 6, pp. 642–655, May 1997.

[15] L. Lucchese and S. Mitra, "Unsupervised segmentation of color images based on k-means clustering in the chromaticity planes," in *Proc. of Content-based Access of Image and Video Libraries*, pp. 74–78, 1999.

[16] S. Ji and H. Park, "Image segmentation of color image based on region coherency," in *Proc. of International Conference on Image Processing*, vol. 1, pp. 80–83, 1998.

[17] J. Wang and E. Adelson, "Representing moving imaging with layers," *IEEE Transaction on Image Processing*, vol. 3, pp. 625–638, September 1994.

[18] S. Ayer and H. Sawhney, "Layered representation of motion video using robust maximum liklihood estimation of mixture models and mdl encoding," *Proc. 5th International Conference on Computer Vision*, pp. 777–784, June 1995. Cambridge, MA.

[19] T. Darrell and A. Pentland, "Cooperative robust estimation using layers of support," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 474–487, May 1995.

[20] G. Borshukov, G. Bozdagi, Y. Altunbasak, and A. Tekalp, "Motion segmentation by multistage affine classification," *IEEE Transaction on Image Processing*, vol. 9, pp. 1259–1268, December 1999.

[21] F. Meyer and P. Bouthemy, "Region based tracking using affine motion models in long image sequences," *CVGIP: Image Understanding*, vol. 60, pp. 119–140, September 1994.

[22] L. Torres, D. Garcia, and A. Mates, "A robust motion estimation and segmentation approach to represent moving images with layers," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp. 2981–2984, 1997.

[23] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," *International Journal on Computer Vision*, vol. 7, pp. 321–331, 1988.

[24] B. Bascle, P. Bouthemy, R. Deriche, and F. Meyer, "Tracking complex primitives in image sequence," in *Proc. of 12th International Conference on Pattern Recognition*, pp. 426–431, 1994.

[25] A. Blake, R. Curwen, and A. Zisserman, "A framework for spatiotemporal control in the tracking of visual contours," *International Journal on Computer Vision*, vol. 11, no. 2, pp. 127–145, 1993.

[26] A. M. Tekalp, *Digital Video Processing*, ch. 3. Prentice Hall, 1995.

[27] T. Meier and K. Ngan, "Video segmentation for content-based coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 1190–1203, December 1999.

[28] D. Zhong and S. Chang, "An integrated approach for content-based video object segmentation and retrieval," *IEEE Transations on Circuits and Systems for Video Technology*, vol. 9, pp. 1259–1268, December 1999.

[29] C. Gu and M. Lee, "Sematic video object segmentation and tracking using mathemtical morphology and perspective motion model," in *IEEE International Conference on Image Processing*, vol. 2, (Santa Barbara, California), pp. 514–517, October 1997.

[30] M. Kim, J. Choi, D. Kim, H. Lee, M. Lee, C. Ahn, and Y. Ho, "A vop generation tool: automatic segmentation of moving objects in image sequences based on spatio-temporal information," *IEEE Transaction on Circuits Systems and Video Technology*, vol. 9, pp. 1216–1226, December 1999.

[31] F. Dufaux, F. Moscheni, and A. Lippman, "Spatiotemporal segmentation based on motion and static segmentation," in *Proc. IEEE Conference on Image Processing*, vol. 1, pp. 306–309, October 1995.

[32] M. Gelgon and P. Bouthemy, "A region level graph labeling approach to motion based segmentation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, (San Juan, Puerto Rico), pp. 514–519, June 1997.

[33] I. Patras, E. Hendriks, and R. Lagendijk, "Video segmentation by map labeling of watershed," tech rep. ict -00-01, Delft University of Technology, November 2000.

[34] M. Dubuisson and A. Jain, "Contour extraction of moving objects in complex outdoor scenes," *International Journal of Computer Vision*, vol. 14, pp. 83–105, 1995.

[35] K. Zhang, M. Bober, and J. Kittler, "Motion based image segmentation for video coding," in *Proc. International Conference on Image Processing*, (Los Alamitos, CA), pp. 476–479, 1995.

[36] J. Choi, S. Lee, and S. Kim, "Video segmentation based on spatial and temporal information," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp. 2661–2664, 1997.

[37] J. Benois-Pineau, F. Morier, D. Barba, and H. Sanson, "Hierarchical segmentation of video sequences for content manipulation and adaptive coding," *Signal Processing: special issue on Video sequence segmentation for content processing and manipulation*, no. 66, pp. 181–201, 1998.

[38] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679–698, November 1986.

[39] E. Dijkstra, "A note on two problems in connection with graphs," *Journal of Numerical Mathematics*, vol. 1, pp. 269–271, 1959.

[40] S. Jehan-Besson, M. Barlaud, and G. Aubert, "DREAM2S: Defomable regions driven by an eulerian accurate minimizationmethod for image and video segmentation," *International Journal of computer vision*, no. 53, pp. 45–70, 2003.

[41] S. Jehan-Besson, M. Barlaud, and G. Aubert, "A 3-step algorithm using region-based active contours for video objects detection," *EURASIP Journal on Applied Signal Processing*, vol. 2002, pp. 572–581, june 2002.

[42] S. Jehan-Besson, M. Barlaud, and G. Aubert, "Region-based active contours for video object segmentation with camera compensation," in *Proc. of IEEE International Conference on Image Processing*, (Barcelone), September 2003.

[43] F. Moscheni, S. Bhatacharje, and M. Kunt, "Spatiotemporal segmentation based on region merging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 897–915, September 1998.

[44] M. Bierling, "Displacement estimation by hierarchical block matching," in *Proc. of SPIE Visual Communication and Image Processing*, vol. 1001, pp. 942–951, 1988.

[45] J. Choi, M. Kim, M. Lee, and C. Ahn, "User-assisted video object segmentation by multiple object tracking," tech. rep., ISO/IEC JTC1/SC29/WG11 MPEG98/m3349, Tokyo, Japan, March 1998.

[46] C. Gu and M. Lee, "Semiautomatic segmentation and tracking of semantic video objects," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 572–584, September 1998.

[47] S. Colonnese and G. Russo, "User interaction modes in semi-automatic segmentation: development of a flexible graphical user interface in java," tech. rep., ISO/IEC JTC1/SC29/WG11 MPEG98/m3320, Tokyo, Japan, March 1998.

[48] A. Mahboubi, J. Benois-Pineau, and D. Barba, "Tracking of objects in video scebes with time varying content," *EURASIP Journal on applied signal processing*, vol. 6, pp. 582–593, 2002.

[49] N. Brady and N. O'Connor, "Object detection and tracking using an em-based motion estimation and segmentation framework," in *Proc. IEEE International Conference on Image Processing*, pp. 925–928, 1996.

[50] D. Elias, *The motion based segmentation of image sequences*. PhD thesis, Trinity college, University of Cambridge, August 1998.

[51] N. Vasconcelos and A. Lippman, "Empirical bayesian em-based motion segmentation," in *Proc. IEEE CVPR*, pp. 527–532, 1997.

[52] P. Torr, R. Szelisky, and P. Anandan, "An integrated bayesan approach to layer extraction from image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 297–303, March 2001.

[53] G. Wu and T. Reed, "Image sequence processing using spatiotemporal segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 798–807, August 1999.

[54] E. Izquierdo and M. Ghanbari, "Ket components for an advanced segmentation system," *IEEE Transactions on Multimedia*, vol. 4, pp. 97–113, March 2002.

[55] O. Sukmarg and K. Rao, "Fast object detection and segmentation in mpeg compressed domain," in *Proc. IEEE TENCON*, (Kuala Lumpur, Malaysia), September 2000.

[56] R. D. Queiroz, "Processing jpeg-compressed images and documents," *IEEE Transactions on Image Processing*, vol. 7, pp. 1661–1672, December 1998.

[57] H. Wang and S. Chang, "A highly efficient system for automatic face region detection in mpeg video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 615–628, August 1998.

[58] J. Benois-Pineau and A. Khrennikov, "Image segmentation in compressed domain by clustering methods with euclidean and p-adic metrics," in *Proc. IPMU*, (Annecy, France), July 2002.

[59] R. V. Babu and F. Ramakrishnan, "Video object segmentation: a compressed domain approach," *IEEE Transactions on Circuits and Systems for Video Technologies*, vol. 14, no. 4, pp. 462–474, 2004.

[60] D. Sandwell and T. David, "Biharmonic spline interpolation of geos-3 and seasat altimeter data," *Geophysical Research Letters*, vol. 14, no. 2, pp. 139–142, 1987.

[61] L. Favalli, A. Mecocci, and F. Moschetti, "Object tracking for retrieval applications in mpeg2," *IEEE Transactions on Circuits and Systems for Video Technologies*, pp. 427–432, April 2000.

[62] F. Porikli, "Real-time vido object segmentation for mpeg encoded video sequences," tech. rep., Mitsubishi Electric Research Laboratories, Cambridge, USA, March 2004.

[63] V. Mezaris, I. Kompatsiaris, N. Boulgouris, and M. Strintzis, "Real-time compressed-domain spatiotemporal segmentation and ontologies for video indexing nad retrieval," *IEEE Transactions on Circuits and Systems for Video Technologies*, vol. 14, pp. 606–621, May 2004.

[64] J. Faquier and N. Boujemaa, "Region-based retrieval: coarse segmentation with fine signature," *Proc. of IEEE International Conference on Image Processing*, 2002.

[65] N. AbouGhazaleh and Y. ElGamal, "Compressed video indexing based on object motion," *Proc. of Visual Communication and Image Processing*, pp. 986–993, June 2000. Perth, Australia.

[66] Y. Tan, D. Saur, S. Kulkarni, and P. Ramadge, "Rapid estimation of camera motion from compressed video with application to video annotation," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 1, pp. 133–146, 2000.

[67] M. Durik and J. Benois-Pineau, "Robust motion characterisation for video indexing based on mpeg2 optical flow," in *Proc. of Content Based Multimedia Indexing*, (Brescia, Italy), September 2001.

[68] C. Stiller and J. Konrad, "Estimating motion in image sequences: a tutorial on modeling and computation of 2d motion," *IEEE Signal Processing Magazine*, vol. 16, pp. 70–91, July 1999.

[69] F. Dufaux and J. Konrad, "Robust, efficient and fast global motion estimation for video coding," *IEEE Transaction on Image Processing*, vol. 9, pp. 497–501, March 2000.

[70] J.-M. Odobez and P. Bouthemy, "Estimation robuste multi-échelle de modèles paramétrés de mouvement sur des scènes complexes," in *Reconnaissance des Formes et Intelligence Artificielle, RFIA'94*, January 1994.

[71] P. Bouthemy, M. Geldon, and F. Ganasia, "A unified approach to shot change detection and camera motion characterisation," *IEEE Circuits and Systems for Video Technology*, vol. 9, pp. 1030–1044, October 1999.

[72] S. Benini, E. Boniotti, R. Leonardi, and A. Signoroni, "Interactive segmentation of biomedical images and volume using connected operators," in *Proc. of International Conference on Image Processing*, Sep 2000. Vancouver, Canada.

[73] J. Benois-Pineau and H. Nicolas, "A new method for region based depth ordering in a video sequence: application to frame interpolation," *Journal of Video Communication and Video Representation*, vol. 13, pp. 363–385.

[74] B.-L. Yeo and B. Liu, "On the extraction of dc sequence from mpeg compressed video," in *Proc. Of International Conference on Image Processing*, vol. 2, (Washington DC), 1995.

[75] A. Mahboubi, J. Benois-Pineau, and D. Barba, "Suivi et indexation des objets dans des séquences vidéo avec la mise-à-jour par confirmation retrograde," *CORESA'2001*, November 2001. Dijon, France.

[76] C. Erdem, F. Ernst, A. Redert, and E. Hendriks, "Temporal stabilisation of video object segmentation for 3d-tv applications," in *Proc. of International Conference on Image Processing*, (Singapore), October 2004.

[77] M. Najim, *Modélisation et identification en traitement du signal*, ch. 2, pp. 35–47. Masson Ed., 1988.

[78] D. Hilbert and S. Cohn-Vossen, *Geometry and the Imagination, The Second-Order Surfaces*, ch. 3, pp. 12–19. New York: Chelsea, 1999.

[79] R. Mollin, *Quadrics*. CRC Press, 1995.

[80] A. Agnetis, "Introduzione all'ottimizzazione vincolata," Università di Siena.

[81] "Mpeg7 interoperability, conformance testing ad profiling." MPEG Requirements Group, Doc. ISO/MPEG N4039, Mar 2001. MPEG Meeting, Singapore.

[82] D. Zhong and S. Chang, "Content-based video object segmentation and retrieval," *IEEE TRansactions on Circuits and Systems for Video Technology*, vol. 9, pp. 1259–1268, December 1999.

[83] A. Dempster, N. Laird, and D. Rubin, "Maximum-likelihood from incomplete data via the em algorithm," *J. Roy. Stat. Soc. B*, vol. 39, pp. 1–38, 1997.

[84] R. Wang, H.-J. Zhang, and Y.-Q. Zhang, "A confidence measure based moving object extraction system built for compressed domain," *Proc. of IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 21–24, 2000.

[85] "www.mathworld.wolfram.com." Wolfram Research.

We don't need no education
We don't need no thought control [...]
All in all it's just another brick in the wall.
All in all you're just another brick in the wall.

Efficient Object Identification in Image Sequences for Content Indexing

Résumé : Les avancées en matière d'acquisition et de stockage des données ont conduit à la disponibilité de grandes bases de données vidéo numériques pour l'utilisateur grand-public. Néanmoins l'interaction avec les données multimédia et vidéo en particulier nécessite des outils de description, organisation et gestion de ces données vidéo. La majorité des documents multimédia sont produits sous forme compressée. De plus comme les utilisateurs demandent des réponses à leur requêtes les plus rapides possible, l'extraction d'information en temps réel même si celle-ci n'est pas très précise est devenue un objectif important. Basée sur des hypothèses formulées ci-dessus, une nouvelle tendance en matière d'analyse pour l'indexation multimédia est parue, elle peut être qualifiée comme paradigme de « rough indexing » ou « indexation grossière ». Nous pouvons inclure notre travail dans cette mouvance d'indexation rapide et approximative dans laquelle uniquement les données imprécises telles que les vecteurs du mouvement et les images DC sont exploitées pour produire une indexation fine des objets en avant-plan. Dans ce paradigme nous proposons de combiner à la fois les informations du mouvement et la segmentation basée-couleur pour extraire les objets pertinents des flux vidéo compressés. Il peut arriver que le mouvement d'un objet soit très similaire au mouvement de la caméra ou que l'objet soit statique, aucun objet ne sera alors détecté. Néanmoins, comme un objet ne peut raisonnablement pas apparaître et disparaitre durant une courte séquence d'images, nous proposons de filtrer les séquences d'objets le long de l'axe temporel. Pour ce faire nous proposons une modélisation de la séquence des objets par une super-quadrique. La méthode proposée donne des résultats prometteurs : les ratés de la détection ou les sur-détections peuvent être corrigés par la méthode de filtrage spatio-temporel proposée.

Discipline : Informatique

Mots-Clefs : estimation du mouvement sur des flux compressés, segmentation spatio-temporelle, modélisation par des surfaces quadriques.

Efficient Object Identification in Image Sequences for Content Indexing

Abstract : The advances in data capturing and storage have made large amounts of video data available for consumer applications. However, interacting with multimedia data, and video in particular, requires tools to describe, organize and manage video data. The majority of multimedia documents is provided in compressed form. Besides, as the users require their retrieval in the fastest way, extracting in formation in real time, even if it is not precise, has become an important objective. Based on the assumptions above, a new trend in analysis methods for indexing multimedia content has appeared which can be qualified as a rough indexing paradigm. In this context we can include our work where only rough data - that is motion vectors and DC images - have been used for fine indexing for foreground objects. In this paradigm we propose to combine motion information and color - based segmentation to extract meaningful foreground objects from compressed video streams. It may happen that the object motion is very much similar to that one of the camera, than no objects get detected. To overcome this, we propose to filter the sequences of objects along the time axis. To do this we propose to model the sequence of objects by a super-quadric. The proposed method gives promising results : isolated miss-detection or over-detection episodes can be corrected by proposed spatio-temporal filtering.

Discipline : Computer-Science

Keywords : motion estimation on compressed stream, spatio-temporal segmentation, modelling by super-quadrics.

LaBRI,
Université Bordeaux 1,
351 cours de la Libération,
33405 Talence Cedex (FRANCE).