FOREGROUND EXTRACTION IN VIDEO CONFERENCES USING MOTION FLOW ANALYSIS

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul
CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI
de către

ing. Mihai Făgădar–Cosma

Conducător ştiinţific:	prof.univ.dr.ing. Vladimir-Ioan Creţu
Referenţi ştiinţifici:	prof.univ.dr.ing. Sergiu Nedevschi
	prof.univ.dr.ing. Dumitru Dan Burdescu
	conf.univ.dr.ing. Mihai Victor Micea

Ziua susţinerii tezei: 17 decembrie 2012

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor

Universitatea „Politehnica" din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Foreword

This thesis is the result of a journey - a journey of knowledge and transformation that was sometimes hard but nevertheless beautiful. I could have never completed it alone and I am grateful to God for giving me the strength and for surrounding me with all the people and situations that made it possible.

I am deeply thankful to my scientific advisor and coordinator, prof. dr. eng. Vladimir Creţu, who has mentored and guided me step by step through this endeavor, alongside assoc. prof. dr. eng. Mihai Micea, who stood beside me throughout my scientific career both as a teacher and as a friend. Thanks to their continuous support, I always felt part of the big family that is the Department of Computer and Software Engineering of "Politehnica" University of Timisoara.

My research activity has benefited greatly from the advice and support of my colleagues from the Immersive Communications team at Alcatel-Lucent. By no means, this is a long list of people that would spread across many pages. Among them are my friends Miguel Casas-Sanchez, Mukul Dhankar and Moulay Fadili with whom I had countless exchanges during our coffee breaks together, my manager Paul Davies who gave me the opportunity to be part of a great team and Marwen Nouri, who introduced me to the amazing concept of Graph Cuts (that turned out to be the missing puzzle in my approach).

Last but not least, I turn towards my family, who surrounded me with love, understanding and unconditional support. I have a lot to thank for to my wife Marilena and my daughter Cristina, for their faith in me and for encouraging me to go further even during my strongest moments of doubt; to my parents for their relentless help and love that shaped me into what I am today, and to my grandparents for always having a word of wisdom when I needed one.

To them, and to all the wonderful people that accompanied me at some point along this path, I express my thanks and my gratitude.

The author,
December, 2012

To Marilena and Cristina,
with all my love

"… if a conclusion is not poetically balanced, it cannot be scientifically true"
-- Isaac Asimov

Făgădar–Cosma, Mihai

Foreground Extraction in Video Conferences using Motion Flow Analysis

Teze de doctorat ale UPT, Seria 14, Nr. 10, Editura Politehnica, 2012, 121 pagini, 44 figuri, 8 tabele.

Cuvinte cheie:
Extragerea prim-planului, videoconferinţă, fluxuri optice, graph cut, segmentarea mişcării, procesare video în timp real

Rezumat,
Teza abordează domeniul segmentării în timp real a elementelor de prim-plan din secvenţe video monoculare, cu accentul pe aplicaţii de tip videoconferinţă. Lucrarea propune o metodologie de segmentare bazată exclusiv pe analiza fluxurilor optice combinată cu analiza informaţiei de culoare şi contrast din cadrele video. Metoda se distinge faţă de alte abordări prin faptul că procesul de segmentare nu foloseşte modele definite a priori cu privire la structura scenei observate sau la cea a obiectelor din prim-plan şi prin eliminarea necesităţii fazelor de iniţializare sau antrenament. Teza introduce un algoritm de agregare a fluxurilor optice dense şi rare în scopul segmentării robuste şi cu precizie a zonelor aflate în mişcare. Rezultatul este supus unei tehnici de integrare temporală pentru a produce o imagine aproximativă a prim-planului expus în urma analizei fluxurilor optice. Ultima fază a metodei introduce un nou algoritm de segmentare nesupervizată de tip graph-cut, responsabil cu obţinerea imaginii precise, la nivel de pixel, a obiectelor din prim-planul secvenţei video.

TABLE OF CONTENTS

TABLE OF FIGURES

INDEX OF TABLES

TABLE OF ABBREVIATIONS

AEC	**A**utomatic **E**xposure **C**orrection
ARGB	**A**lpha, **R**ed, **G**reen and **B**lue color space
BG	**B**ack**g**round
CMRF	**C**oupled **M**arkov **R**andom **F**ield
CPU	**C**entral **P**rocessing **U**nit
EM	**E**xpectation-**M**aximization algorithm
FG	**F**ore**g**round
FPS	**F**rames **P**er **S**econd
GC	**G**raph **C**ut
GMM	**G**aussian **M**ixture **M**odel
GPU	**G**raphics **P**rocessing **U**nit
GT	**G**round-**T**ruth
GUI	**G**raphical **U**ser **I**nterface
HMM	**H**idden **M**arkov **M**odel
HS	The **H**orn-**S**chunck dense flow estimation method
HSV	**H**ue, **S**aturation and **V**alue color space
KLT	The **K**anade-**L**ucas-**T**omasi sparse feature tracker
LK	The **L**ucas-**K**anade dense flow estimation method
MAP	**M**aximum ***A P**osteriori*
MAV	**M**ean **A**nnoyance **V**alue
MoG	**M**ixture **o**f **G**aussians
MPE	**M**ean **P**ercentage **E**rror
MRF	**M**arkov **R**andom **F**ield
MSSIM	**M**ean **S**tructural **SIM**ilarity
OF	**O**ptical **F**low
PDF	**P**robability **D**ensity **F**unction
RGB	**R**ed, **G**reen and **B**lue color space
RT	**R**eal **T**ime
S-T	**S**patio-**T**emporal
SSIM	**S**tructural **SIM**ilarity
TSM	**T**emporally **S**table **M**ask
YUV	Luminance (**Y**), blue- (**U**) and red-(**V**) chrominance color space

ABSTRACT

New teleconferencing concepts such as immersive videoconferences have added a new dimension to remote collaboration, by bringing participants together in a common virtual space. In order to achieve this task, the conferencing system extracts the image of each person from the incoming video streams and translates it into the virtual space that is shared between participants. At its core, such a system must rely on autonomous foreground extraction methods that are capable of performing an accurate segmentation in real-time.

Stereo-based methods are well known for their ability to handle such tasks, but they require dedicated hardware. To make immersive systems available to the general public, there is a requirement for algorithms capable of handling monocular video streams captured in various conditions using common off-the-shelf hardware such as webcams. In this context, most of the accurate foreground / background segmentation methods known in literature rely either on some form of assumption related to the position of the person in the video stream, on a previously known background or on prior learning and training using manually labeled sequences. Given the virtually infinite set of environments and situations in which a conference participant can find himself, these methods are prone to producing inconsistent results from one case to another.

This thesis addresses the problem of foreground extraction in monocular video sequences by introducing a new method focused on eliminating the need for initial training as well as any a priori assumptions or knowledge related to the observed scene contents and background. Starting from accurate motion cues obtained through aggregation of dense and sparse optical flow information, the system builds a temporally stable mask (TSM) of foreground detected through motion. The temporal stability of the mask in absence of motion information is achieved through the use of image statistics and similarity measures. In the last stage, an heuristic algorithm combines the TSM and sparse optic flow information to generate the hard foreground and background constraints for a graph-cut algorithm, which produces the final, pixel-accurate segmentation.

The perceived quality of segmented foreground represents a key aspect for achieving a true immersive experience. The segmentation produced by the described technique is evaluated using a state-of-the-art perceptual metric in order to provide an objective assessment of its accuracy and reliability. Experiments performed on real video sequences yield encouraging results, proving the validity of the foreground extraction approach.

Not last, the execution performance aspect is addressed by exploring the potential for parallel execution, supported by all modern multiprocessor architectures. This work details the available parallelization options provided by the method, with experimental results showing its real-time execution capabilities.

1. INTRODUCTION

1.1. The research theme

In today's digital world we are surrounded by a large number of applications that rely on image processing in order to accomplish a certain task. Image processing algorithms are used to analyze and give meaning to an observed scene by extracting core data and filtering out unneeded information for the application that employs them. For example, surveillance cameras detect motion in an observed scene; gesture recognition systems such as the Microsoft Kinect™ track body motion with the purpose of controlling interactive media content [1]; digital cameras rely on face recognition in order to better focus on the subject while medical imaging systems perform a 2D or 3D representation of the human body to diagnose illnesses.

In this respect, foreground extraction is a widely used technique and in the same time an ill-posed problem [2, 3]. What is considered as foreground (FG) for a certain application may as well be classified as background (BG) in a different scenario and there is no unique solution to the FG / BG segmentation problem. This is easy to illustrate if we consider a video sequence involving a crowded street with both pedestrian and vehicle traffic, as in Figure 1-1. For this same scene, a traffic monitoring system will label vehicles as FG and the rest as part of the background while a crowd monitoring system like the one described in [4] will consider the pedestrians as FG and everything else as BG. This outlines the importance of developing and implementing different flavors of FG / BG segmentation algorithms according to the specifics of the application that employs them.

(a) **(b)** **(c)**

Figure 1-1. Foreground segmentation as an ill-posed problem: (a) original image;
(b) segmentation by a traffic monitoring system;
(c) FG segmented by a pedestrian monitoring system.

In the recent years, videoconference systems have become ubiquitous grace to the large scale introduction of fixed and mobile broadband internet as well as the availability of affordable and easy to use smart communication devices. In an age where projects and businesses expand across continents and people are always on the move, videoconferencing has become the essential tool in supporting remote collaboration. After successfully achieving the goal of real-time audio-video communication and digital media sharing, the next logical step in videoconferencing

is towards delivering an *immersive experience* by providing participants with a shared virtual space that further enhances collaboration options and the feeling of natural interaction [5-7].

At the foundation of the immersive conferencing concept lies the ability of the videoconference system to extract, in real-time and with high accuracy, the foreground (FG) information from each participant's video stream. The extracted FG is then integrated into the virtual space so as to give the impression that all conference participants are present in the same location at the same time. We can immediately observe that in order to deliver a convincing immersive experience, a high *perceived quality* level is required from the FG segmentation. Taking into account the ill-posed aspect of FG segmentation in general, this becomes a challenging task with no unique solution.

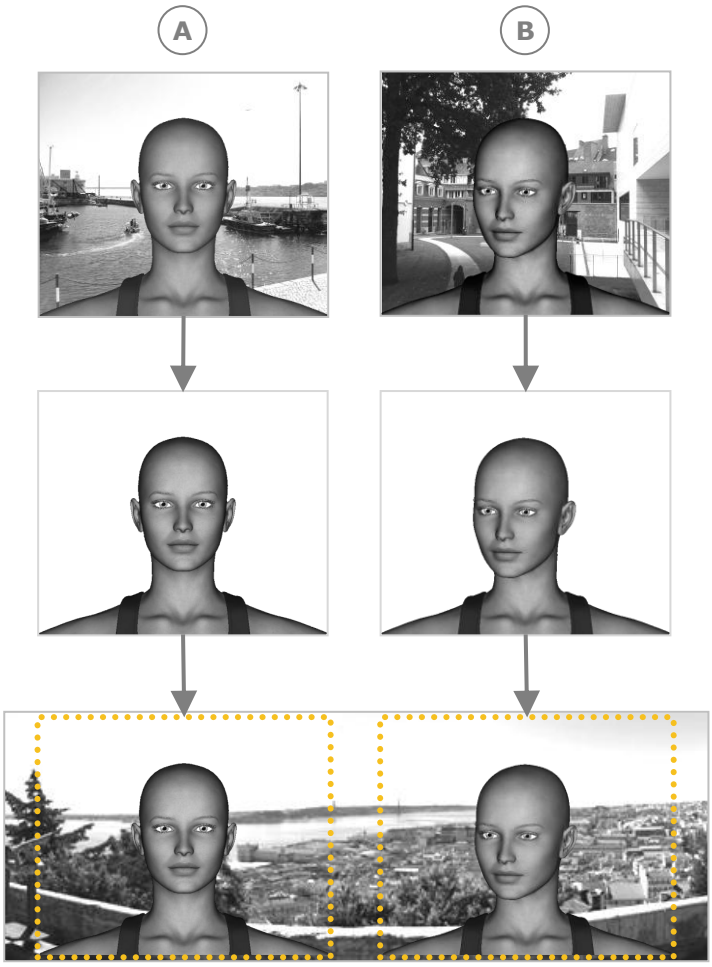


Figure 1-2. FG extraction and BG substitution in immersive videoconference systems

1.2. Emerging trends in video conferencing

State-of-art video conference systems that adhere to the *immersive experience* concept, operate according to the steps illustrated in Figure 1-2 and detailed below:

1. each participant's video stream is subject to a *FG extraction process*, with the purpose of isolating the person silhouette from the original background;
2. the resulting stream from step 1 is subject to an (optional) post-processing phase in order to ensure that it fits the virtual space in the best possible way, for example by adjusting the contrast and luminosity or even through advanced techniques such as image-based relighting [8];
3. the result from step 2 is integrated into the new background which simulates the virtual space, and the obtained scene is presented to all participants; the new background may be a static image or an (usually synthetic) video sequence.

We immediately notice that the quality of the *FG extraction* process is a key point for such applications. Standard video stream quality assessment methods are not sufficient in this case, as we need to also quantify the degree to which the FG / BG segmentation result appeals to the conference participants. [9] have proposed a set of perceptually-driven metrics based on psychophysical experiments which enable the measurement of FG segmentation quality from a human perspective in different scenarios. For immersive videoconferencing, which falls into the broader group called mixed reality, the authors found that the following artifacts have the most impact on the perceived FG segmentation quality (in descending order of their *annoyance* effect to a human observer):

1. *flickering*, which causes any of the artifacts described below and in Figure 1-3 to exhibit an erratic variance between consecutive video frames;
2. *inside holes*;
3. *border holes*;
4. *added regions*;
5. *added background*.

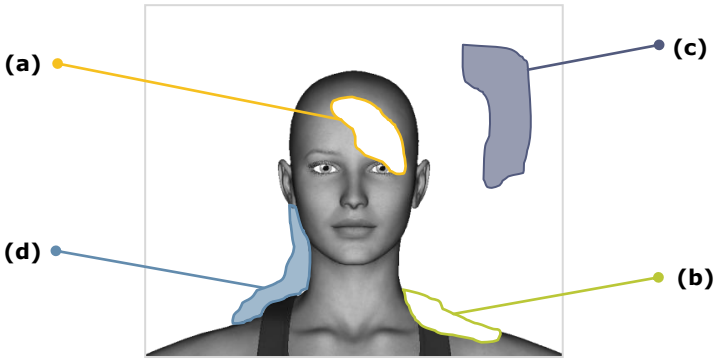


Figure 1-3. Artifacts in segmented FG: (a) inside hole; (b) border hole; (c) added region; (d) added BG

As a result, in order to rank higher in terms of quality, a FG segmentation method should put a higher priority on removing artifacts with higher annoyance factors.

In modern videoconferencing systems, defining what is considered as FG or BG can be a challenging issue. Let us consider the case of a person (the main subject) standing in front of the camera as illustrated in Figure 1-2, and of another person located at a certain distance behind the first one, also caught on the camera. The immediate question, arising from the ill-posed nature of FG segmentation, is *shall the second person be considered as part of the FG or as part of the BG* ? If the person is considered as FG, what happens if we have a constant or discontinuous flow of moving persons behind the main subject ? Otherwise, if the 2nd person is considered initially as BG, it may suddenly appear as FG if it moves close enough to the main actor. This example was given to emphasize the fact that having a *perfect* FG extraction for videoconference systems may be an impossible task since defining a *perfect* FG / BG segmentation may prove ambiguous even for a human intelligence.

1.3. Thesis objectives

In line with the research theme mentioned in the beginning of this chapter, this thesis focuses on addressing the problem of foreground extraction in monocular video sequences with direct applicability to immersive videoconferencing applications. The proposed main objectives of the thesis are as follows:

1) Address the ill-posed nature of foreground extraction by developing a *model-less binary segmentation method* that relies exclusively on motion, contrast and color information in order to identify and extract the FG objects in the observed scene. Compared with prior art, the new approach must produce similar output quality without relying on any *a priori* knowledge or training in respect to the scene or the objects being segmented and must approach real-time CPU execution performance. This objective aims to prove that motion-based foreground identification can be sufficient, feasible and aligned with the main traits of human perception.
2) In line with the first objective, *design and implement a motion segmentation algorithm* capable of performing an accurate identification of moving image regions in the presence of unfavorable conditions such as lighting changes, camera noise and video compression artifacts, which are often encountered in live video streams.
3) Study the techniques and methods that allow the *temporal integration of detected motion* into a coherent FG representation and develop an algorithm that generates an image of FG detected through motion.
4) Since motion alone may not completely expose non-rigid objects, study the techniques that enable the *pixel-accurate segmentation of foreground objects starting from coarse or incomplete representations*, as well as their applicability to the particular case of videoconferencing applications.

1.4. The proposed approach

Since no *a priori* information is available to the FG extraction system, the concept behind the proposed approach is that FG objects in the observed scene can be revealed by motion cues extracted between consecutive frames. Once an object (or part of it) is revealed by a motion cue, the corresponding image region is registered as FG in a temporally stable mask (TSM). The TSM will persist the region

as part of the FG detected through motion as long as its recorded properties do not exhibit significant variation from one frame to another.

Considering that detected motion might not reveal complete objects but parts of them, the TSM is very likely to contain an under-segmented FG representation. As shown in Figure 1-4, the final step in the proposed approach generates a pixel-accurate FG image by completing and correcting the TSM segmentation based on edge and color information.

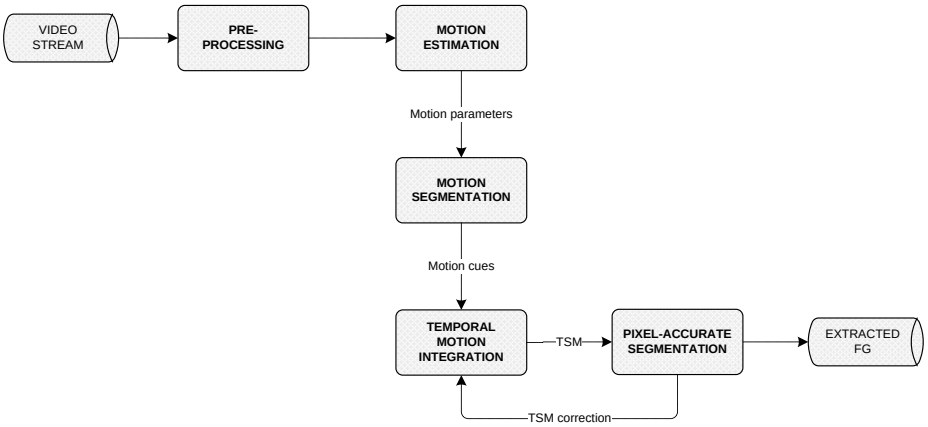


Figure 1-4. High-level block diagram of the proposed approach

1.5. Overview of the thesis

The thesis is structured into 7 chapters which group and consolidate the results obtained during our research activity:

- *Chapter 1* is dedicated to this introduction and outlines the research theme, our main objectives and a high-level description of the proposed approach to foreground extraction.
- *Chapter 2* represents a review of the current state-of-art in FG extraction, with an accent on motion segmentation and tracking in monocular video sequences. We also discuss the particular case of videoconferences and their specific requirements in respect to FG extraction.
- *Chapter 3* starts by presenting the concept of optical flow, the various methods for computing optical flow and a comparison between them. It then describes our novel approach to accurate motion segmentation based on the aggregation of dense and sparse OF estimations.
- In *Chapter 4* we introduce the concept of temporally stable masks as means for temporal integration of detected motion cues. We present the statistical model and the similarity measures used as indicators in the process of FG / BG labeling, the TSM algorithm, as well as the relevance of the obtained results in the context of videoconferencing applications.
- *Chapter 5* describes the general concepts behind the graph-cut algorithm and its applicability to image segmentation, and introduces a novel algorithm for the automatic generation of hard constraints as input for an

unsupervised graph-cut segmentation, based on the information provided by sparse OF estimation and the TSM.

- *Chapter 6* discusses the results obtained by applying the complete FG extraction method. We begin by reviewing implementation details and the execution performance of the proposed approach, followed by an assessment of segmentation quality using a state-of-art perceptual objective metric.
- *Chapter 7* draws the conclusion, summarizes our contributions to the field and outlines future research perspectives.

2. STATE OF THE ART

2.1. Overview

Foreground / background segmentation has been an active research area in the field of video sequence processing and the literature describes a wide range of algorithms and methods that address this problem [14, 31, 46, 47]. For a better insight into this broad topic, the present chapter is structured in 3 major parts.

The first part reviews the state-of-the-art in the domain of FG extraction in videoconferencing applications, with particular emphasis on methods that are capable of running in *real-time* and relying on *common hardware*, by which we mean *monocular* web cameras and embedded cameras found in laptops and mobile devices such as smart phones and tablets. While acknowledging the importance and the benefits brought by methods that employ dedicated hardware, we believe that ubiquitous access to immersive conferencing requires methods that leverage common setups, which are mostly capable of capturing and streaming monocular video sequences.

The second part narrows the scope to motion analysis in monocular video sequences, covering its four main aspects: detection, estimation, segmentation and tracking. Virtually all FG / BG segmentation methods, ours included, rely on a simpler or more complex form of motion analysis, therefore it is important to review and assess the available algorithms and methods in this field.

The last part in this chapter is dedicated to a discussion based on our findings in the literature. This helps crystallize our research perspectives and outline their importance in the context of monocular videoconferences.

2.2. Foreground extraction in videoconference systems

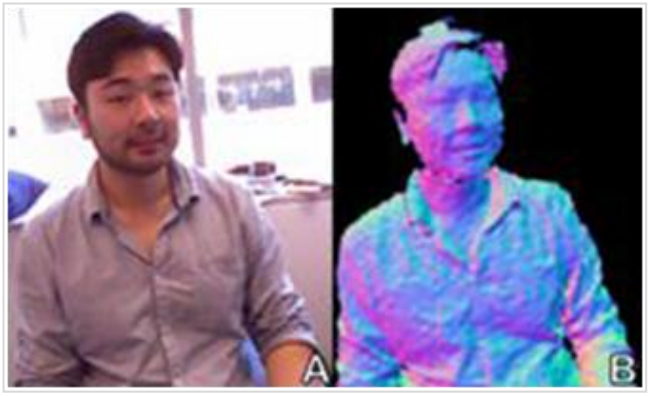
2.2.1. Stereo and multiple camera-based approaches

The best performers in the field are systems based on disparity maps obtained from stereoscopic [10] or multiple cameras [7]. These methods produce very accurate results by relying on additional depth information made available through the use of dedicated video capture hardware and specific algorithms.

[10] describe a method capable of performing real-time FG layer segmentation from stereo video, by fusing color, contrast and stereo correspondence information into a probabilistic model. The stereoscopic approach allows the authors to separate BG from FG motion and to focus on the objects that are closer to the camera, while a contrast-sensitive color model enables an accurate FG segmentation, generated by a Layered Graph Cut algorithm [11] applied on the fused probabilistic model.

[7] introduces the *Kinected Conference* concept, named after the use of the Microsoft Kinect™, a device capable of providing RGB + depth information by combining a regular camera sensor with an infrared laser projector and camera [12]. The depth map produced by the device and illustrated in Figure 2-1 enables

the system to highlight certain objects in the observed scene while blurring the rest, a concept with immediate applicability to videoconferencing applications. [13] take this concept even further, by using the Kinect sensor data to construct real-time 3D models of foreground objects.



(a) **(b)**

Figure 2-1. Kinect raw color image (a) and normals from the on-chip estimated depth map (b) (source: [13])

As mentioned in the beginning of this chapter, we have included this short review of stereoscopic and multiple-camera approaches for reasons of completeness. Although fast and accurate, such systems cannot support ubiquitous access to immersive conferences due to their requirement for dedicated hardware and setup. In order to make immersive conferencing reachable by the same audience as classical videoconferencing, the former needs to make use of common webcams. It is obvious that, in the context of monocular video sequence processing, a different class of algorithms is required in order to successfully handle the task of FG extraction and throughout this thesis we will focus exclusively on this scenario.

2.2.2. Foreground segmentation in monocular video sequences

2.2.2.1. Techniques based on background subtraction

In respect to monocular FG objects segmentation, the majority of algorithms rely on *background subtraction* [14]. This technique is based on an empty image of the observed scene provided during the initialization stage of the algorithm.

[15] handle the problem of human profile extraction using inexpensive desktop cameras by using a statistical BG model in the normalized RGB color space, adapted to compensate for camera's automatic exposure correction (AEC) function. In the FG extraction phase, BG is first subtracted from the current frame in normal RGB space and then a correction is applied on the result in the normalized RGB space in order to eliminate BG pixels erroneously classified as FG due to AEC or shadows. What makes the proposed method interesting is the use of a physics-based model to extract the person's contour. A drape (1-pixel thick line) is lowered

from the top of the FG bounding-box and wraps the FG region's contour, being modeled by two components, as seen in Figure 2-2:

- a mass (up thrust) component which ensures that homogenous FG regions have the power to hold the drape in place, and
- an elastic component that causes the drape to emulate on the contour.

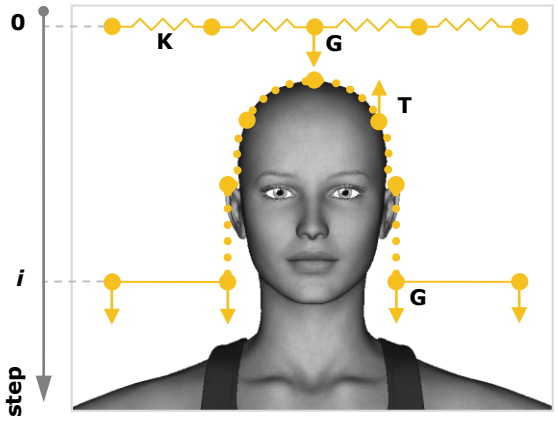


Figure 2-2. Using a drape model to extract the silhouette of a conference participant (K = constant of elasticity; G = gravitational pull; T = up thrust generated by FG regions)

The same approach is extended in [16] by using drapes directed from all four sides of the bounding-box followed by an *AND* operation between the results obtained from each drape in order to reveal the final silhouette of the FG object.

In [17] the FG layer is extracted by combining BG subtraction with color and contrast cues. The key concept revolves around background contrast attenuation, which reduces contrast in the BG layer while preserving it around FG/BG boundaries, based on the observation that color image gradients are dissimilar between the two layers.

The method proposed in [18] uses a known and stationary BG image and a frontal human body detector in order to perform an initial segmentation of the person in the scene. The result is subject to a coarse to fine segmentation process [19] which builds a GMM model of FG and BG pixels in order to provide the necessary input to an unsupervised graph cut (GC) segmentation [11]. In addition, a BG contrast removal process is used to attenuate the influence of cluttered backgrounds and a self-adaptive initialization level sets scheme is applied in order to find the most salient edges along the person's contour.

The major drawback of these otherwise accurate methods is their requirement for the initial clean BG image, which cannot be satisfied in most videoconference scenarios since people are usually in the scene starting from the first frame and the number of potential backgrounds is virtually infinite.

2.2.2.2. Methods based on trained models

Other methods have replaced the need for an initial background image with a learning model trained using manually labeled video sequences.

Criminisi et al. [20] have adapted stereoscopic approaches to monocular video sequences by fusing together motion, color and contrast cues obtained in the YUV color space with spatio-temporal (S-T) priors generated during training phase in a probabilistic framework capable of identifying the FG and BG layers in each frame. The segmentation itself is achieved by minimizing an energy functional which resembles the spatio-temporal HMM shown in Figure 2-3, decomposed as a sum of 4 terms:

- a *temporal prior*, responsible of enforcing the pixel labeling consistency across a 3 frame time window;
- a *spatial prior* which enforces the spatial consistency of labels between neighboring pixels;
- a *color likelihood* term which provides evidence for pixel labeling from their color distribution;
- a *motion likelihood* term that estimates pixel labels based on the assumption of stasis in the BG and motion in the FG.

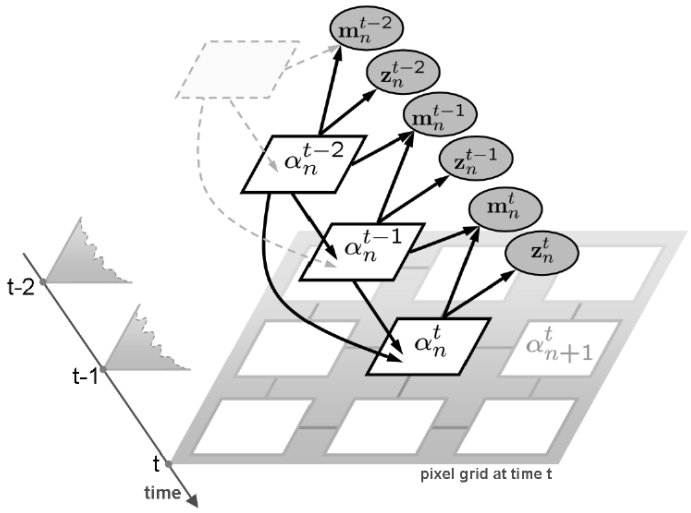


Figure 2-3. 2nd order HMM fusing S-T priors with color and motion observables in a 4-pixel neighborhood: m = motion observable; z = color observable; α = {FG, BG} label (source: [20])

In order to obtain a reliable priors and likelihoods model, extensive training must be done using hand-labeled ground truth data and the weights of the 4 energy terms must be calibrated for different video sequences [21]. The accuracy of this method (see Figure 2-4) is similar to the stereo one in [10], except for cases when FG color distribution resembles the one in the BG or when there is insufficient motion in the sequence [21]. It is worth noting that this approach sacrifices speed in favor of quality, as the proposed method is computationally intensive.

Figure 2-4. Results of the bilayer segmentation method proposed by Criminisi *et al.* (source: [20])

Further researches [22] have lead to additional improvements by replacing the HMM with tree-based classifiers trained on ground-truth segmentations that imitate the disparity maps used in stereoscopic vision. The tree classifiers operate on motion information encoded in the form of *motons*, which are motion descriptors similar to the textons used to describe texture information. Pixels are assigned to different motons by using a clustering algorithm operating on the S-T derivatives of the captured video frames. This method allows a better segmentation of the FG which is closest to the camera, being able to discard BG motion.

Despite their relatively high accuracy, both presented methods can be prohibitive due to the effort required to manually label the video sequences used in the learning phase.

2.2.2.3. Techniques that rely on imposed scene constraints

A third way of addressing the FG segmentation problem is encountered in literature in the form of constraints placed on the nature and position of FG objects.

[23] propose an object-oriented camera algorithm targeting the 2nd part of the MPEG-4 standard, related to object-based video compression and handling. The algorithm performs spatio-temporal (S-T) motion segmentation by combining a low-complexity segmentation technique with a marker extraction and update process, by following the steps below:

1. The segmentation technique starts with an image smoothing step in which a morphological open-close operation is used to remove noise and fine-grained textures from the video frame with the purpose of avoiding over-segmentation.
2. This step applies a morphological gradient to extract the edges from the smoothed image obtained in the previous step.
3. Temporal information is extracted by applying a scene cut algorithm on the current and previous frames in order to determine image regions that exhibit changes. The resulting change detection mask (CDM) is used to decide which of the two implemented marker extraction algorithms will be applied next.
4. Marker extraction is performed by taking in consideration the pixel count of the CDM. A marker is defined as a set of contiguous pixels from the edge image which exhibit a lower gradient value than a given threshold.
    a. If the pixel count of the CDM is higher than a threshold, the markers are (re)computed from the current frame and the CDM is discarded; the authors call this the *I-type* algorithm, in analogy with the I-type frames found in a MPEG-4 encoded video stream, which carry full scene information.

   b. If the CDM pixel count is lower than a threshold, the markers extracted from the previous frame are retained. The CDM is applied as a mask to the current frame and the masked pixels are set to 0 (not belonging to any marker) while unmasked pixels retain their respective marker associations. This algorithm is called *P-type*, since it takes in consideration the CDM, similar to P-type frames in MPEG-4 video streams.

5. The output of the marker extraction algorithm is subject to a watershed transform [24] which segments the image into regions. To avoid over-segmentation, a region merging algorithm based on a region adjacency graph and difference in adjacent regions average colors is applied as a post-processing step.
6. To extract the FG, the authors start from the assumption that the object of interest (the videoconference participant) is located in the center of the frame. This limitation allows them to overlay a predefined foreground object mask (FOM) on the region adjacency graph obtained in step 5. The regions that intersect with the FOM are classified as FG while the remaining ones are considered as part of the BG.

The proposed algorithm has a low complexity and is suitable for implementation on mobile devices, as exemplified by the authors using an ARM7 architecture. Further optimizations have been published in [25], which introduce block-level processing of the markers in the I-type and P-type algorithms in order to further reduce complexity and execution time while keeping similar quality results.

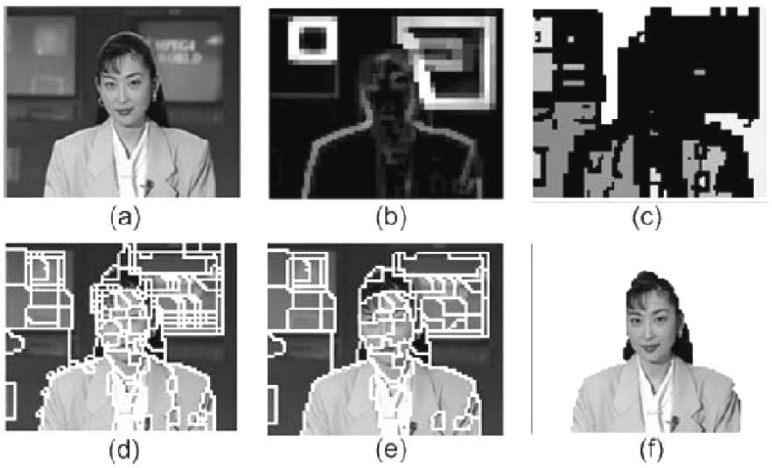


Figure 2-5. Object-oriented camera algorithm: (a) input frame; (b) edge image; (c) I-type markers; (d) watershed transform; (e) region merging; (f) FG image (source: [23])

Limitations of the proposed methods include the assumption that the FG object is located in the center of the image and the use of the FOM, which drastically reduce the number of objects that can be detected as FG and their position and relative size in the observed scene. In turn, this limits the number of scenarios in which such methods can be successfully applied. In case of videoconferencing applications, the segmentation needs to take into account extra movements, other than only those related to head and torso. A videoconference system needs to be

robust enough to handle cases in which, for example, a person uses hand gestures and body language in order to support the presentation of a topic or to show an exhibit to the other participants.

2.2.3. Summary

We conclude this section by summarizing the main advantages and disadvantages of the presented methods.

Table 2-1. Overview of state-of-the-art FG extraction methods

Method	Type	Characteristics	Advantages	Disadvantages
[10]	Stereo	Disparity maps Graph Cut	Very accurate	Requires stereoscopic camera
[7]	Multi-cam	Depth maps	Fast Very accurate	Requires Microsoft Kinect hardware
[15]	Monocular	BG subtraction Drape model	Accurate	Requires initial BG image
[16]	Monocular	BG subtraction with multiple thresholding Drape model	Very accurate	Requires initial BG image Threshold levels may require calibration
[17]	Monocular	BG subtraction Contrast attenuation	Accurate	Requires initial BG image
[18]	Monocular	BG subtraction Torso detection Contrast attenuation Graph Cut	Very accurate	Requires initial BG image Assumes subject is facing the camera and torso is fully visible
[20]	Monocular	Trained priors HMM Graph Cut	Accurate	Requires prior training Issues with similar BG/FG colors and insufficient motion
[22]	Monocular	Trained priors Tree classifiers Motons	Very accurate	Requires manual labeling of training sequences Requires prior training
[23] **[25]**	Monocular	S-T segmentation Specific algorithms for I and P frames	Fast Medium accuracy	Uses constraints related to subject position in the scene Only 1 person detected

By looking at the data presented in Table 2-1, it can be seen that all methods that do not make use of dedicated hardware rely on a certain form of training or *a priori* knowledge regarding the observed scene or its background. This observation has lead our research on the path towards a FG extraction system that eliminates the *a priori* learning / assumption aspect and focuses instead on building an accurate model of FG detected through motion. In order to expand this concept, we will first need to review the state-of-art in the field of motion analysis in monocular video.

2.3. Motion analysis in monocular video sequences

When discussing about motion analysis in video sequences, we usually refer to 4 main aspects: detection, estimation, segmentation and tracking [24, 26-28]. There are classes of algorithms applicable to each of the 4 domains; however, between detection, estimation and segmentation the boundary is not always clear, due to the presence of algorithms that belong to more than one of them. For example, the optical flow algorithms which will be discussed in the next chapter can be used to perform both motion detection and estimation in the same time.

Motion detection is responsible with identifying the areas in each video frame where motion occurs, usually by comparing the information between consecutive frames. In case of motion estimation, on top of detection we consider the ability to compute motion parameters such as velocity and orientation. Segmentation relies on the results obtained from the previous two steps, detection and estimation, in order to perform the accurate separation between moving objects and background in the observed scene. The fourth step, object tracking, is used to record and predict the segmented object(s) motion from the video sequence over a certain time interval.

2.3.1. Motion detection

2.3.1.1. Frame differencing

The easiest method but also the less accurate for detecting motion is the frame differencing method [24]. This method works on the assumption that between two consecutive frames I_t and I_{t-1}, motion occurred if a change in pixel data has been observed. Thus, we can say that the pixel located at coordinates $[x, y]$ is labeled as motion if

$$|I_t(x,y) - I_{t-1}(x,y)| > \tau \quad (2.1)$$

where τ is an application-specific threshold and $|.|$ means absolute difference. It is extremely easy to prove that this method is very sensitive to noise and to illumination changes in the image. To remove the effects of noise, the two frames can be pre-processed by using a Gaussian, median or bilateral smoothing filter; however, this will not solve the problem of illumination changes being incorrectly classified as motion.

2.3.1.2. Background subtraction

A more reliable and widely used set of motion detection algorithms are those based on BG subtraction or BG differencing. In essence, these algorithms work by building a mathematical model of the observed scene BG through a process called BG learning. In the ideal case, the BG model should fulfill the following requirements [14]:

- adapt to gradual or fast illumination changes (due for example to shadows, weather changes, clouds or flickering caused by artificial lighting);
- compensate for camera oscillation and motion;
- model high-frequency BG objects (e.g. trees swaying in the wind);

- model BG geometry changes (e.g. parked cars, objects that are introduced in the scene and remain motionless for a long period of time).

The mathematical model M_{BG} of the BG aggregates the evolution of each pixel over a certain time frame also known as observation period, during which the scene is assumed to be relatively static. The model is able to compensate for small motions in the BG, such as trees swaying in the wind or small illumination changes.

Once M_{BG} has been learned, it can be applied to subsequent frames through the BG differencing process which is similar to the frame differencing described in paragraph 2.3.1.1. The pixels in the newly captured image are compared with the learned BG model. If the new pixel data at location $[x, y]$ fits into the model $M_{BG}(x, y)$ within a certain threshold τ, the pixel is labeled as BG, otherwise it is promoted to FG. This can be described mathematically as:

$$Label(x,y) = \begin{cases} FG, & if\ |I(x,y) - M_{BG}(x,y)| > \tau \\ BG, & otherwise \end{cases} \quad (2.2)$$

There are many methods proposed in the literature for building the BG model, and several surveys [29-31] covering the progress in this area. The majority of methods rely on statistical modeling of each BG pixel using **Gaussian or Mixture of Gaussians (MoG)** distribution models [32-34] with an excellent state-of-art presented in [35]. However, as discussed next, there are also exceptions which yield some of the best results in the field.

The **Codebook model** associates a set of cylinders [36] or axis-aligned boxes [24] defined in the image color space (RGB, YUV or HSV) in order to account for the variation in BG values associated to each pixel. In the BG learning phase, shown in Figure 2-6, if a newly observed pixel value is close to a code element (box or cylinder, depending on the implementation), that element is expanded to incorporate this value. If the value falls too far from any of the recorded code elements, a new element is created to incorporate the new value. In the BG subtraction phase, a pixel value which does not fall close enough to any of the pixel's codebook elements will be labeled as FG, while one that falls within or close enough to a code element will be labeled as BG. The codebook model is updated periodically by cleaning stale code elements (elements which have not been accessed for a certain number of frames or a certain time interval).

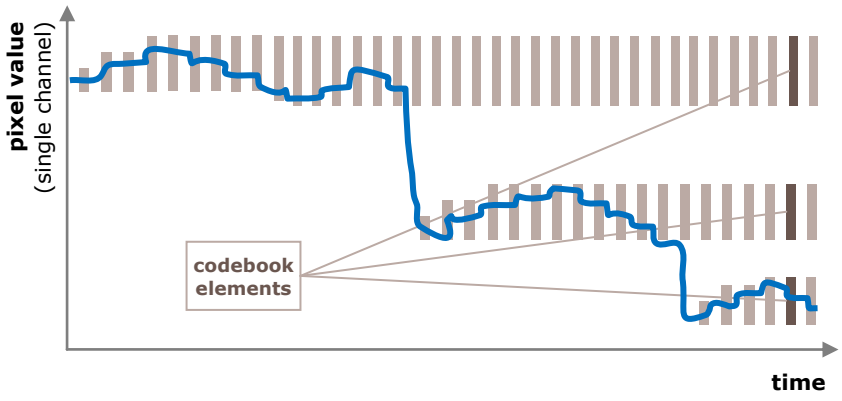


Figure 2-6. Creation of the codebook BG model (source: [24])

The **ViBE algorithm** [28] models each BG pixel using a collection of N sample values taken from previous frames. Instead of relying on a probability density function (PDF) or a GMM to model the distribution of the samples, the algorithm compares the pixel value in the current frame $I_t(x, y)$ with the set $M_{t-1}(x, y) = \{I_1(x, y), I_2(x, y), \ldots, I_N(x, y)\}$ of samples previously collected for that pixel. The comparison is done by considering a sphere $S_R(x, y)$ of radius R centered on the pixel and computing the cardinality $|.|$ of the intersection between this sphere and the sample set $M_{t-1}(x, y)$. If the inequality:

$$|S_R(x,y) \cap M_{t-1}(x,y)| > \tau \quad (2.3)$$

holds, the pixel is labeled as BG, otherwise it is considered as part of the FG. The ViBE algorithm also contains a model update component, so that the set of BG samples is updated according to a set of policies that ensure the consistency and relevance of the recorded model.

The vast majority of BG subtraction algorithms rely on a learning phase necessary to construct the BG model. This assumes that no moving objects that should be recognized as FG cross the scene during that interval; some algorithms even rely on a reference BG image. This constitutes no problem for applications such as video surveillance or traffic monitoring; however, it makes the algorithms harder to use for videoconferencing applications, where the FG (the conference participant) is present from the very first frame of the video sequence. ViBE is a notable exception to this rule, being able to learn the BG model on the fly, starting from the first frame of the sequence.

2.3.1.3. Optical flow

Both a detector and an estimator, the optical flow [2] is a popular algorithm in the field of motion analysis, with use in applications ranging from simple surveillance systems to complex 3D reconstruction of objects and human pose recognition.

As we will see in the next chapter, optical flow stands at the foundation of our proposed method for motion detection; due to its relatively high complexity and the fact that it comes in several different flavors, we chose not to discuss it here but instead to dedicate a significant portion of the following chapter to its presentation.

2.3.2. Motion estimation

2.3.2.1. Expectation Maximization

The Expectation Maximization (EM) algorithm [37, 38] is designed for parameter estimation in statistical models which depend on a set of unobserved latent variables, based on the maximum likelihood (ML) or maximum *a posteriori* (MAP) techniques. The algorithm works iteratively with each iteration being split in two steps:

1. the **Expectation (E)** step, in which the expected log-likelihood function of the latent variables is computed based on the current estimation of the parameters;
2. the **Maximization (M)** step, which computes the parameter values for which the log-likelihood function obtained in the E step is maximized.

Thus, if we consider X as a random vector representing the observed data; Z the random vector of unobserved latent variables; z the realization of Z and θ the set of parameters describing X, the n^{th} iteration of the EM algorithm can be formally written as:

$$\theta_n = \arg\max_{\theta}\{E_{Z|X,\theta_{n-1}}\{\ln P(X,z|\theta)\}\} \quad (2.4)$$

One of the most important aspects of the EM algorithm is that it guarantees convergence to a local minima since the log-likelihood function is increased with each iteration, as shown in [39].

The EM algorithm has been extensively used for motion estimation and motion segmentation purposes, especially for estimating the PDF of GMMs in case of motion segmentation applications [40]. In such cases, the vector Z is identified as being the set of K pixel labels, while the parameters θ can be modeled by the means μ_k and/or the standard deviations σ_k of each cluster of pixels in the image.

In other scenarios like [41], pixel clustering is achieved by considering cluster labels as being the vector Z and the direction and probability of each cluster as part of the parameter set θ. In [42] the set θ is represented by the velocity of the optical flow field and an intensity appearance model in an attempt to assign dense motion to predefined image layers such as FG or BG.

One of the drawbacks of the EM algorithm is the fact that the number of components required to estimate the PDF function has to be specified *a priori* and that the algorithm outcome depends on the initial values of the parameters [43]. Also, depending on the model being chosen, the algorithm can be computationally intensive and may not always be suitable for real-time processing of video sequences.

2.3.2.2. Optical flow

The optical flow [44, 45] acts as motion estimator by generating the optical flow field, which provides the displacement vectors along the image axes for each moving pixel. As in the case of motion detection, we will not detail the algorithm here since we will treat it separately in Chapter 3.

2.3.3. Motion segmentation

The literature describes a multitude of methods and algorithms targeting motion segmentation in video sequences. Many surveys have been carried on this topic [9, 27, 35, 46, 47], describing the difference in quality and performance between various approaches and methods.

So far, the problem of motion segmentation does not have a definitive solution, due to its ill-posed nature [2, 3]. Methods that produce excellent results in certain scenarios or application may fail to yield them in a substantially different context.

2.3.3.1. The main issues faced by motion segmentation

Motion segmentation has to deal with a certain number of issues which have a negative impact on the robustness of the method [35, 47]:

1. noise image due to a poor quality image source (camera and/or video stream encoding technique);
2. camera jitter;
3. camera automatic adjustments (e.g. automatic exposure, automatic focus);
4. local or global illumination changes due to the time of day, on/off switching of artificial lighting or shadows;
5. occlusions (caused by the overlap between moving FG objects or moving FG objects and BG elements);
6. foreground aperture [48], when parts of large, homogenous moving regions become part of the BG instead of being classified as moving FG pixels;
7. moved and inserted background objects;
8. multimodal (dynamic) BG;
9. waking and/or sleeping FG objects (objects that were initially motionless and then start moving and respectively objects that were previously moving and then come to rest).

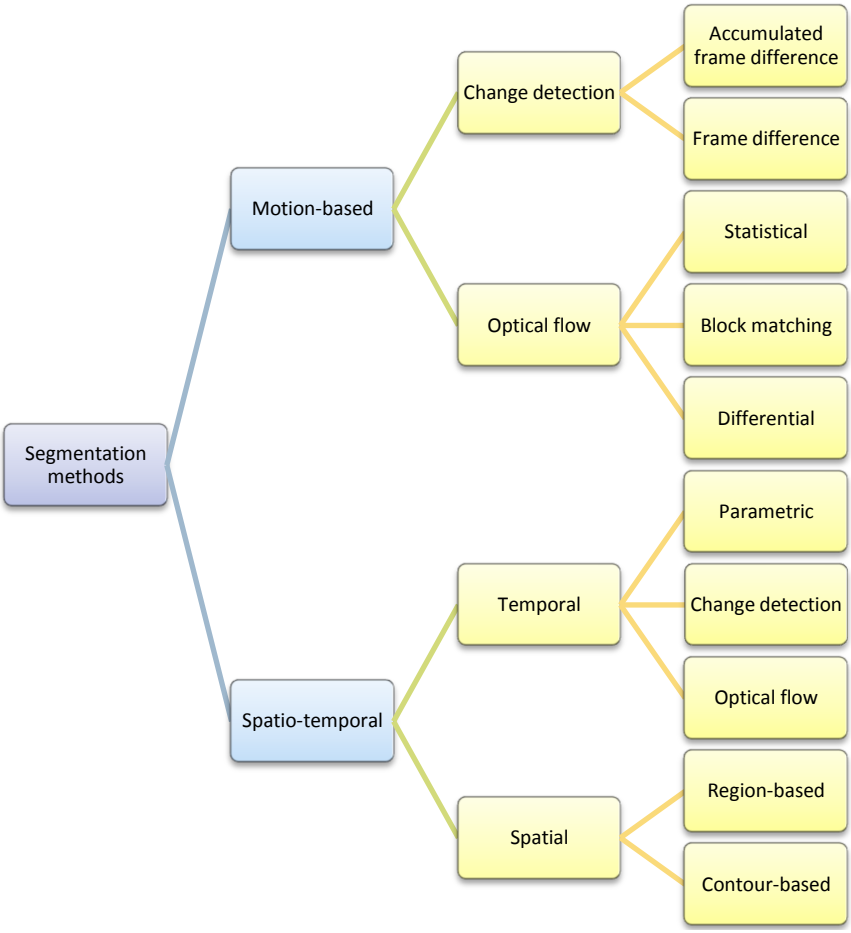


Figure 2-7. Classification of object motion segmentation methods (source: [46])

2.3.3.2. Classification of segmentation methods

In the literature, the classification of motion segmentation methods is usually not consistent. However, there are several main categories that can be found in the majority of classifications. By referring to the taxonomy in [46] we will consider the current 2D segmentation methods as belonging to two main categories: *motion-based* methods and *spatio-temporal* techniques, as illustrated in Figure 2-7.

2.3.3.2.1. Motion-based segmentation

Motion-based segmentation relies on an initial detection step, performed using one of the techniques already described in paragraph 2.3.1: *change detection* (carried on in the simple form of frame differencing or in the more reliable way of BG registration) or *optical flow*.

Segmentation itself is achieved through a clustering algorithm. This can take the form of connected component analysis [24], active contours [27] or statistical analysis [49]. The way motion is represented is a crucial aspect of motion-based segmentation, since it dictates the way clustering should be performed [46].

According to the literature, motion-based segmentation exhibits a tendency towards over or under-segmentation due to its inability to accurately estimate object boundaries, a problem handled much better by spatio-temporal techniques.

2.3.3.2.2. Spatio-temporal segmentation

As mentioned by its name, this type of segmentation relies on two components: a *spatial* and a *temporal* one.

The ***spatial component*** helps identify FG object boundaries and is calculated from a single image. This stage can be implemented using different approaches such as morphological operations coupled with a modified watershed algorithm [23, 25], the mean-shift algorithm described in [50], active contours [51] or graph cuts [52, 53].

Active contours (AC), also referred to as *snakes*, represent a robust approach in identifying object boundaries in an image, suitable for the spatial segmentation phase. Starting from a coarse set of control points located inside or outside of the object, the AC evolves iteratively towards matching the nearest object contour by minimizing its energy. In [54] a clustering algorithm is used to obtain the approximate convex hull of the object, represented by the set of N vertices $v = \{p_1, p_2, \ldots, p_N\}$. The AC energy of v is represented in the discrete domain as:

$$E(v) = \sum_{i=1}^{N} \left(\alpha_i E_{cont}(p_i) + \beta_i E_{curv}(p_i) + \gamma_i E_{img}(p_i) \right) \quad (2.5)$$

where E_{cont} represents the continuity of the snake, E_{curv} its smothness energy, E_{img} corresponds to the image forces acting on the snake (e.g. edge-attraction) and α, β, γ are their respective weights. The literature describes many ways of expressing the snake energies according to different usage scenarios, with a comprehensive survey being given in [55].

The ***temporal component*** is computed on a sequence of images and tracks the object movement across frames in order to ensure the temporal coherence and guidance of the segmentation process. The usual representation is that of an object mask that corrects and completes the FG object boundaries

generated by the spatial component; in turn, this component is updated based on the latest segmentation result [56]. More advanced representations employ several layers of masks generated according to estimated motion parameters such as velocity and orientation [42].

Statistical approaches are frequently used in this stage, due to segmentation being expressed as a classification problem. The expectation maximization (EM) algorithm, described in paragraph 2.3.2.1, is used by [57] in combination with a Markov Random Field (MRF) in order to represent and compute the object map. Other approaches rely on Bayes' theorem and Maximum A posteriori Probability (MAP) to continuously evaluate and update the pixel labels [58]. In case the observed events that lead to pixel classification are hard to model, Hidden Markov Models (HMMs) can be employed as classifiers [59], with the drawback that such a method requires a prior training phase [47].

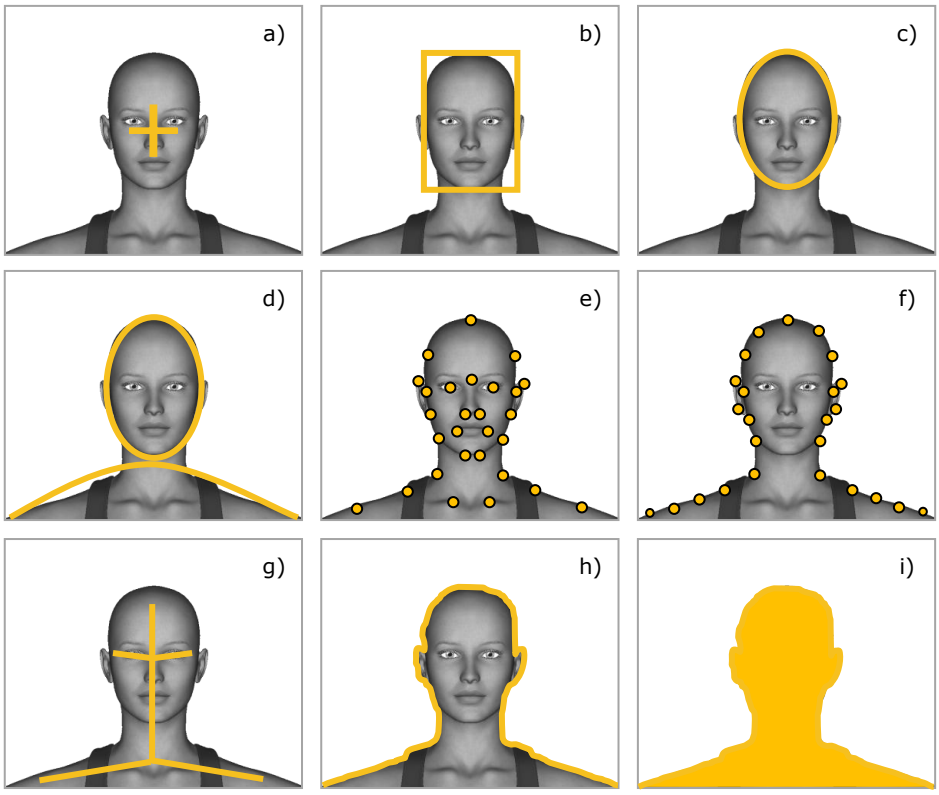


Figure 2-8. Object shape representations for motion tracking: a) centroid; b) bounding box; c) bounding ellipsoid; d) body parts approximated as ellipsoids; e) point features; f) contour points; g) skeleton; h) contour; i) silhouette (ghost mask)

2.3.4. Motion tracking

Once objects have been identified in the captured video stream, the last step involves tracking their representation across the video sequence. Depending on the application requirements, tracking can be expressed simply as following the

object(s) path from one frame to the other, or it can be more complex if other parameters are involved such as object shape or area [55].

2.3.4.1. Object representations

In the context of tracking, objects are represented by their shapes and appearances, and this representation is then matched against the set of observed object features [55].

Based on the requirements of the application, one can select an appropriate object ***shape representation*** from the set of possibilities illustrated in Figure 2-8 (profile tracking in a videoconference scenario) and explained in Table 2-2.

Table 2-2. Description of object shape representations for motion tracking [55]

Point(s) *Figure 2-8 a) and e)*	The object can be represented by a single point called centroid (or center of mass) or by a set of control points; this method is suitable when tracking very small objects or when the tracking problem can be reduced to following a set of control point trajectories.
Bounding primitive *Figure 2-8 b) and c)*	A bounding rectangle or ellipse can be employed to roughly approximate the image area occupied by the moving object. This kind of representation allows for the estimation of object size and for modeling motion through translations, affine or homography transformations.
Articulated shapes *Figure 2-8 d)*	Suitable for articulated (non-rigid) objects, this method represents each articulated part of the object as a primitive 2D or 3D shape (ellipsoid, parallelepiped, cylinder etc.). The object joints are modeled through kinematic models (e.g. joint angle).
Skeleton *Figure 2-8 g)*	This representation is achieved by applying a skeletonization process (medial line extraction) to the object silhouette. The resulting skeleton can model both rigid and articulated objects motion, and can also act as an object descriptor for object recognition applications.
Contour and silhouette *Figure 2-8 f), h) and i)*	The *contour* models the object boundary and can be represented either through a set of control points or as a continuous curve. In case of control points, their location is usually determined by corners or significant changes in contour orientation. In addition to the contour, the *silhouette* also covers the area of the object (the region located inside the contour).

The object ***appearance representation*** can be used either as a standalone input to the tracker or combined with a shape representation in order to complete the object model and increase tracking accuracy. Table 2-3 lists the most common appearance representations along with their descriptions, following the taxonomy from [55].

Table 2-3. Description of object appearance representations for motion tracking

Probability densities of object appearance	Coupled with a contour or silhouette shape representation, this statistical approach models the object appearance features (such as color, intensity, texture etc.) using probability densities. The probability density can be expressed parametrically (e.g. a $<\mu, \sigma^2>$ Gaussian distribution) or non-parametrically (e.g. the histogram representation used by the CamShift tracker [60]).

Templates	Suitable for objects whose appearance models do not exhibit significant variations, the template representation encodes the information about object shape and appearance obtained from a single object view.
Active appearance models	Active models represent object shape and appearance information extracted from an initial training phase. Feature points (or landmarks) are defined on the object and represented as a vector encoding color, texture or edge information. The resulting vector space is analyzed using Principal Component Analysis in order to obtain the object model.
Multiview appearance models	This approach encodes the shape and appearance information extracted from different object views; in order to represent this data, a subspace is generated from the given views over which algorithms such as Principal Component Analysis or Independent Component Analysis are applied.
Trained classifiers	Trained classifiers using Support Vector Machines (SVM) or AdaBoost algorithms [61] can be very efficient. The drawback is that in order to achieve accurate object identification during the tracking process, the object appearances in all possible views must be known *a priori* from the training phase.

The ***object features*** that are monitored during the tracking process are closely related to the selected form of object representation. Ideally, these features should uniquely distinguish the object from other elements in the image. There are 4 types of such features [55]:

- the *color space* [62] used to represent the color information in the image (e.g. Grayscale, RGB, YUV, HSV etc.) can have an impact on robustness to illumination changes and to the perceptual quality of the object representation;
- *edges* [63] are robust to illumination changes and - based on the assumption that object boundaries usually generate a strong change in the luminosity gradient of the image - these features are widely used, especially in conjunction with contour or silhouette shape representations;
- *optical flow*, represented as a dense field of pixel displacement vectors, relies on the brightness constancy assumption between consecutive frames [44, 45];
- *textures* describe the smoothness and regularity in the variation of color across the object surface; texture descriptors can be built in many ways (e.g. textons, wavelets, Gabor filters) [64], the main benefits being their robustness to illumination changes and the possibility to assess region similarities.

2.3.4.2. Object detection

In order to track the object, one must first detect it and this process usually occurs at the beginning of the video sequence. As expected, object detection is related to object representation since the result of the detection must be compatible with the chosen representation method. The literature describes a wide range of object detectors, most of which we have already covered in the previous paragraphs. We will refer below to the classification presented in [55] and include a brief description in Table 2-4.

Table 2-4. Classification of object detection methods

Point detectors	This class of detectors identifies a set of interest points in the image which are considered as representative for the local region they belong to, as shown in Figure 2-8 e). The most representative point detectors are the KLT detector [65] and the SIFT detector [66], which provide robustness against changes in illumination and camera viewpoint (in case of SIFT detector).
Segmentation	This class refers to the family of detectors described in chapter 2.3.3. The suitable object representations may vary according to the detector, e.g. region-based for mean-shift [50] and graph-cuts [52] or contour-based for the AC detectors [51].
BG subtraction	Described in detail in paragraph 2.3.1.2, these detectors are suitable for region-based representations of the objects.
Supervised classifiers	By learning different object views in an *a priori* training phase, these detectors are able to identify the occurrences of the learned object in the input image (a good example is the face detector described in [67]). As object representation, the one based on bounding primitives is the most appropriate.

2.3.4.3. Object tracking

Tracking establishes the correspondence of the detected objects between frames, as illustrated in Figure 2-9 for the case of a face tracking application.

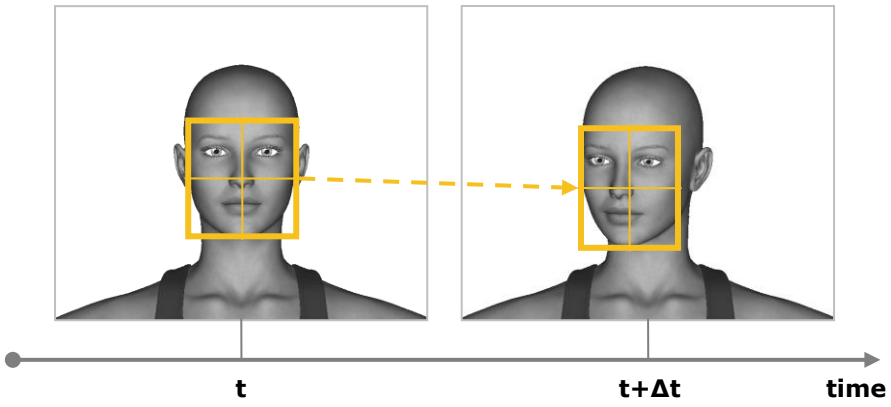


Figure 2-9. Face tracking using a kernel-based method (CamShift)

Depending on the object representation, different methods of tracking can be employed. For very small or rigid objects, represented using a point or bounding primitive, the correspondences are easier to represent since only translations and respectively affine transformations can be employed. For non-rigid objects represented by contours or silhouettes, tracking is a more complex issue and statistical methods (parametric or non-parametric) must be used to determine their contour or shape evolution across the video sequence.

Based on the above discussion, we can split object tracking methods into 3 categories [55]:

- *Point tracking.* As the name suggests, these trackers are used to track point representations. Representative for this field are the Kalman tracker and the particle filters (Condensation algorithm), very well explained in [24].
- *Kernel tracking.* These trackers employ templates with rectangular or elliptical shape and associated histograms. Tracking takes place by matching the kernel in consecutive frames and by computing its motion represented using translation, rotation and scaling parameters. The most frequently employed trackers from this category are the mean-shift [60], its adaptive version called CamShift [68] and the Kanade Lucas Tomasi (KLT) tracker [65].
- *Silhouette tracking.* In essence, trackers from this category perform an object segmentation in the temporal domain by extracting information from the object region in form of edge maps or appearance models. At a given point in time, tracking uses the motion priors extracted from previous frames in order to map the next location of the object in the frame. Unlike in the other 2 categories where consecrate solutions exist, silhouette trackers employ a wide range of methods, from using the Hausdorff distance to match object edges in subsequent frames [69] to HMMs [70] or minimization of contour energy functional.

2.4. Discussion

From the previous two sections we can easily notice that there is no perfect solution to the FG / BG segmentation problem even if we narrow the field to videoconference applications only. By referring specifically to section 2.2, we observe that recent advances have pushed the segmentation quality forward but this comes at the expense of required *a priori* training phases and higher computational complexity. Based on the state-of-art review presented in the present chapter, we can now identify the main research perspectives in the field of FG extraction in *monocular* videoconferences.

The most important direction is towards ***model-less FG extraction methods*** that do not rely on *a priori* learned BG models or motion statistics and, furthermore, are not tied to assumptions regarding the position or nature of subjects (i.e. person is located near the center of the frame, person's face is always visible and facing the camera). This aspect is important since during a videoconference we may often encounter situations when the FG is composed of more than a single human subject. For example, a participant may want to show an object to its interlocutors and demonstrate its functionality, another may hold a pet or a child in its lap and we may also see two or more persons sitting together in front of the same camera. The possibilities are almost infinite and training a classifier to cover all of them is simply impossible.

In the above-mentioned direction of model-less methods, motion cues play a very important role. Recording motion history across past video frames and ensuring its temporal coherence may be the only way to make sure that we account for all possible FG objects, regardless of their nature. We can thus consider the term of *temporally stable masks* (TSMs) that aggregate and track the motion observed since the beginning of the video sequence in both immediate and distant FG with emphasis on maximizing the perceptual quality of the segmentation (as described in paragraph 1.2).

The literature describes many ways of motion detection based on either dense motion information or sparse sets of invariant object features; these are usually obtained as the output of optical flow estimators. In most cases, dense motion data provides coarse and noisy outputs while sparse features, obtained with pixel accuracy, require a form of shape reconstruction which is an ill-posed problem in itself. We can thus identify a new direction in the ***aggregation of dense and sparse motion flows into temporally stable masks***, since this would allow an accurate estimation of moving objects without the need of shape reconstruction, by constraining dense motion flows with the set of sparse features. The resulting motion cues can then be temporally integrated by a TSM to form a coherent mask of the current FG detected through motion.

In the following two chapters we will focus on this newly-identified research perspective. We will outline the fundamentals of optical flow estimation, followed by a description of our proposed approach towards aggregating dense and sparse optical flows into temporally stable masks in the context of FG identification and persistence in videoconferencing systems.

3. A METHOD FOR MOTION SEGMENTATION BY AGGREGATING DENSE AND SPARSE OPTICAL FLOW INFORMATION

3.1. Introduction

Motion segmentation represents a core component in a wide range of applications. Surveillance systems use it to reveal moving objects in an observed scene, gesture recognition systems segment and track body motion with the purpose of controlling interactive media content [1] while immersive communication systems rely on motion segmentation to perform real-time foreground extraction and reunite participants in virtual meeting rooms [5]. The present chapter introduces our first contribution to the field of foreground extraction in monocular video sequences, in the form of a novel method for robust and accurate motion segmentation.

Based on the conclusions drawn in paragraph 2.4, and aligned with the proposed research objectives, the presented approach [71] brings together the accuracy of sparse optical flow (OF) estimation with the robustness of a dense OF method in order to accurately segment moving regions between consecutive video frames.

The chapter begins with an introduction to the concept of optical flow, accompanied by formal descriptions of the most important OF estimation methods found in the literature. Next, the reasoning and the formulations behind our original method for motion segmentation are detailed, followed by a summary of the most relevant experimental results and observations.

3.2. Optical flow

3.2.1. Definition. General considerations

Motion sensing and its understanding is an intrinsic ability of living forms endowed with the sense of vision. For human beings this is a natural process that we often take for granted regardless of whether we are moving in a scene or just observing it from a stationary position. In the world of computer vision things are more complicated and the (still unresolved) requirement is to obtain a general and flexible representation of motion that can be used in a variety of applications while being robust and computationally efficient [72].

Optical flow is the representation of the apparent motion of the observed scene projected onto the plane of a moving or stationary camera [73]. The concept was first introduced by the American psychologist James J. Gibson during World War II, as *the information carried by light resulting from environmental structure and the animal's path through the environment*. Optical flow (OF) can be described by a 2-dimensional velocity field which results from the observed motion of the objects in the scene and/or the motion of the observer, as illustrated in Figure 3-1. There are particular cases in which moving objects in the scene may not generate any

apparent motion (e.g. a perfectly smooth rotating sphere), therefore a distinction must be made between the concepts of motion flow and optical flow [74]. What we aim to determine is the motion flow, optical flow being only an approximation of the motion field; in most cases this approximation provides enough information to reconstruct the motion field, a process that occurs naturally in human brains.

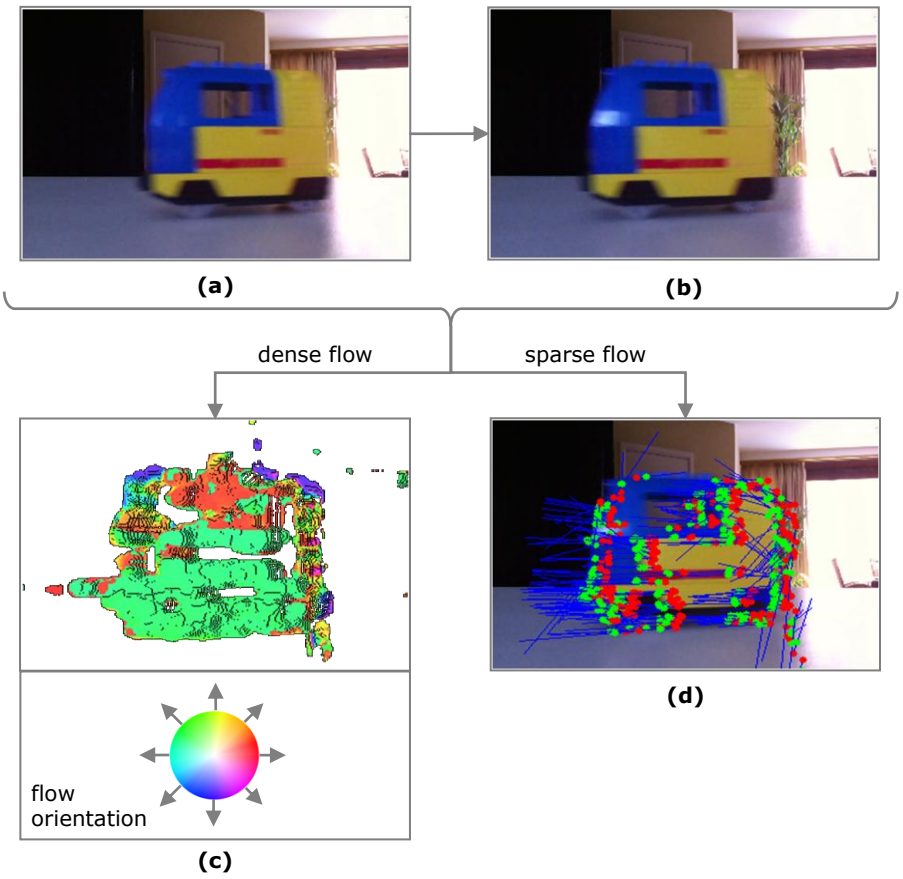


Figure 3-1. Optical flow field on the Train sequence: (a) frame at time t; (b) frame at time t+1; (c) dense optical flow and colors used to encode velocity orientation; (d) sparse optical flow field

Let us consider the image sequence $I(x, y, t)$, where (x, y) denotes the pixel location in the rectangular image domain Ω and t represents the time, $t \geq 0$. To compute the optical flow we start from the assumption that intensity values of pixels belonging to image objects do not change between two consecutive frames [2]:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{3.1}$$

an equation known in literature as the *brightness constancy assumption* [75].

Assuming small object movements between consecutive frames and that $dt \to 0$, a 1st-order Taylor expansion to the right-hand quantity will give us the equation

$$I(x+dx, y+dy, t+dt) = I(x,y,t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + e \quad (3.2)$$

where e denotes the high-order terms of the expansion, with $e \approx 0$.
From the equations (3.1) and (3.2), we obtain:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0 \quad (3.3)$$

and after dividing with dt we obtain the *optical flow constraint equation*:

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = I_x u + I_y v + I_t = 0 \quad (3.4)$$

where $u = \frac{dx}{dt}, v = \frac{dy}{dt}$ denote the optical flow field velocities along the x and y image coordinates.

3.2.1.1. The aperture problem

The optical flow constraint equation alone is not sufficient to compute the unknown velocities u and v. For non-vanishing image gradients this equation allows us to determine only the *normal flow* component, which is perpendicular on image edges [2]. This is known in the literature as *the aperture problem* and is illustrated graphically in Figure 3-2. In order to compute the remaining flow components another set of constraints is required, which also differentiates between various OF methods.

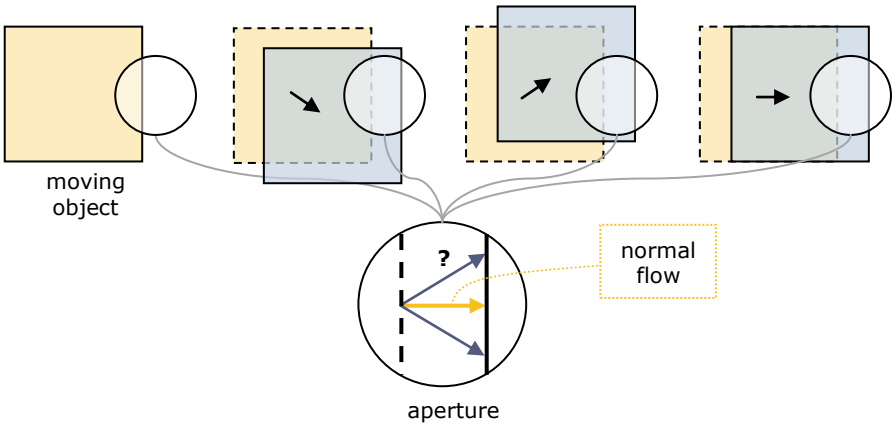


Figure 3-2. The aperture problem: when viewing exclusively through the aperture, object motion estimation is ambiguous.

As described in [73] the following types of constraints can be added to equation (3.4) in order to create a system that is solvable in its unknowns u and v:

- *Data conservation constraints*. These constraints exploit the fact that image measurements taken across small image regions yield similar values even if the location of the region changes in time. Alone, data conservation assumptions are not able to solve the aperture problem and in practice they must be combined with additional constraints like those described below.

- *Spatial coherence constraints*. Constraints from this category rely on the assumption that neighboring points should exhibit similar optic flow velocities as they most probably belong to the same object surface. In literature we encounter them under the name of *smoothness constraints*, as they assume optic flow to exhibit smooth variations across local neighborhoods.
- *Temporal coherence constraints*. In order to ensure the stability and persistency of the observed optic flow, temporal constraints assume that the motion of an object surface changes gradually over time. The effect of these constraints can be observed on two levels: first they reduce the noise in the optic flow estimation and second, they allow for faster computation by integrating previous results in subsequent estimations.

3.2.1.2. Method classification

Extensive surveys found in literature [2, 73, 75, 76] describe many ways to compute OF according to the set of additional constraints added on top of equation (3.4) and the way the resulting OF is represented. If we refer to the latter criterion, we can split OF estimation methods in two main categories, also illustrated in Figure 3-1:

- ***Dense optical flow***. Dense flow methods estimate the displacement vector $[u,\ v]^T$ for every pixel in the image and generate a continuous (dense) vector field covering the entire image.
- ***Sparse optical flow***. Sparse flow methods estimate the displacement vector only for specific points in the image, called *features*; the set of feature points is obtained by running a point detector algorithm, as described in paragraph 2.3.4.2. The result is a sparse vector field which indicates the velocity of each feature by matching its location between two consecutive input frames.

3.2.2. Dense flow estimation

Depending on the additional constraints imposed by the OF computation method, dense OF estimation in a succession of images can take place at a *local* or at a *global* level [2]. As the names suggest, local constraints work by minimizing an energy functional defined over a subset of the image (usually a pixel neighborhood) while global constraints deal with the energy minimization problem over the whole image domain Ω.

3.2.2.1. Local methods

3.2.2.1.1. Lucas-Kanade

At the foundation of local dense OF estimation methods stands the work of Lucas and Kanade [77] which describes a differential method based on the assumption that the optic flow is relatively constant in the neighborhood of each pixel. This *spatial coherence* assumption holds only if there are small and approximately constant displacements between two consecutive video frames [45].

Formally, the *local smoothness constraint* introduced by the Lucas-Kanade (LK) method translates into applying the optic flow constraint (3.4) to each pixel q_i, $i=1..n$ within the neighborhood N_p centered on a given pixel p. Giving each pixel q_i the same importance level may introduce significant errors in computing the image displacement $h = [u, v]^T$ for some cases [78]; hence, an additional weighting component is introduced in the form of a Gaussian kernel K_p matching the size of the neighborhood N_p, which assigns higher weights to pixels which are closer to the location p for which the displacement is being estimated. As a result, the problem of optic flow estimation can be formulated as a problem of minimizing the energy:

$$E_{LK}(u,v) \coloneqq K_P * [(I_x u + I_y v + I_t)^2] \tag{3.5}$$

within the neighborhood N_p [2].

The minimum energy occurs when $\frac{\partial E_{LK}}{\partial u} = 0$ and $\frac{\partial E_{LK}}{\partial v} = 0$; this results in a system of n equations in the unknowns u and v, which can be written in matrix form as:

$$K_p A h = K_p B \tag{3.6}$$

where:

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix},\ h = \begin{bmatrix} u \\ v \end{bmatrix} \text{ and } B = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

with $q_i \in N_p$, $i = 1..n$.

Equation (3.6) describes a system which is *over-determined*, since it contains more equations than unknowns. In most cases such systems are *inconsistent* since there is no solution to satisfy all equations simultaneously (although there is a multitude of solutions that satisfy a subset of them). It is possible, however, to find the approximate solution that minimizes the error in respect to each equation by applying the weighted version of the *least squares* principle. The problem is thus reduced to solving the system:

$$A^T K_p A h = A^T K_p B \tag{3.7}$$

By considering $G = A^T K_p A$ and $b = A^T K_p B$, we obtain the formula for the displacement:

$$h = (A^T K_p A)^{-1} A^T K_p B = G^{-1} b \tag{3.8}$$

with its detailed form:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i w_i I_x(q_i)^2 & \sum_i w_i I_x(q_i) I_y(q_i) \\ \sum_i w_i I_x(q_i) I_y(q_i) & \sum_i w_i I_y(q_i)^2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} -\sum_i w_i I_x(q_i) I_t(q_i) \\ -\sum_i w_i I_y(q_i) I_t(q_i) \end{bmatrix} \tag{3.9}$$

where $w_i = K_p(q_i)$ denotes the weighting factor applied to each pixel in the neighborhood N_p.

In order to validate the 1st-order Taylor expansion performed in (3.2), equation (3.8) holds only for small pixel displacements. In practice, computing an accurate estimation of h requires an *iterative* Newton-Raphson approach [45, 79]. At a given step k in the computation, the initial frame is moved by the previous estimate h^{k-1} and the residual optic flow vector η^k is computed between $I(x + u^{k-1}, y + v^{k-1}, t)$ and $I(x + u, y + v, t + dt)$. The complete pixel displacement at step k is obtained as:

$$h^k = h^{k-1} + \eta^k \tag{3.10}$$

and is known to converge after relatively few iterations.

Bruhn *et al.* [2] outline some of the limitations of the Lucas-Kanade method, namely:

- in case of smooth regions characterized by vanishing image gradients, the matrix G is not invertible and the optic flow cannot be computed;
- in regions where the smaller eigenvalue of the matrix G is close to 0 it is impossible to compute a reliable optic flow value due to the aperture problem.

The above-mentioned limitations break the density of the estimated optical flow; in order to restore it, other methods such as interpolation must be applied, which in turn lower the accuracy of the resulting estimation.

The local smoothness constraint holds only for small disparities, which are sometimes less than the pixel spacing in the frame. In such cases, the method is still usable if the current optic flow is extrapolated from disparities computed for previous frames or the method is applied on scaled-down versions of the image. In terms of noise robustness, the LK approach provides good noise resilience especially as the size of the $\mathrm{N_p}$ neighborhood increases [2].

3.2.2.1.2. Farnebäck

A different local constraint was introduced by Gunnar Farnebäck, who used polynomial expansion in order to approximate the neighborhood $\mathrm{N_p}$ of pixel p with a quadratic polynomial [80]. The polynomial expansion coefficients are obtained by applying a weighted least squares method thoroughly described in [81], which yields the following result:

$$I(x) \sim x^T A x + b^T x + c \tag{3.11}$$

where $x = [\mathrm{x}, \mathrm{y}]^{\mathrm{T}}$ represents the vector of image coordinates while A is a symmetric matrix, b a vector and c a scalar that correspond to the polynomial expansion coefficients.

If the polynomial described in (3.11) is subject to a *global displacement* with a value h, a new quadratic polynomial would be obtained:

$$I^{'}(x) = I(x-h) = x^T A' x + b^{'T} x + c' \tag{3.12}$$

having the new expansion coefficients:

$$\begin{cases} A^{'} = A \\ b^{'} = b - 2Ah \\ c^{'} = h^T A h - b^T h + c \end{cases} \tag{3.13}$$

Provided that the symmetric matrix A is invertible, the displacement can be immediately obtained from the 2nd equation in (3.13):

$$h = -\frac{1}{2} A^{-1}(b^{'} - b) = A^{-1}\Delta b \tag{3.14}$$

In reality, the displacement exhibits a spatial variation and so do the polynomial expansion coefficients, which can be expressed as functions of the image location x. This marks the transition from global polynomials to *local polynomial approximations*.

According to [80] the 1st equation in (3.13) does not hold on a local polynomial approximation so the term $\bar{A}(x) = \frac{A^{'}(x)+A(x)}{2}$ is introduced. Now, equation (3.14) can be rewritten in the form of the *optical flow constraint*:

$$\bar{A}(x)h(x) = \Delta b(x) \tag{3.15}$$

Similar to Lucas and Kanade, Farnebäck introduces a local smoothness constraint by making the assumption that the optic flow varies slowly within the local neighborhood $\mathrm{N_p}$. Considering that each pixel $\mathrm{q_i} \in \mathrm{N_p}$ is weighted by a function

$w(q_i)$, the problem of estimating the displacement $h = [u, v]^T$ is reduced to minimizing the energy:

$$E_{FB}(h) = \sum_i w(q_i) \, \|\bar{A}(q_i)h - \Delta b(q_i)\|^2 \tag{3.16}$$

which gives the value for the optic flow velocity:

$$h = \left(\sum_i w(q_i) \, \bar{A}(q_i)^T \bar{A}(q_i) \right)^{-1} \sum_i w(q_i) \, \bar{A}(q_i)^T \Delta b(q_i) \tag{3.17}$$

In order to solve the problem of estimating large displacements caused by fast and ample motions between frames, the Farnebäck (FB) approach includes two additional capabilities:

- support for *a priori knowledge*, which allows the use of an a priori determined displacement field (i.e. computed in a previous point in time) as an input for the computation of the current displacement; in turn, this capability leads to *iterative displacement estimation*;
- *multi-scale estimation*, which relies on a pyramidal decomposition of the image [24] to estimate displacement starting from coarser levels (where large motion can be estimated) and progressing towards the full image level (where small motion is observed) by refining the estimates at each step.

As a result of the approach described above, Farnebäck's method provides better accuracy that most 2-frame optical flow estimation methods at the expense of increased computational time but with a good potential for parallelization. As a weakness, the assumption of a slowly varying displacement field causes flow discontinuities to be smoothed out at object border regions [80].

3.2.2.2. Global methods

3.2.2.2.1. Horn-Schunck

As opposed to local methods, global dense OF estimators introduce smoothness constraints which apply to the whole image domain Ω. The reference work in this area is the one from Horn and Schunck [2, 44].

The Horn-Schunck (HS) method relies on the minimization of a global energy functional in order to enforce flow smoothness across the whole image:

$$E_{HS}(u, v) = \int_\Omega [(I_x u + I_y v + I_t)^2 + \alpha(|\nabla u|^2 + |\nabla v|^2)] dx dy \tag{3.18}$$

where $\alpha > 0$ is a regularization constraint which controls the flow smoothness (smoothness weight). Larger values of α will have a penalty effect on large flow gradients by increasing the value of the energy functional.

In order to minimize the functional (3.18) the Euler-Lagrange equations are used [2], giving the following system:

$$\begin{cases} \Delta u - \frac{1}{\alpha}(I_x^2 u + I_x I_y v + I_x I_t) = 0 \\ \Delta v - \frac{1}{\alpha}(I_x I_y u + I_y^2 v + I_y I_t) = 0 \end{cases} \tag{3.19}$$

where Δ represents the Laplacian operator applied in the spatial domain, $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$.

The Laplacian can be estimated by using finite differences. For example, by considering $\bar{u}$ as the weighted average of the horizontal flow u in a neighborhood N_p centered on the location p where the flow is computed, we can write $\Delta u = \bar{u} - u$. By substituting the Laplacian estimation in (3.19) we obtain a linear system, with immediate solutions for every pixel in the image. However, since the estimation relies on the flow coming from the neighborhood N_p the solution must be computed *iteratively* every time a neighbor is updated, resulting in the following set of equations applicable in each iteration j:

$$\begin{cases} u^{j+1} = \bar{u}^j - \dfrac{I_x^2 \bar{u}^j + I_x I_y \bar{v}^j + I_x I_t}{\alpha + I_x^2 + I_y^2} \\ v^{j+1} = \bar{v}^j - \dfrac{I_x I_y \bar{u}^j + I_y^2 \bar{v}^j + I_y I_t}{\alpha + I_x^2 + I_y^2} \end{cases} \tag{3.20}$$

According to [2], the HS method can be characterized by an important benefit and also by a significant limitation. The benefit is that image areas with vanishing gradients are *filled in* from the motion boundaries due to the presence of the flow regularization term in the energy functional. The limitation resides in the method's increased sensitivity to noise. Noise causes high gradients in the image which impact the data term ($I_x u + I_y v + I_t$) of the regularization functional (3.18). The smoothness term is weighted by the same constant α regardless of the presence or absence of high image gradients, therefore this term will have less relevance in case of noisy structures. An increase of the smoothness weight may compensate the undesired noise effect but at the same time in will over-smooth the sought-after fill in effect.

3.2.3. Sparse flow estimation

3.2.3.1. The Kanade-Lucas-Tomasi feature tracker

Based on the early work of Lucas and Kanade reviewed in paragraph 3.2.2.1.1, [82] have developed a complete method for detecting and tracking point features in image sequences. Their research shows that for an accurate displacement estimation, defining a good window N_p which is to be tracked across consecutive frames must be done in accordance to the tracking algorithm being employed and the assumptions being made in respect to the optic flow computation, since not all image regions contain motion information.

In the previous chapters we have seen that the spatial image gradient (I_x, I_y) plays a major role in the computation of the optic flow. By referring to equation (3.8):

$$h = G^{-1} b$$

Tomasi and Kanade consider that the window N_p can be correctly tracked between two frames only if the system represents a good measurement and can be solved reliably. This means that the coefficient matrix G is above the image noise level and is well conditioned, which translates into the following conditions imposed on its eigenvalues (λ_1, λ_2):

a) both λ_1 and λ_2 must be large, in order to represent reliable patterns such as corners or rich textures well above the image noise level;

b) λ_1 and λ_2 cannot differ by several orders of magnitude, in order to meet the conditioning requirement [83], meaning that slight variations in one of the coefficients of G caused by errors will not induce large variations in the estimation of the optic flow displacement h.

If the minimum eigenvalue of G is sufficiently large to fulfill condition a), condition b) will also be matched since the maximum eigenvalue cannot be arbitrarily large due to the fact that intensity variations in an image are limited by the maximum allowed pixel value. As a result, we can now write the condition for reliable feature selection [65, 82] as:

$$\min(\lambda_1, \lambda_2) > \lambda \tag{3.21}$$

where λ is a predefined threshold. According to [82], λ can be safely chosen as $(\lambda_{min} + \lambda_{max}) / 2$, where λ_{min} is determined from a smooth image region and λ_{max} from a corner or a rich textured region.

As shown in [84] conditions a) and b) are fulfilled by the Harris corner detector [85], which is able to identify edge corners that exhibit a strong invariance to illumination changes, noise, rotation and scale.

The tracking method based on the equation (3.8) from the original LK approach and the feature selection criterion (3.21) defined by Tomasi and Kanade, is known in literature as the Kanade-Lucas-Tomasi tracker (or simply KLT).

In order to verify the correctness of feature tracking based on the KLT method, [65] have proposed an extension which checks tracked features against an affine motion transformation applied between non-consecutive video frames. The pixel displacement is represented as an affine motion field,

$$h = Dx + d \tag{3.22}$$

where $D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix}$ represents a deformation matrix and d denotes the translation of the feature window N_p centered on the feature p located at coordinate x.

While between two consecutive frames it can be safely assumed that D = 0 and we are left with the original LK equations, between non-consecutive frames the influence of deformation cannot be ignored. If the dissimilarity between the affine transformed feature and its representation in the current image varies abruptly, this is an indication that the feature is no longer reliably tracked and that it must be dropped.

3.2.3.2. Pyramidal Lukas-Kanade algorithm

In paragraph 3.2.2.1.1 we have mentioned the fact that the LK approach works only in case of small displacements, sometimes even smaller than the distance between 2 adjacent pixels. The KLT approach described in the previous section suffers from the same drawback. In order to compensate for this aspect, pyramidal image decomposition can be used to estimate the displacement at different scale levels [79, 86].

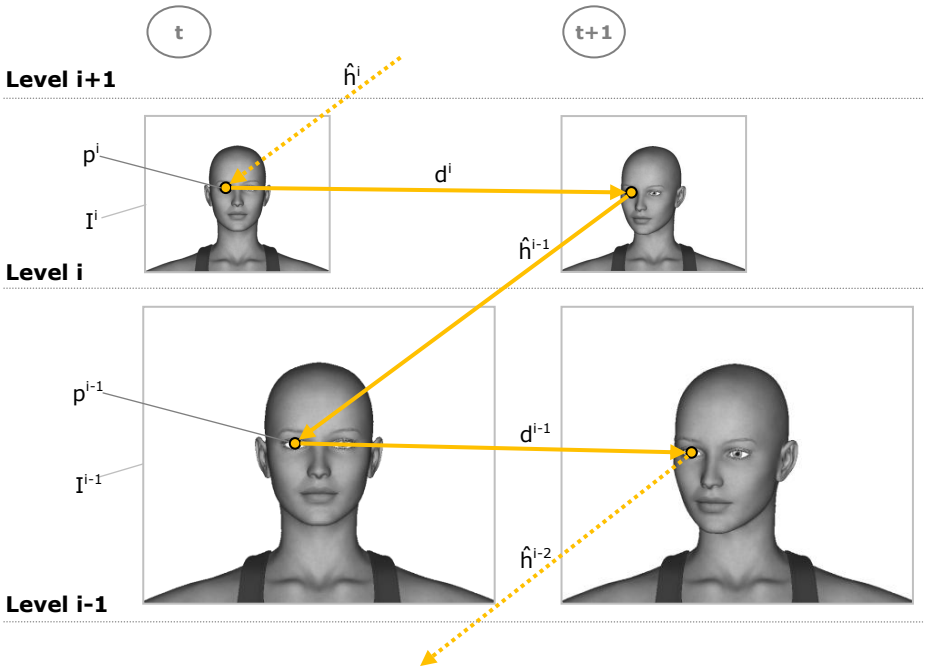


Figure 3-3. Pyramidal optical flow estimation

In a first step, the image is decomposed over a number of n levels. The image I^i corresponding to level i is obtained by scaling the image I^{i-1} (associated to level $i-1$) by a factor of 2. As a result, the coordinates of pixel p in the original image $I=I^0$ will be translated at level i by the formula $p^i = \frac{p}{2^i}$.

Next the optic flow estimation is computed, starting from the topmost level of the pyramid until the final level is reached. At a given level i in the pyramid the displacement vector h^i can be represented as:

$$h^i = \hat{h}^i + d^i \tag{3.23}$$

where:

- $\hat{h}^i$ is the initial estimation of the optic flow at level i, obtained based on all previous computations carried from level n to level $i+1$;
- d^i represents the residual displacement which minimizes the energy $E^i{}_{LK}(\hat{h}^i + d^i)$ described by equation (3.5) and applied to level i of the pyramid.

Since d^i is very small, it can be computed by using the regular iterative LK algoritm described in paragraph 3.2.2.1.1. Once h^i is computed, the estimation for the next level $i-1$ is obtained as:

$$\hat{h}^{i-1} = 2h^i \tag{3.24}$$

Considering that the initial estimation at the top of the pyramid is $\hat{h}^n = 0$, the final displacement will be given by the formula:

$$h = \sum_{i=0}^{n} 2^i \, d^i \tag{3.25}$$

We can immediately observe that the more levels are added to the pyramid, the larger is the maximum displacement that can be detected.

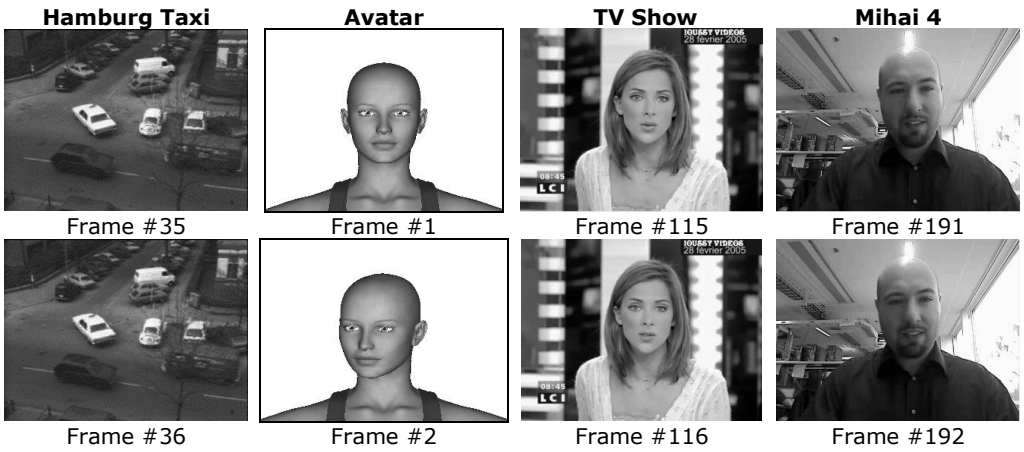
3.2.4. Discussion. Comparison between methods

30 years after the initial optical flow formulations, countless methods for estimating motion flow in moving image sequences have been devised, with more than 200 of them starting from the LK or HS methods [87]. Apart from the differences in problem formulation that have been outlined by several major surveys and articles [75-77, 88], almost all optical flow approaches have a common set of characteristics:

- *sensitivity to noise and illumination changes*, which generate erroneous displacement estimations;
- problems handling *occlusions and disocclusions*, where pixels disappear and respectively appear between consecutive frames due to overlapping between the projections of scene objects on the camera plane;
- issues handling *optic flow at object boundaries* due to smoothness constraints;
- difficulty in *handling large motions*, mainly caused by the 1st-order Taylor series approximation from equation (3.2);
- unreliable estimation in the presence of *motion blur*, which violates the brightness constancy assumption of equation (3.9) [89].

Of course, some of the proposed methods in literature handle particular aspects better than others or behave better in estimating specific type of motion (e.g. affine motion). However, due to the ill-posed nature of the optic flow problem there is no method which scores highest in all the above-mentioned categories. As shown in [87], pixels in a video sequence can be grouped in classes of algorithm suitability, based on the type of optical flow algorithm to which they respond best.

Figure 3-4 and Figure 3-5 show the results we have obtained by applying the LK, HS, Farnebäck and KLT optic flow algorithms to several test video sequences, both natural and synthetic.



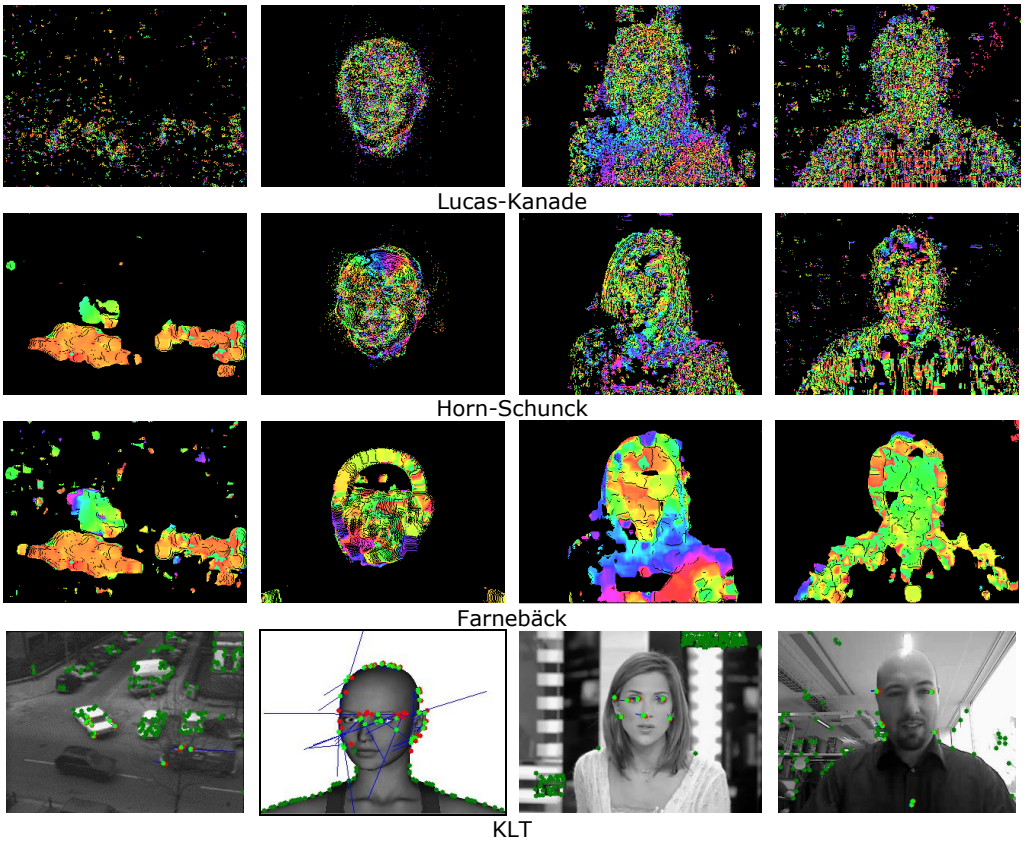


Figure 3-4. Optical flow estimation on different video sequences

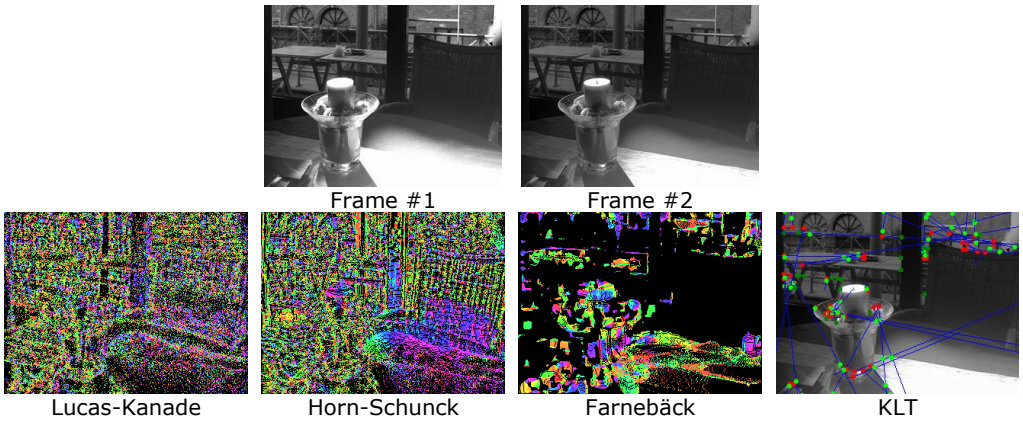


Figure 3-5. Impact of illumination changes on optical flow estimation on the Home sequence

Note that the dense optical flow estimations (LK, HS and Farnebäck) have been filtered so that only significant displacements are shown ($|h| > 0.5$ pixels). For the sparse flow estimation (KLT) all successfully tracked features have been displayed;

for features with a significant displacement, the previous position is shown in red and the current one in light green, with the displacement vector painted as a blue line, scaled by a factor of 5 for visibility purposes.

Based on the results illustrated above and additional experiments performed on a database of more than 40 video sequences, we can make the following remarks:

a) The LK method is the most sensitive to noise and video compression, as clearly seen in case of the *Hamburg Taxi* and *Mihai 4* sequences. However, it is the one to deal best with smooth regions, as observed in the dark shirt area of sequence *Mihai 4*.
b) Due to its global smoothing approach, the HS method deals reasonably well with noise and false positives due to video compression are also reduced. However, like its LK counterpart, this method is sensitive to illumination changes [90; 91], as illustrated in Figure 3-5 for the *Home* sequence.
c) The FB method is the most robust dense OF method in terms of noise and illumination changes, but it clearly exhibits an oversmoothing effect due to the quadratic polynomial approximation, as seen in all 4 sequences along object boundaries. This method produces false negatives in smooth image areas, as it can be seen in the *Avatar*, *TV Show* and *Mihai 4* sequences.
d) The KLT tracker provides pixel-level accuracy and robustness to noise, while still being sensitive to illumination changes due to the use of LK approach for each pyramid level. It can be seen that the number of successfully tracked features is higher in case of synthetic videos (i.e. the *Avatar* sequence), which are free of noise and compression artifacts. This suggests that applying the KLT method on higher intensity spatial gradients (edges) obtained from the input image may provide better results with more features suitable for object contour tracking.

3.2.4.1. Optical flow estimation in the context of videoconferencing applications

In the context of monocular videoconferencing applications, optical flow estimation is confronted with the following situations:

a. ***Illumination changes***. Local or global illumination changes between consecutive frames may be caused by a variety of factors, including reflections, changes in natural or artificial lighting, automatic camera exposure etc. Basically, illumination changes violate the brightness constancy assumption and introduce significant false positives into the OF estimation, as shown in Figure 3-5.
b. ***Noise***. Common video capture hardware in particular can introduce a lot of white noise in the captured video stream, especially in low-lighting conditions. Depending on the sensitivity of the estimation method, noise adversely affects the resulting optic flow vector field, as evidentiated in Figure 3-4 - *Hamburg Taxi* sequence for the LK method.

c. ***Compression artifacts***. Videoconferencing implies transmitting the video feeds collected from participants over a communication medium implemented using wired or wireless networks (or a mixture of the two). To make this process efficient in terms of required bandwidth and transmission speeds, compression is applied to the video component of the stream by the means of a video encoding algorithm such as H.263, H.264 or VP8 (for the astute reader, we mention that compression is applied independently to the audio component as well using audio-specific algorithms such as Nelly-Moser or Speex, but their scope goes well beyond the one of our paper). Depending of the choice of video encoding algorithm compression and the quality of the encoding, artifacts may be introduced in the video stream in the forms of blocking or oversmoothed regions [92]. This encoding noise can again introduce false positives or false negatives (e.g. for oversmoothing) in the OF estimation.

d. ***Occlusions and disocclusions***. Many types of (dis)occlusions can occur in videoconference systems, like objects and people moving in the background behind the conference participant, hand gestures performed by the main subject which occlude parts of his own body or large motions performed by the participant which expose previously unseen parts of the background. It is a known fact that dense OF in particular is not capable of occlusion handling, as explained in [93] and [94].

e. ***Smooth foreground regions***. In previous chapters and in Figure 3-4 – *Mihai 4* sequence we have shown that OF methods are unable to estimate the motion of smooth image regions due to their lack of details and the aperture problem. In videoconferences people often wear uniformly-colored clothing and in case of smooth and non-reflective surfaces OF estimation is a close to impossible task.

f. ***Camera instability***. OF estimation is highly sensitive to camera motion. A shaking camera or a camera mounted on a vibrating assembly will produce a frame-to-frame jitter effect, which manifests as a high frequency motion of the whole scene in the recorded video. Optic flow estimation methods are unable to distinguish the jitter from real motion and this leads to an over-estimated motion flow with a high rate of false positive responses visible especially in the background regions of the scene.

g. ***Motion blur***. In case of videoconference applications this effect is usually caused by hand gestures performed by the participants as part of non-verbal communication or in order to interact with the conferencing system [95]. This has a negative impact on the reliability of the OF estimation, due to violations on the brightness constancy assumption [89]. Sparse OF estimation is particularly affected since blurred image areas contain mostly smooth gradients and low textures that prevent the identification of good features to track.

In conclusion, using a single OF estimation method may not yield the expected results in a videoconference application. A dense OF method such as Farnebäck's may provide good results for situations a), b) and c) but due to oversmoothing it will not accurately identify moving object boundaries. The sparse KLT method can provide precise control points along moving object boundaries with subpixel accuracy and may be used to partially solve problem d) and even problem f) [96] but it will be sensitive to illumination and produce very noisy estimations when faced with situation a) or too few estimations for case g).

These observations effectively support our approach of aggregating dense and sparse OF estimations in order to compensate the drawbacks of each component method and to benefit from their strong points. As we describe in the next paragraphs, while this proves to be a solution for cases a)- c) and partially d), case e) requires a totally different, non OF-based approach and case f) requires a dedicated image processing algorithm. Case g) is solvable using some of the latest dense OF techniques [89], but the solution is too computationally expensive for the particular case of videoconferencing applications.

3.3. Aggregation of dense and sparse optical flow estimations

In the first part of the present chapter we have outlined the different approaches to OF estimation, their benefits and drawbacks as well as our reason for attempting to fuse the information provided independently by dense and sparse OF methods.

Since we aim at obtaining accurate boundaries of moving objects in the videoconference scene, it is obvious that dense OF provides insufficient information. Dense OF exhibits an over-smoothing effect which crosses the object boundaries, as seen in Figure 3-4. In the same figure we notice that sparse OF estimated with the KLT method produces pixel-accurate feature tracking at the expense of high noise sensitivity.

In the presence of stronger image gradients, the number of features successfully tracked by the KLT method is higher since more regions satisfy the inequality (3.21), as explained in chapter 3.2.3.1 and illustrated in Figure 3-4 – *Avatar* sequence. This observation enables us to use the sparse optic flow to track features located on the outer and inner edges of moving objects as a ***first step*** in our algorithm, by applying the KLT method on the edge images of consecutive video frames (see Figure 3-6).

The features obtained from the KLT tracker represent a set of control points, some precisely located on object boundaries and others found inside the object. For an accurate silhouette representation we need to identify the features located on the boundary. Since each object can have an arbitrary concave shape, extracting its hull from the set of control points is an ill-posed problem [97]. The ***second step*** of our algorithm shows that by converging the over-smoothed region obtained from dense OF estimation to the nearest control points produced in the first step, it is possible to accurately determine the concave hull of the moving object.

The concave hull obtained in the second step is modeled as a discrete set of control points. As a consequence, its boundary is composed of line segments that connect the control points and lacks the edge smoothness that is characteristic to the real object. In the ***third step*** of the algorithm we address this problem by using an active contour that maps on the real object boundary starting from the control points which define the concave hull.

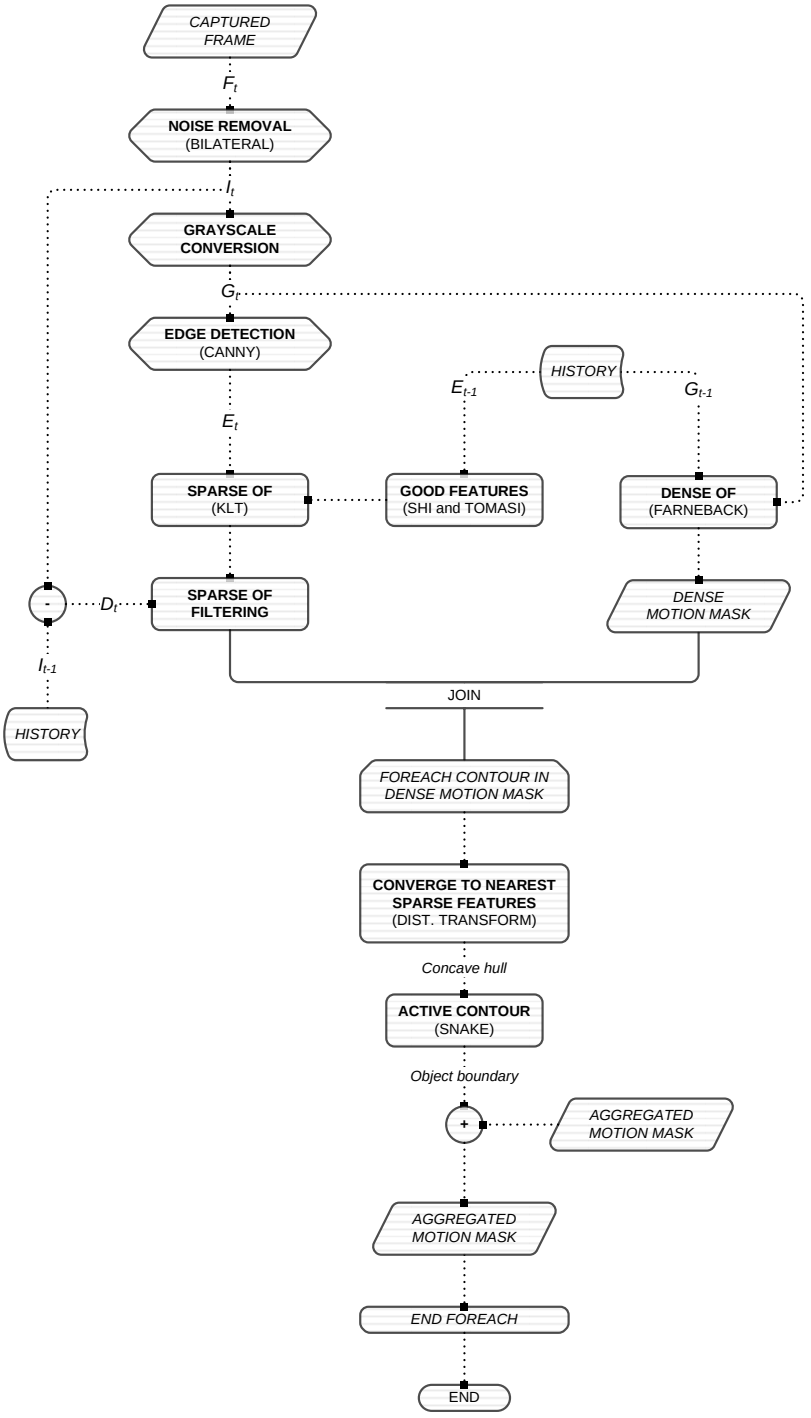


Figure 3-6. Block diagram of the dense-sparse flow aggregation algorithm

3.3.1. Pre-processing stage

The pre-processing stage prepares the captured input frame F_t for applying the algorithm steps and is composed of 3 stages:

a) ***Noise removal***. Noise is present in most (if not all) captured video streams and has negative impact on the accuracy of motion flow estimation. The literature abounds in noise filtering methods, the *median*, *Gaussian*, *bilateral* and *mean-shift* being some of the most popular ones [24, 98]. Noise filters exhibit a gradient smoothing effect which in turn decreases the effectiveness of edge detection algorithms. As we are interested in maintaining significant edges in the image for the sparse OF estimation, we employ the *bilateral* filter due to its dual image smoothing and edge-preservation characteristics [99]. We denote with $I_t = Bilateral(F_t, \sigma_r, \sigma_d)$, the image obtained after applying the bilateral filter on the captured input frame. We have determined experimentally that the best results are obtained for $\sigma_r = 50$ color levels and $\sigma_d = 3$ pixels, which is in accordance with the reports in the literature [99].

b) ***Grayscale conversion***. OF estimation algorithms as well as the edge detectors take as input single-channel grayscale images. On the other hand, F_t and I_t are color images that represent pixel data using 3-channel RGB or YUV color spaces. In this stage the grayscale image G_t is obtained by applying the following known transformations to each pixel p of the image I_t:

$$G_t(p) = \begin{cases} 0.30 \cdot I_t^R(p) + 0.59 \cdot I_t^G(p) + 0.11 \cdot I_t^B(p) & if\ color\ is\ RGB \\ I_t^Y(p) & if\ color\ is\ YUV \end{cases}.$$

c) ***Edge detection***. Accurate edge maps are required for the sparse OF estimation and tracking while maintaining real-time computational capabilities. Based on our previous research in the field [63], the best method satisfying both conditions is the *Canny* edge detector. Although other methods exist that achieve an even higher accuracy [100], they are simply too computationally intensive to apply to real-time application scenarios. Therefore, we consider the binary edge image $E_t = Canny(G_t)$.

3.3.2. Step 1. Generating control points from sparse flow and edge images

The purpose of this step is to identify a subset of point features φ, which belongs exclusively to moving objects in the scene, from the set of sparse OF features Φ tracked between two consecutive images.

Considering I_{t-1} and I_t to be the current and previous frames in the input video stream and $M_t \subset I_t$ as being the ground truth for motion segmentation between the two frames, we aim to fulfill the following criteria:

$$\varphi \subset M_t \wedge \varphi \cap \rho^-(M_t) \neq \emptyset \tag{3.26}$$

where $\rho^-(\cdot)$ represents the internal morphological gradient [101]. In other words, we want to ensure that we have features in the subset φ that are located on the boundary of true moving regions and that subset φ itself is included in these regions.

Considering that set Φ is obtained by applying the pyramidal LK algorithm described in [79] and by taking in consideration the results and discussion from

chapter 3.2.4, a way to ensure that we obtain the set φ which satisfies rule (3.26) is to identify the good features from the previous edge image E(t-1) and to track their new positions in the current edge image E(t). The reasons for this approach are that:

a) the contour $\rho^{-}(M_t)$ must match the boundary edges of moving objects (since M_t is the ground truth of motion segmentation) and in order to fulfill criteria (3.26) we need to have sparse OF features located on those respective edges;
b) the number of good features that can be identified in a binary edge image is higher than the one obtainable from the originating grayscale image since most edge pixels that exhibit a change in edge direction will satisfy inequality (3.21) and qualify as good features to track;
c) by using binary edge images, successfully tracked features will always be located on an edge, according to the brightness constancy assumption; tracking a set of features located on the edge of a moving object will return their new positions on the updated edge location;
d) edge images are less sensitive to noise and lighting changes so we can avoid the false positive responses shown in Figure 3-5.

To keep the obtained results accurate, for set Φ we establish an upper limit $|h|_{max} = 2^{L+1} - 1$ for the feature displacement magnitude, where L denotes the number of levels in the pyramidal decomposition. This limit is set according to the algorithm limitations described in [79]. For subset φ we introduce an additional lower limit $|h|_{min} = 1$ for the magnitude, in order to account only for the features that exhibit significant motion.

In practice we do not possess the information provided by the ground truth M_t and it is possible for subset φ to contain outliers that belong to $I_t - M_t$. We cannot completely remove these outliers in the absence of ground truth information. However, we can rely on the assumption that the new position of a moving feature must be accompanied by a significant change in pixel intensity between the previous and the current frames. This corresponds to the absolute frame differencing concept described in paragraph 2.3.1.1 and is expressed formally as:

$$\left|K_p * G_t(p) - K_p * G_{t-1}(p)\right| \geq \tau_D \tag{3.27}$$

where K_p represents a Gaussian smoothing kernel centered on the reference pixel p (required in order to avoid noisy outputs) and τ_D is a global threshold. In our experiments we have used a Gaussian kernel of size 5 x 5 and $\tau_D = 5$.

Figure 3-7 shows the results obtained after this step for some of the video sequences introduced in chapter 3.2.4. Sparse OF features are represented as small circles on top of the edge image E_t. Dark green circles represent stationary OF features with displacements below the $|h|_{min}$ threshold. Moving OF features have been represented using a light green circle for their current position, a red circle for their previous position and a blue vector to mark the displacement. Therefore, the union of green and light green circles represent the features from set Φ, while only the light green circles correspond to the features from subset φ.

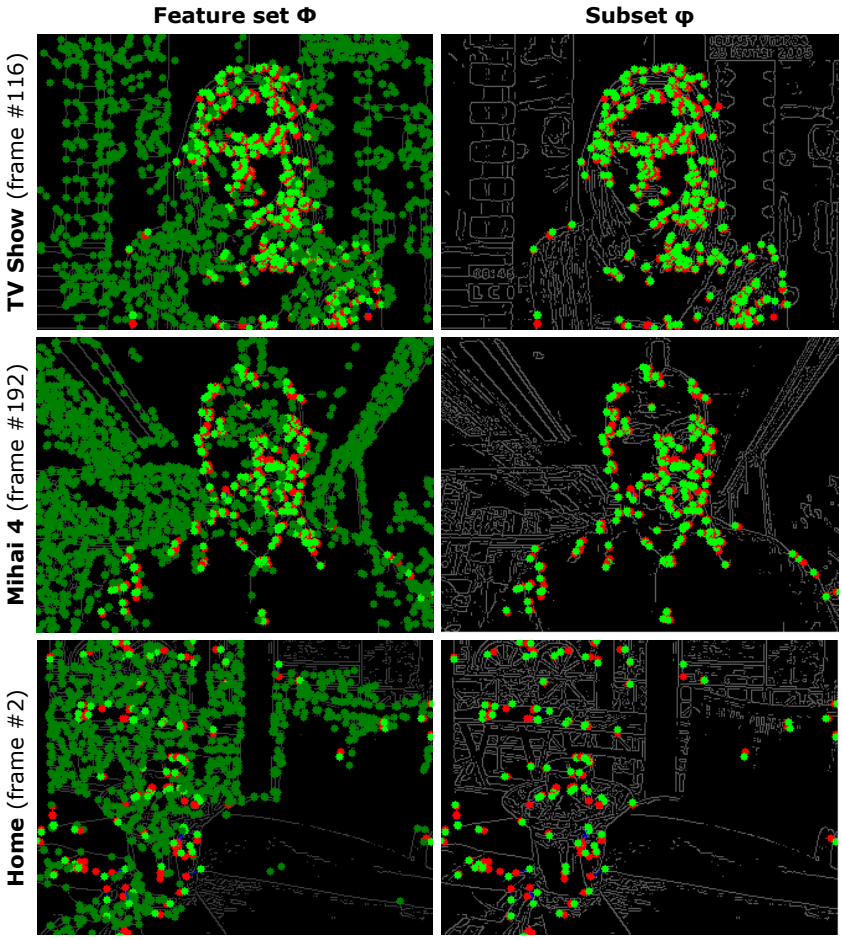


Figure 3-7. Control points on moving objects generated using sparse OF techniques

3.3.3. Step 2. Determining the concave hull of moving objects using dense flow

By examining the results shown in Figure 3-7, it is obvious that the boundaries of moving objects in the scene cannot be determined exclusively from the set of control points φ, as we lack information about the following two aspects:

a) the number and shape of moving objects in the scene;
b) the localization of each feature in respect to a moving object, which can be either on the boundary or inside the segmented object region.

The solution for a) comes from another motion cue: *the dense optic flow* estimation. We have already discussed the available dense OF estimation methods in chapter 3.2.2. Based on the results and discussions presented in chapter 3.2.4 we choose the Farnebäck estimator due to its robustness against noise and lighting condition changes and its relative accuracy compared to LK or HS methods. For the astute reader we mention that although more accurate dense OF estimation

methods exist based on the HS approach [102], they are simply too expensive to compute in real-time on a live video feed.

By applying the Farnebäck method to the grayscale representations G_{t-1} and G_t of two consecutive frames, we obtain the resulting displacement matrices U and V which correspond to pixel velocities along the x and y image axes. Using this result, we perform the following sequence of operations:

1. Generate the binary dense optical flow motion mask M_D:
$$M_D(p+\vec{v}) = \delta(|\vec{v}|, \tau_v) \tag{3.28}$$
where:
   - $p = [x \quad y]^T$ is the pixel position vector relative to image G_{t-1}
   - $\vec{v} = [U(p) \quad V(p)]^T$ represents the dense OF displacement of pixel p,
   - $|.|$ denotes the vector Euclidean length,
   - $\delta(x,\tau) = \begin{cases} 1, if\ x \geq \tau \\ 0, otherwise \end{cases}$, and
   - τ_v represents a velocity threshold, which is set to 0.5 in order to filter out subpixel displacements.
2. Apply connected components analysis to mask M_D, which produces a set of disjoint regions $\{S_1, S_2, \ldots, S_n\}$ which satisfy the condition
$$S_i \cap S_j = \emptyset, \forall i \neq j \bigwedge \forall q | M_D(q) = 1, \exists i \in \overline{1..n} | q \in S_i \tag{3.29}$$
3. For each region S_i, i=1..n:
   a. extract the contour $\mathcal{C}(S_i) = \rho^-(S_i)$ where ρ^- represents the internal morphological gradient;
   b. determine the set of sparse features $\varphi_i = \{f \in \varphi | f \in S_i\}$;
   c. generate the concave hull
   $$\hat{\mathcal{C}}(S_i) = \{f_j \in \varphi_i \mid f_j = \arg\min_f(\|p_j - f\|), p_j \in \mathcal{C}(S_i)\} \tag{3.30}$$
   where $\|.\|$ represents the Euclidean norm;
   determine the new region $\hat{S}_i | \rho^-(\hat{S}_i) = \hat{\mathcal{C}}(S_i)$.

Figure 3-8 contains a graphical representation of step 2 of the algorithm, which can be viewed as a distance transform [103] from the contours of dense OF regions to the concave hulls of enclosed sparse OF features. For each connected-component $S_i \subset M_D$, we search for the sparse features in the set φ that are closest to each of the points that make up the contour $\mathcal{C}(S_i)$. These features act as the concave hull of a new shape $\hat{S}_i$ that represents a moving region in the current frame.

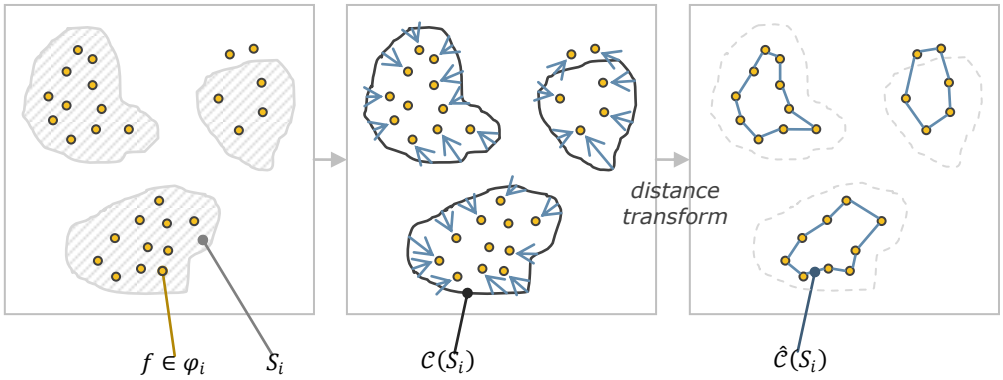


Figure 3-8. Graphical representation of the aggregation of dense and sparse OF information

3.3.4. Step 3. Extracting accurate moving object boundaries using active contours

Each concave hull $\hat{\mathcal{C}}(S_i), i = 1..n$ constructed in step 2 of our algorithm represents a polygon whose vertices are usually located at distances greater than 1 pixel. This has a negative impact on the smoothness of the extracted boundary along the edges of the polygon, since the polygon is an approximation of the actual moving region.

In order to obtain a smooth boundary for each region $\hat{S}_i$, which is represented by the body of the polygon $\hat{\mathcal{C}}(S_i)$, we use an active contours-based approach [104, 105]. The concept of active contours, also known in literature as snakes, has already been introduced in paragraph 2.3.3.2.2.

The snake is initialized from the set $\{p_j | p_j \in \rho^-(\hat{S}_i)\}$ and evolves iteratively towards the nearest discontinuities in the image $\mathrm{G(t)}$ while minimizing the energy functional defined in equation (2.5). Since the vertices of $\hat{\mathcal{C}}(S_i)$ are by definition located on the outer edges of the moving region, the snake converges to the nearest edges of this region that connect pairs of neighboring vertices.

In order to achieve snake convergence, two of the most popular algorithms have been compared: the *Greedy Snake* approach described in [106] and the *Gradient Vector Flow* (GVF) technique proposed by Xu and Prince [107].

The Greedy Snake algorithm [106] is known for its fast convergence and good results. The quality of the extracted boundary depends on the values selected for the weights α, β and γ, which control the relative importance of continuity, curvature and respectively edge attraction terms in the snake energy functional. Based on results available in the literature [108, 109] and experiments performed on our database of test videos, we have chosen the following values:

$$\alpha = 0.45, \beta = 0.65, \gamma = 0.50.$$

The GVF algorithm [107] computes a diffusion of the gradient vectors of a grayscale edge map, which allows the snake to converge inside concave boundary regions [110]. This feature comes at the expense of computation time, which is significantly larger in case of the GVF approach. Similar to the Greedy Snake algorithm, GVF relies on the weights α, β and γ to control the final snake appearance, with an additional regularization parameter μ that controls the tradeoff between smoothness and attraction to strong gradients in the vector flow energy formulation. In our experiments we have kept the values for the α, β and γ parameters same as for the Greedy snake, and set μ to the original value recommended by the authors, $\mu = 0.2$.

The output of the snake algorithm is a new set of contours $\mathcal{C}_i^S = Snake\left(G_t, \rho^-(\hat{S}_i), \alpha, \beta, \gamma\right), i = 1..n$. The body of each contour $\mathcal{C}_i^S$ corresponds to a moving region in the image, denoted with S_i^S. The results produced by the two snake convergence algorithms are demonstrated in Figure 3-9, for frame #9 of the *Green Screen 3* sequence.

As seen from the obtained contours, the GVF method has the tendency to follow object concavities, as seen on the person's left shoulder, where the snake pursued the higher magnitude gradients associated with folds in the clothing. Surprisingly, the Greedy snake reacted better at the region between the neck and the right shoulder and produced a more complete and accurate object segmentation. Combined with a computation time one order of magnitude smaller than the one of GVF, this result turns the balance in favor of the Greedy snake, which becomes the algorithm of choice throughout the rest of our thesis.

The algorithm concludes by building the *aggregated motion mask* that will serve as input to the TSM algorithm, as the reunion of all S_i^S regions:

$$M^A = \bigcup_{i=1}^{n} S_i^S \tag{3.31}$$

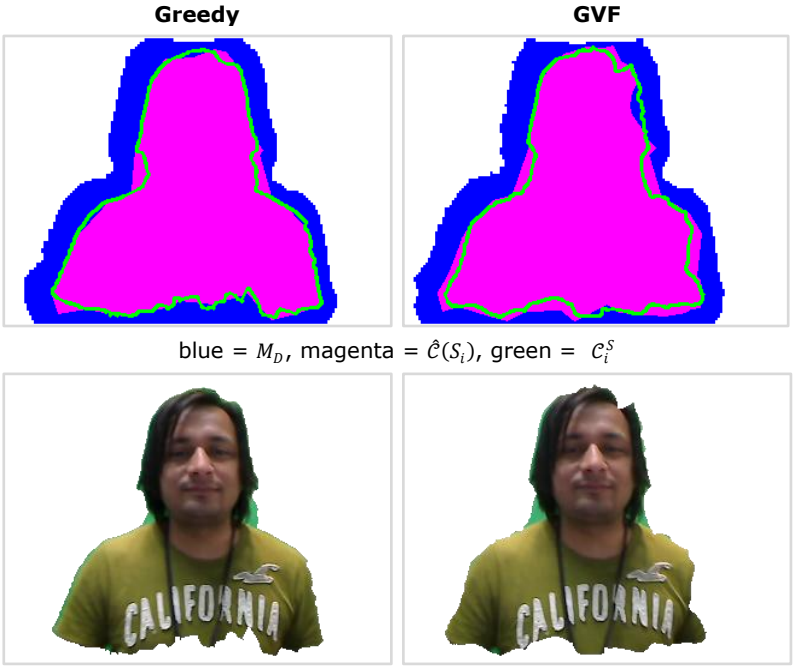


Figure 3-9. Comparison between Greedy and GVF snake results

3.3.5. Motion segmentation results

The output results of the described dense and sparse OF aggregation algorithm are illustrated in Figure 3-10 for a set of different video sequences.

We can observe that the moving region M^A has been segmented with pixel accuracy in the *Green Screen 3* and *Avatar* sequences, which contain simple backgrounds and FG regions that possess a large number of trackable features. For the *Mihai 2* sequence, the region is also accurate but under-segmented compared to the actual moving person; this is due to the presence of video encoding artifacts that affect some of the image edges and the presence of smooth regions (shirt) that prevent accurate OF computation. In the classic *Hamburg Taxi* sequence the 3 cars have been detected by our method, even if the sequence is extremely noisy. We note, however, that the moving FG regions have been only partially outlined due to insufficient sparse OF features, and leaking BG regions can be observed near the white car.

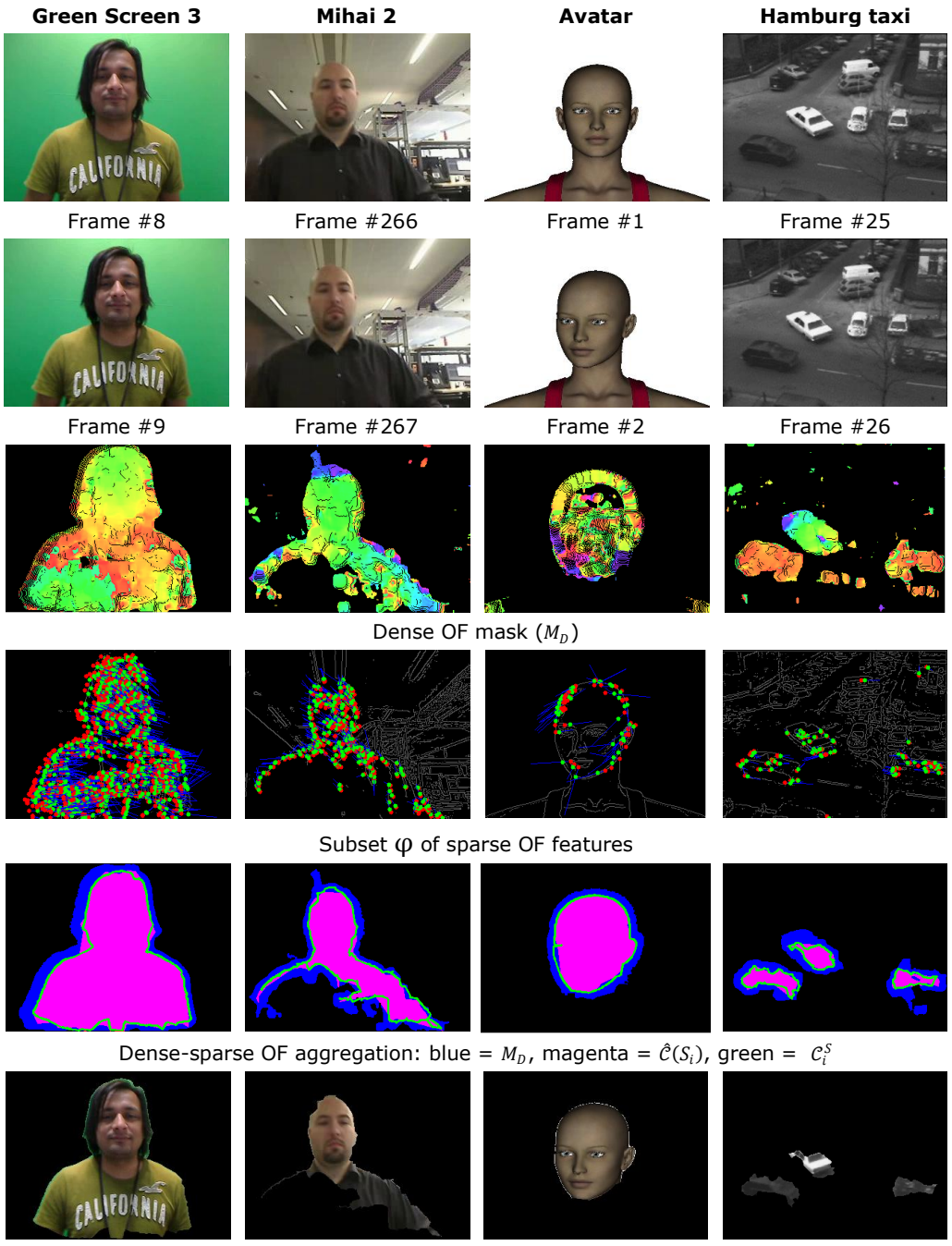


Figure 3-10. Results obtained by aggregating dense and sparse OF information

Based on the results presented above and experiments performed on our database of test video sequences we can conclude that the our novel algorithm for dense and sparse optical flow aggregation produces the expected results, especially in case of large foreground regions. This makes the algorithm suitable for use in videoconferencing applications where participants occupy a significant part of the scene, in order to provide accurate motion cues for FG segmentation.

3.4. Summary

In this chapter we have introduced a new approach to accurate motion segmentation which combines the results produced by dense and sparse OF estimations in order to extract the moving regions between two consecutive frames in a monocular video sequence. By combining the two OF estimation techniques we are able to overcome the drawbacks of each method if taken individually, the result being a pixel-accurate set of moving regions that is resilient to noise and video compression artifacts [71].

The *aggregated motion mask* produced by the algorithm provides essential information about the FG objects present in the observed scene. However, this information is valid exclusively for the moment t at which it was computed. In order to benefit from it over a longer timeframe, a method is required which enables a FG extraction system to incrementally build and update the FG image based on the motion data obtained at different moments. In the next chapter we will describe how to achieve this desiderate through the concept of *temporally stable masks*.

4. TEMPORALLY STABLE MASKS: A METHOD FOR TEMPORAL INTEGRATION OF DETECTED MOTION

4.1. Introduction

In chapter 3 we have introduced and detailed a new method capable of producing an accurate image of motion detected between pairs of consecutive frames in a video sequence. The aggregated motion mask M^A obtained through this method represents a *positive motion prior*, which exposes a part of the real FG and is valid only at the time of its creation.

In order to achieve a coherent FG mask over a longer, ideally indefinite period of time, we want - at any given moment t - to be able to integrate into a single *temporally stable mask*, TSM_t, all motion cues $\{M_1^A, M_2^A, \dots M_t^A\}$ obtained until that particular moment. By this definition, we can express TSM_t as:

$$TSM_t \coloneqq f(M_i^A, i = \overline{1..t}) \tag{4.1}$$

In an ideal case, TSM_t is free of false positives and contains the current positions of all FG pixels that were exposed by at least one motion cue by the moment t, for any t ≥ 1. This enables us to express TSM_t based on its value known at t-1:

$$TSM_t \coloneqq f(TSM_{t-1}, M_t^A) \tag{4.2}$$

with the mention that the actual formulation of TSM_t may depend on additional parameters not expressed at this stage. This is the equivalent of incrementally building and updating an image of the FG detected through motion.

Once an image region is exposed as FG by a certain motion cue M_i^A, it will be persisted as part of $TSM_{t \geq i}$ until we gather sufficient evidence that it has changed enough as not to be accounted for as FG anymore. In order to achieve this behavior we propose an approach based on image statistics to compute a *negative motion prior* that can invalidate a previous FG assumption for a given image region.

The chapter starts by introducing the statistical models used to compute the negative motion prior, followed by a description of the TSM algorithm. We conclude with a discussion of the strengths and weaknesses of the proposed method based on experimental results.

4.2. Image statistics as indicators for foreground labeling and persistence

In order to compute the image statistics we divide the image plane I into equally-sized blocks of $m \times n$ pixels, which results in a total of $M \times N$ blocks. The block approach offers higher robustness against noise and lighting changes [31, 35] and in our experiments we have chosen $m = n = 4$.

In every block $B_{i,j} \subset I$ we model the pixel intensities independently for every image color channel. We also define a *similarity measure* between the pixel

intensities in two different images I_1 and I_2, which allows us to compute the negative motion prior for each image block. Formally, the similarity measure is expressed as:

$$\delta_{i,j,c}(I_1, I_2) = \begin{cases} 1, if\ d_{i,j,c}(I_1, I_2) < \tau_d \\ 0, otherwise \end{cases} \quad (4.3)$$

where d is a distance function between two intensity distributions and τ_d is a threshold which depends on the chosen distance function d. The concept is illustrated in Figure 4-1 for block pixel intensities modeled using a Gaussian kernel.

Based on the above considerations, the blockwise temporal dissimilarity between two consecutive video frames I_t and I_{t-1} can be defined as:

$$\mathcal{D}_{i,j}^t = \max_{c=1..C} d_{i,j,c}(I_t, I_{t-1}). \quad (4.4)$$

When $\mathcal{D}_{i,j}^t < \tau_d$ we achieve *persistence* of the block label; if $\mathcal{D}_{i,j}^t \geq \tau_d$, the temporal similarity measure becomes a negative prior that *invalidates* the current block label.

In the context of labeling persistence we have evaluated two alternatives for modeling the block pixel intensities. The first model is based on a *Gaussian kernel*, while the second relies on a more advanced concept called *structural similarity (SSIM)*.

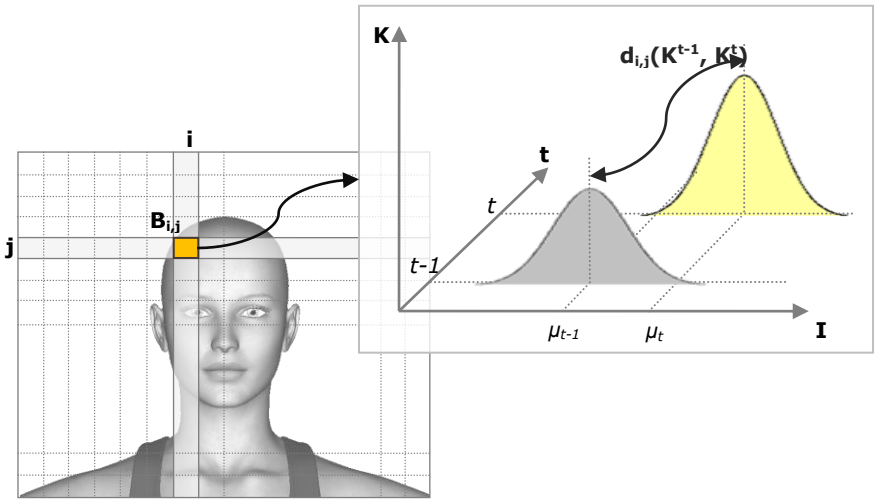


Figure 4-1. Statistical image modeling to support FG/BG labeling and persistence

4.2.1. Gaussian model approach

This approach models the pixel color intensities in every block $B_{i,j}$ using a *Gaussian kernel*:

$$K_{i,j,c} \sim \mathcal{N}(\mu_{i,j,c}, {\sigma^2}_{i,j,c}), i = \overline{1..M}, j = \overline{1..N}, c = \overline{1..C} \quad (4.5)$$

where

$$\mu_{i,j,c} = \frac{1}{m \cdot n} \sum_{\boldsymbol{x} \in B_{i,j}} I^c(\boldsymbol{x})$$

$${\sigma^2}_{i,j,c} = \frac{\sum_{\boldsymbol{x} \in B_{i,j}} \left(I^c(\boldsymbol{x}) - \mu_{i,j,c}\right)^2}{m \cdot n} = \frac{\sum_{\boldsymbol{x} \in B_{i,j}} I^c(\boldsymbol{x})^2}{m \cdot n} - {\mu_{i,j,c}}^2$$

represent the *mean* and respectively the *variance* of the intensity values. For clarity, in case of single-channel images we will omit the channel index c.

The next step is to define the distance between two Gaussian kernels, in accordance with equation (4.3). For our algorithm we substitute d with the *Hellinger distance (H)* [111], which produces a result in the [0, 1] domain. A distance close to 0 indicates a high degree of similarity between two probability distributions, while a value of 1 means total dissimilarity. The Hellinger distance is related to the *Bhattacharrya coefficient (BC)* [112], which has been previously used in the literature as a similarity indicator [60; 113].

$$d_{i,j,c}(I_1, I_2) = H\left(K^1_{i,j,c}, K^2_{i,j,c}\right) = \sqrt{1 - BC\left(K^1_{i,j,c}, K^2_{i,j,c}\right)}, \text{ where}$$
$$BC(K_1, K_2) = \sqrt{\frac{2\sigma_1\sigma_2}{{\sigma_1}^2 + {\sigma_2}^2}} \cdot e^{-\frac{1}{4}\frac{(\mu_1-\mu_2)^2}{{\sigma_1}^2+{\sigma_2}^2}} \tag{4.6}$$

The similarity threshold $\tau_{d=H} = 0.7$ has been determined experimentally using our test video database.

4.2.2. Structural similarity approach

Structural similarity (SSIM) has been introduced as a quality measure for assessing the level of visual degradation caused by image manipulation during different stages such as acquisition, processing, compression, storage and transmission [114]. The structural information for every image pixel is obtained from local luminance and contrast and is then compared between the original and the altered versions of an image in order to quantify their similarity.

Taking in consideration the initial SSIM implementation of [115], the block pixel intensities may be modeled using the same Gaussian distribution described in formula (4.5). In this case, the distance d in equation (4.3) can expressed as:

$$d_{i,j,c}(I_1, I_2) = \frac{1 - SSIM\left(K^1_{i,j,c}, K^2_{i,j,c}\right)}{2};$$
$$SSIM(K_1, K_2) = \frac{(2\mu_1\mu_2 + C_1)(2\sigma_{12} + C_2)}{({\mu_1}^2 + {\mu_2}^2 + C_1)({\sigma_1}^2 + {\sigma_2}^2 + C_2)} \tag{4.7}$$

where:

- $\sigma_{12} = \frac{\sum_{x \in B_{i,j}} (I_1^c(x) - \mu_1)(I_2^c(x) - \mu_2)}{m \cdot n}$ is the *covariance* between pixel intensity values of block $B_{i,j}$ in images I_1 and I_2 respectively;
- $C_1 = (k_1 L)^2$ and $C_1 = (k_2 L)^2$ are stability constants, with L being the dynamic range of the pixel values (= 255 for 8 bit per channel images), $k_1 = 0.01$ and $k_2 = 0.03$.

The SSIM measure produces values in the range $[-1,1]$, where 1 is achieved on completely identical datasets. In turn, distance d takes values in the interval $[0,1]$ where, as expected, a value of 0 means no difference between datasets and a value of 1 denotes total dissimilarity.

In order to avoid blockiness in the resulting SSIM index map, [114] propose an improvement, by applying a 11×11 Gaussian weighting function $\mathbf{w} = \{w_k | k = \overline{1..N}\}$ with $\sigma = 1.5$ when computing the local statistics centered on a pixel $\boldsymbol{x} \in I$. The estimates for the local mean, standard deviation and covariance are now computed as:

$$\mu_{\boldsymbol{x},c} = \frac{1}{N}\sum_{k=1}^{N} w_k I^c(x_k) \tag{4.8}$$

$$\sigma^2{}_{\boldsymbol{x},c} = \frac{1}{N}\sum_{k=1}^{N} w_k \left(I^c(x_k) - \mu_{\boldsymbol{x},c}\right)^2$$

$$\sigma_{12}{}_{\boldsymbol{x},c} = \frac{1}{N}\sum_{k=1}^{N} w_k \left(I_1^c(x_k) - \mu_{1}{}_{\boldsymbol{x},c}\right)\left(I_2^c(x_k) - \mu_{2}{}_{\boldsymbol{x},c}\right)$$

By substituting the new formulas from (4.8) in (4.7), we obtain the structural similarity index computed between I_1 and I_2 on color channel c on a 11×11 window centered on pixel $\boldsymbol{x}$, denoted with $SSIM_{\boldsymbol{x},c}(I_1, I_2)$.

The improved technique provides locally isotropic SSIM measurements for every image location. In this case, distance d can be expressed in terms of mean SSIM (MSSIM), as:

$$\begin{aligned} d_{i,j,c}(I_1, I_2) &= \frac{1 - MSSIM_{i,j,c}(I_1, I_2)}{2}; \\ MSSIM_{i,j,c}(I_1, I_2) &= \frac{1}{m \cdot n}\sum_{x \in B_{i,j}} SSIM_{x,c}(I_1, I_2) \end{aligned} \tag{4.9}$$

Same as for the Gaussian approach described in the previous paragraph, we have determined $\tau_{d=SSIM} = 0.35$ based on experiments performed on our database of test videos.

4.3. Spatio-temporal motion segmentation algorithm

4.3.1. Algorithm flow

Having defined two motion cues, the *spatial one* represented by the motion mask M^A and the *temporal one* given by the similarity measure δ, we can now outline the sequence of our S-T motion segmentation algorithm as a flow diagram, as shown in Figure 4-2.

The algorithm flow is independent of the approach taken towards modeling the pixel intensities, and can accommodate other models than those described in section 4.2. For every block $B_{i,j}$ the algorithm checks if significant motion has been detected, according to the following rule:

$$\left|\{x \in B_{i,j} | M^A(x) = 1\}\right| \geq \tau_m \tag{4.10}$$

where τ_m is a threshold specifying the minimum number of "moving pixels" required to label a block as FG. Experiments have shown that $\tau_m = 0.33$ provides good results especially along object borders.

If sufficient motion is present in the block, it will be labeled as FG. Otherwise, the similarity measure $\mathcal{D}_{i,j}^t$ is computed and compared with the similarity threshold τ_d. If the temporal instances of the block are not similar, the label is set to BG and the algorithm moves to the next block. If similarity is confirmed, the algorithm applies a special step dedicated to border region handling which allows to decide between persisting the current block label or defaulting to BG.

After all blocks have been processed, a morphological cleaning operation is applied in order to filter out the small FG / BG regions from the resulting TSM.

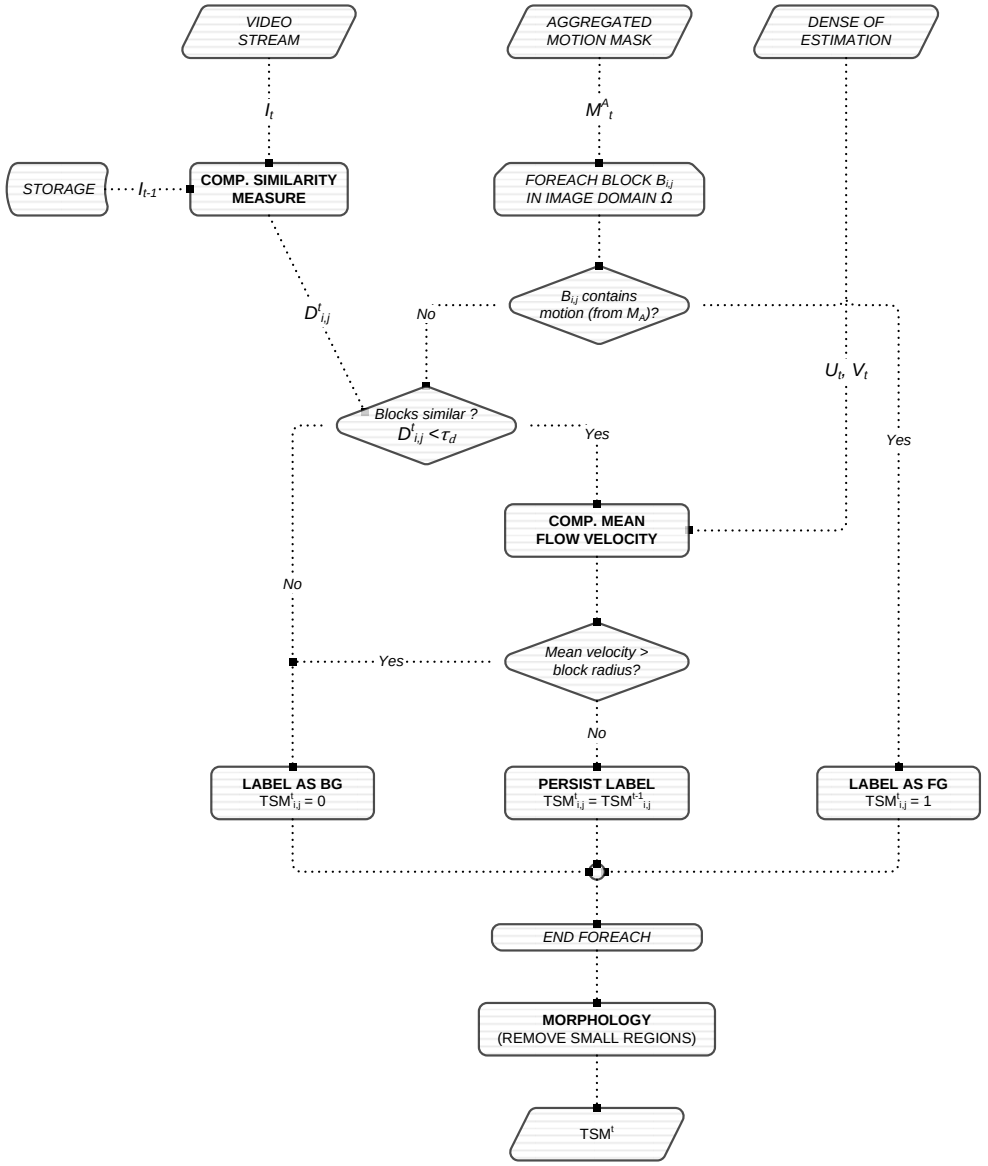


Figure 4-2. Flow diagram for the TSM algorithm

We complete the above description with the pseudocode of the TSM algorithm:

```
procedure UpdateTSM(I_t, I_t-1, U_t, V_t, τ_d, τ_m) {
      foreach (block B_i,j in I) {
            // calculate statistics
            D_i,j = 0;
            foreach (channel c in channels(I)) {
                  d_i,j,c = DistanceMetric(B_i,j, c, I_t, I_t-1);       // eq. (4.6) or (4.9)
                  D_i,j = max(D_i,j, d_i,j,c);                          // eq. (4.4)
            }
```

```
            μU = mean(Ut, Bi,j);                              // mean X OF displacement
            μV = mean(Vt, Bi,j);                              // mean Y OF displacement

            // integrate motion priors
            if (count(Bi,j, x => MA(x) ≠ 0) >= τm) {              // eq. (4.10)
                  // *** positive prior ***
                  TSM(i,j) = FG;
            }
            else {
                  // *** negative motion prior ***
                  if (Di,j >= τd) {
                        // no similarity between previous and current blocks
                        TSM(i,j) = BG;
                  }
                  else if (μU >= m/2 || μV >= n/2) {
                        // motion leaving a block not covered by MA
                        TSM(i,j) = BG;
                  }
                  // *** else preserve current label ***
            }
      }    // end foreach

      // eliminate small FG blobs
      foreach (CC in ConnectedComponents(TSM)) where (Area(CC) < τAREA) {
            Label(CC, BG);
      }

      // eliminate small BG blobs
      foreach (CC in ConnectedComponents(Not(TSM))) where (Area(CC) < τAREA) {
            Label(CC, FG);
      }
}
```

As seen in Figure 4-2 and the pseudocode, the algorithm includes an heuristic approach which is detailed next. It is worth mentioning that although this approach does not produce the same level of results in all possible scenarios, the overall quality of the TSM is improved in a significant manner.

4.3.2. Border region handling

A special case that we have encountered during the development of the TSM algorithm is related to the way the positive motion prior is applied. A TSM block is labeled as FG if at least τ_m of its pixels belong to the FG defined by the motion mask M^A. This can result in TSM blocks around the border of moving regions that carry a FG label although they are largely composed of actual BG pixels.

An issue arises when, in a subsequent frame, the FG region moves away from its previous location, as shown in Figure 4-3. In such cases the previous border blocks found at the far end of the moving region may not change their intensity model enough to trigger a FG -> BG label change, since they are mostly composed of actual BG pixels. By keeping their previous FG label, these blocks will cause false positives and unwanted artifacts in the TSM.

In order to solve this issue, we introduced an additional condition as part of the negative motion prior. We first compute the mean dense OF displacement $\bar{h}_{i,j} =< \bar{u}_{i,j}, \bar{v}_{i,j} >$ in each block not currently covered by the motion mask M^A. If any of the X or Y-axis components of the mean block displacement has a magnitude larger that the block radius ($m/2$ and respectively $n/2$), we mark the block as BG.

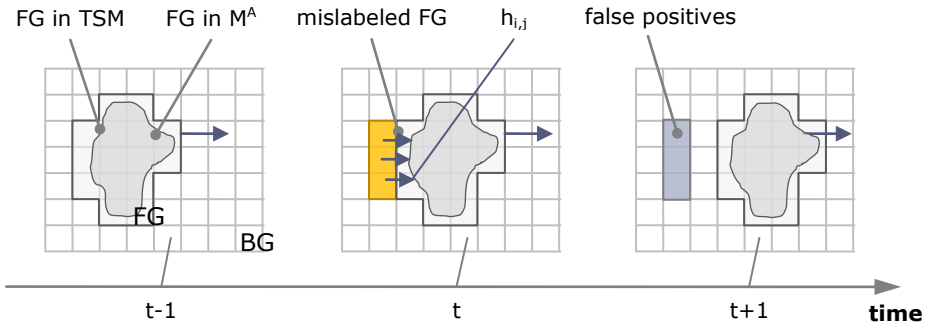


Figure 4-3. Misclassification of border regions

4.4. Experimental results

We have tested the TSM algorithm on a database of 40+ video sequences, most of them reproducing videoconference scenarios in different environments and several containing randomly captured scenes. Figure 4-4 shows a selection of the results obtained using the TSM algorithm in conjunction with Gaussian and respectively SSIM statistical models.

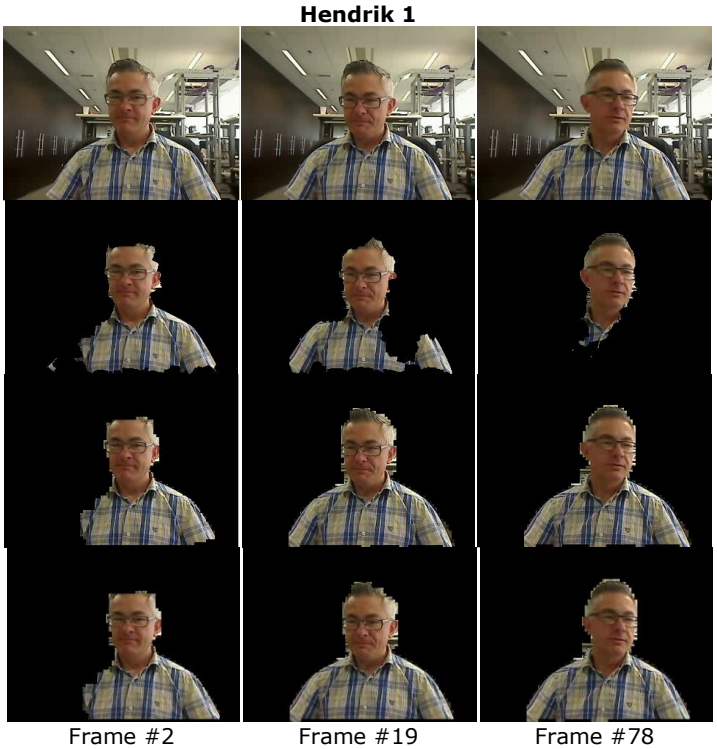


Figure 4-4. Selected results produced by the TSM algorithm: 1st row - input frame; 2nd row - positive motion prior; 3rd row –TSM using Gaussian Model; 4th row – TSM using SSIM.

TV Show

Frame #9 Frame #14 Frame #43

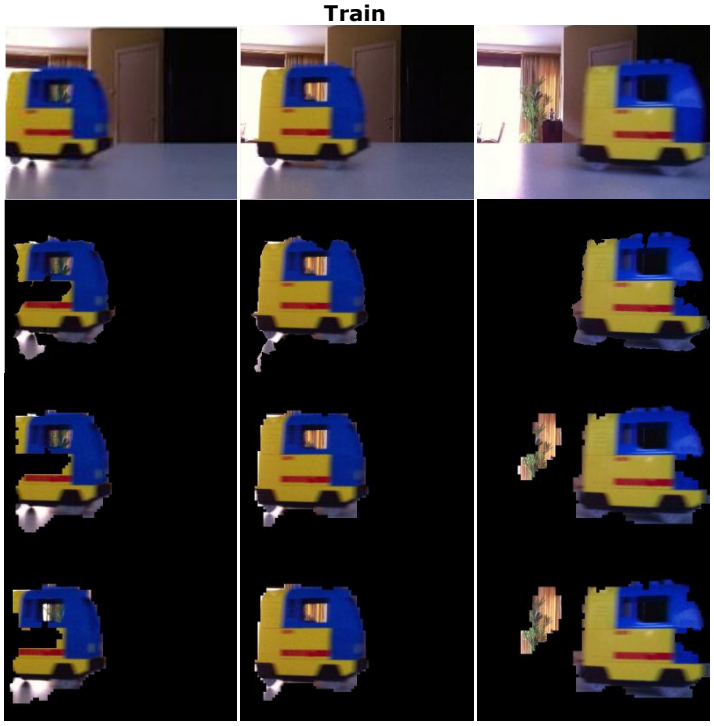
Green Screen 3

Frame #2 Frame #53 Frame #80

Mihai 3

Frame #154 Frame #270 Frame #279

Train



Frame #82 Frame #86 Frame #99

4.5. Discussion

By analyzing the results shown in Figure 4-4, we can make the following comments:

- The *Hendrik 1* sequence represents a good case, in which the TSM converges very fast to the actual FG due to the presence of significant and complementing motion cues. These motion cues draw their quality from the fact that the person's clothing presents a rich texture on which OF can be accurately estimated. Even in the case of a cluttered background and compression artifacts, the TSM maintains a lock on the FG as it can be seen in frame #78 when the person turns his head. It can also be observed that both Gaussian and SSIM approaches produce identical results.

- The *TV Show* sequence is another case of good convergence and stability, due to the chromatic difference between BG and FG. The presenter's hair and clothes exhibit a fine texture and a reflectance factor that allow good OF estimation. In parts of the scene where the FG-BG transition is very smooth (e.g. presenter's left shoulder) the TSM presents a small instability. Again, there is virtually no difference between the results obtained using the Gaussian similarity and those built using SSIM.

- In the *Green Screen 3* sequence the person's boundary is accurately determined due to the presence of strong border edges. We can see in frame #53 the first sign of TSM instability in respect to smooth regions, represented by the small border hole in the person's T-shirt. In the same frame there is fast hand motion which causes a *motion blur* effect with negative impact on the sparse OF estimation: the person's left arm contains no trackable sparse features, which eliminates a part of the arm from the aggregated motion cue. In frame #80 there is both *motion blur* and *occlusion* caused by the right hand motion, which causes most of the right arm to be erased from the TSM (although it is recovered 1 frame later due to the arrival of new motion cues). This is the first sequence in which the difference between Gaussian and SSIM models becomes visible. SSIM is able to better persist smooth image regions, as seen in frame #53, but the same tendency can cause negative effects, such as the background leaking region in the bottom-right corner of frame #80.

- The *Mihai 3* sequence is one of the most challenging cases, which illustrates the drawbacks of the TSM algorithm. The smoothness and low reflectivity of the author's shirt significantly impacts both the positive motion prior and the negative motion prior calculated in the TSM. As a result, smooth regions on the person's torso and sleeves are quickly discarded from the TSM due to their statistical instability. In frame #270 we notice false positives caused by the intense white light on top and left of the scene; once such a stable BG region makes its way into the FG, it will persist there until it is removed by an occlusion. In frame #279 the person's face is persisted from previous cues as the hand moves into the scene. In this challenging sequence SSIM overtakes the Gaussian model due to its ability to better persist the smooth regions on the person's clothing, at the expense of a slightly higher background leaking effect.

- In the *Train* sequence we are faced with continuous and blurred motion, as well as noise present in the medium and low-lit areas. Since FG colors stand out compared to the BG, the moving object is properly identified from frame to frame. The result is also very robust to noise. Between frames #86 and #99 a misclassified FG region appeared which was caused by the feature-rich plant in the BG being disoccluded by the passing train. Sparse OF features were wrongly tracked by the KLT algorithm from the back of the train to the plant, causing the TSM to expand into the BG. Once the expansion occurred, the region's extreme statistical stability kept it as part of the FG as the train passed by. Like for the *Hendrik 1* and *TV Show* sequences, the differences produced between the Gaussian and the SSIM versions of the TSM are minor.

Another important aspect observed during our experiments is related to the negative influence of occlusions of the quality of the TSM. Occlusion happens when a moving or persisted region A that is marked as FG at moment t-1 is partially or totally covered by another moving region B, also marked as FG, at moment t. After B eventually uncovers A at moment $t+\tau$, the occluded blocks in A will be labeled as BG if the occluded and occluding regions (from A and respectively B) have dissimilar statistics. This happens, according to equation (4.3), between moment $t+\tau-1$ which precedes the disocclusion of A by B and $t+\tau$ when the previous FG belonging to A is uncovered.

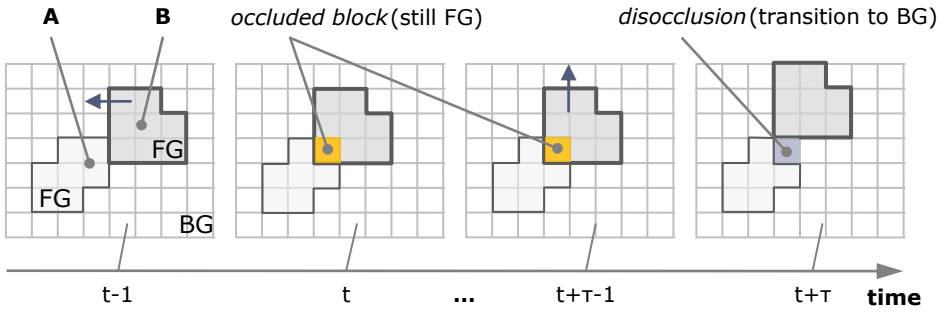


Figure 4-5. The effect of occlusions on the TSM

Such a scenario occurs frequently in videoconferences when a participant gesticulates or performs different hand gestures that are recognized by the system. In this case, the moving region B represents the person's hand, while region A can be the person's body or another FG scene object. The occlusion case is illustrated schematically in Figure 4-5.

Based on the previous observations and the list of challenging OF situations detailed in chapter 3.2.4.1, we are now able to summarize in Table 4-1 the advantages and the drawbacks of the TSM-based approach to motion segmentation.

From the assessment presented in Table 4-1 we can conclude that 3 of the 7 challenging situations to OF estimation (detailed in chapter 3.2.4.1) have been successfully addressed by combining the method for dense and sparse optical flow aggregation introduced in chapter 3 with the TSM algorithm. We have also achieved our goal of model-less extraction of FG exposed through motion, in line to the research objectives presented in chapter 1.3. From the 5 identified drawbacks, 4 are

well known in literature and represent a constant preoccupation of research communities [93, 94, 96, 116]. Only the 4th disadvantage and part of the 2nd are related to our particular approach, for which we need to pursue new directions, as described in the following chapters.

Table 4-1. Advantages and disadvantages of the TSM-based S-T motion segmentation

Advantages	Disadvantages
1. robustness to smooth illumination changes, unless abrupt and significant lighting changes take place	1. sensitivity to camera shakiness, which can introduce a large amount of false positives
2. robustness to noise and compression artifacts	2. occlusions may erase parts of the TSM FG (depending on the FPS rate of the captured video)
3. model-less FG extraction, free of *a priori* assumptions related to moving objects	3. difficulty persisting smooth FG regions
4. accurate motion cues, presenting a low rate of false positives	4. cumulated errors in the TSM can generate resilient false positives
5. good handling of cluttered BG scenarios	5. sensitivity to fast motions and motion blur
6. accurate FG extraction in case of sufficient texture and / or reflectivity on FG objects	

Another important remark is that the proposed TSM represents a *coarse mask*, which has a lower resolution than the original input image. This resolution is imposed by the block size used to compute the image statistics, which corresponds to just one pixel at the TSM level. If we consider the role of the TSM, which is to ensure FG persistence across a video sequence, a lower-resolution mask can be sufficient if we find a method to translate it into a pixel-accurate FG mask for the input image.

4.6. Summary

In this chapter we have presented a novel algorithm for S-T motion segmentation in video sequences, which relies on motion cues obtained by aggregating dense and sparse optic flow estimations. We introduced the concept of *temporally stable masks* - designed to incrementally build an image of the FG exposed through motion without the use of any *a priori* models or assumptions related to the persons or objects present in the observed scene – and have shown its capabilities of persisting FG information by using image statistics.

In accordance with the 3rd objective of our thesis, the role of the TSM algorithm is oriented towards recording, persisting and updating the FG exposed through motion rather than providing a complete FG mask for a video sequence.

Given the strengths and weaknesses presented in section 4.5, the TSM alone cannot provide an accurate FG/BG segmentation for a videoconference, except for some favorable scenarios; however, it can supply at any time a confident FG estimation that can be used to generate such a segmentation.

With this remark we now move to the next chapter which addresses the 4th objective of our thesis, related to obtaining a pixel-accurate FG segmentation based on the foreground information supplied by the TSM.

5. AN HEURISTIC APPROACH TO UNSUPERVISED GRAPH CUT SEGMENTATION FOR ACCURATE OBJECT EXTRACTION IN VIDEO CONFERENCE SCENARIOS

5.1. Overview

By exploiting the strong points of both dense and sparse OF, the motion detection algorithm described in chapter 3 was demonstrated to produce accurate results that are resilient to noise, compression artifacts and smooth lighting variations. As the approach relies on OF estimation, the algorithm's weakness lies in the handling of smooth, low-reflective image regions for which OF is known to produce unreliable results.

The same factors described above affect the stability of the TSM, preventing it from persisting smooth regions with low color and texture in the FG. Coupled with the influence of occlusions and motion blur, the TSM can exhibit an under-segmentation effect. Based on these considerations, we have concluded at the end of chapter 4 that the TSM alone cannot provide the final FG mask for an immersive videoconferencing scenario, but can successfully act as a confident indicator of FG detected through motion and an intermediate step towards the sought-after pixel-accurate segmentation.

The present chapter is dedicated to the pixel-accurate segmentation process that is responsible of regenerating the FG image starting from the TSM information. This is also the final processing stage employed by our FG extraction method. The main goal is to reconstruct the missing parts of the TSM based on color and contrast cues, taking into account the potential similarity between FG blocks and adjacent BG ones.

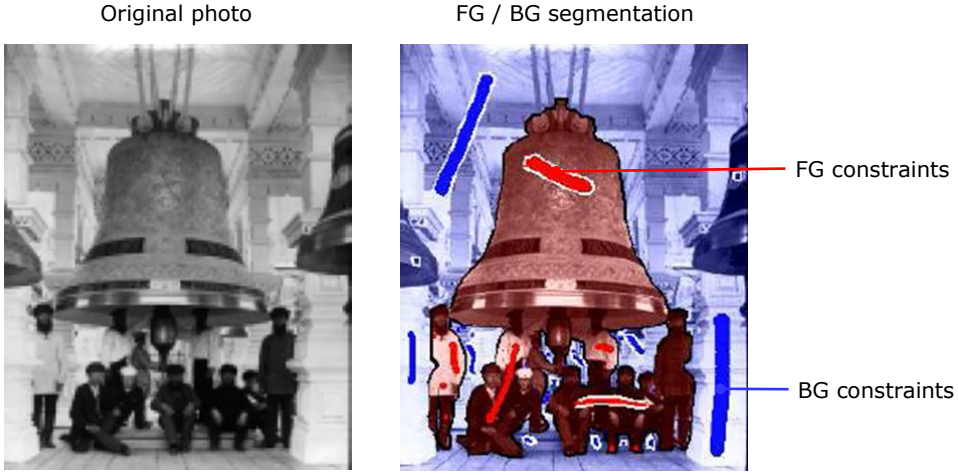


Figure 5-1. Constrained graph cut segmentation (source: [53])

In this respect, many of the recent state-of-art segmentation techniques presented in section 2.2.2 [17-20] have proven the effectiveness of the *graph cut* approach introduced in [11]. This has steered our research towards integrating a graph cut algorithm as a final step in the FG extraction process.

In the literature, graph cut has been known to produce very accurate results in applications that rely on *interactive segmentation* [117-119]. In these scenarios, the user is required to provide FG and BG seeds that are treated a hard constraints by the graph cut algorithm and used as starting points when performing the segmentation. An example of user-defined hard constraints for an image is given in Figure 5-1. *Unsupervised GC segmentation* has also been addressed with similar qualitative results [120; 121], but so far the proposed algorithms run only on still images as their complexity and running times (in the order of seconds) make them prohibitive for use in real-time video sequence processing.

While aiming to achieve the same level of accuracy in our approach, we require the means of generating the hard constraints in an automatic manner, starting from the information that is already available from the OF and TSM information. The research direction described by this statement corresponds with the 4th and final objective of our thesis.

The chapter is composed of two major sections. The first section is a presentation of the general concepts behind the graph-cut algorithm and its applicability to image segmentation. In section two, we introduce a novel algorithm for the automatic generation of hard constraints as input for an unsupervised graph-cut segmentation, based on the information provided by sparse OF estimation and the TSM.

5.2. Graph cut-based image segmentation

5.2.1. Introduction

Markov Random Fields (MRFs) have been widely used in statistical image analysis in order to estimate spatially varying quantities such as intensity or texture from noisy measurements [122]. In a Bayesian framework, estimating such a quantity is equivalent to finding its *maximum a posteriori* (MAP) probability [123].

Considering y to be a certain hypothesis, its posterior probability given a set of observations $\mathbf{x}$ can be computed using the Bayesian rule [124]:

$$P(y|\mathbf{x}) \propto P(\mathbf{x}|y) \cdot P(y) \ .$$

$P(y)$ denotes the prior probability of y and is used to model the contextual constraints placed upon the hypothesis. $P(\boldsymbol{x}|y)$ is the likelihood probability, which models the noise in the measurements [122]. In this context, the MAP of y can be expressed as:

$$\hat{y} = \arg\max_y P(y|\mathbf{x}) \ .$$

According to Hammersley-Clifford theorem which describes the Markov-Gibbs equivalence, $P(y)$ is characterized by a Gibbs distribution, and can be formulated as [124]:

$$P(y) = \frac{1}{Z} e^{-\frac{1}{T}E(y)} \tag{5.1}$$

where Z is a normalization constant, T is aglobal control parameter called temperature and $E(y)$ represents the prior energy.

Similarly, the likelihood probability can be expressed in exponential form:

$$P(\mathbf{x}|y) = \frac{1}{Z_x} e^{-E(\mathbf{x}|y)}$$

which ensures that the posterior probability obeys the Gibbs distribution:

$$P(y|\mathbf{x}) = \frac{1}{Z_E} e^{-E(y|\mathbf{x})}$$

with the posterior energy expressed as:

$$E(y|\mathbf{x}) = \frac{1}{T} E(y) + E(\mathbf{x}|y) \tag{5.2}$$

Based on the above considerations, finding the MAP can be expressed as a problem of minimizing the energy $E(y|\mathbf{x})$:

$$\hat{y} = \arg\min_{y} E(y|\mathbf{x}) \tag{5.3}$$

The literature describes a multitude of methods for solving equation (5.3), including dynamic programming, belief propagation [125] or graph cuts [11]. We have oriented towards the graph cut approach since it is known to produce faster and more accurate results compared to other methods [126].

5.2.2. Pixel labeling using pair-wise CMRF models

In the context of image analysis, the MRF can be described using the following components [122]:

- A set $\mathcal{P}$ of sites distributed in a rectangular lattice, with each element corresponding to an image pixel.
- A neighborhood $\mathcal{N} = \{\mathcal{N}_p \mid p \in \mathcal{P}\}$, where each element $\mathcal{N}_p$ contains the neighboring sites of p.
- A set $Y = \{Y_p \mid p \in \mathcal{P}\}$ of random variables. Each random variable takes a value y_p from a possible set of labels Ω.

In the particular case of binary FG / BG segmentation, a 4- or 8-connected neighborhood is considered and the set of labels can be defined as $\Omega = \{BG = 0, FG = 1\}$. We denote a possible segmentation with $y = \{y_p \mid p \in \mathcal{P}\}$, named a *configuration* in MRF terminology.

Considering an image of size $w \times h$, in total there are $|\Omega|^{w \times h}$ possible configurations. A MRF defined over a 4-connected neighborhood will require $(w \times h)^2 \times |\Omega|$ parameters for the unary terms that model the likelihood and $2 \times ((w-1)h + (h-1)w) \times |\Omega|^2$ parameters for the pairwise terms associated with the prior probability. This extremely large parameter space explains why in practice the aim is to find the most probable labeling $\hat{y}$, which leads to the MAP / energy minimization approach.

The prior energy $E(y)$ associated with a particular configuration can be formulated in terms of clique potentials defined over a subset of the random variables:

$$E(y) = \sum_{c} \psi_c(y_c)$$

where c represents a clique, ψ_c is the clique potential describing the prior probability of a particular realization of y_c, and y_c is a subset of y defined over c. In our particular case, clique potentials involve pairs $\{\mathrm{p}, \mathrm{q}\}$ of neighboring pixels, so that the prior energy can be expressed as:

$$E(y) = \sum_{\{\mathrm{p},\mathrm{q}\} \in \mathcal{N}} B_{p,q} \cdot \delta_{y_p \neq y_q} \tag{5.4}$$

where $B_{p,q}$ is a smoothness term and $\delta_{y_p \neq y_q} = \begin{cases} 1 \; if \; y_p \neq y_q \\ 0 \; otherwise \end{cases}$ is a Kronecker delta function that represents pixel interaction potential.

The likelihood energy can also be expressed as a sum of data terms R_p associated with each site [11]:

$$E(\mathbf{x}|y) = \sum_{\mathrm{p} \in \mathcal{P}} R_p(\mathbf{x}; y_p) \tag{5.5}$$

The set of observations $\mathbf{x}$ is gathered from the features in the image (i.e. color, texture).

By substituting the energy terms in equation (5.2) with their corresponding formulae from (5.4) and (5.5), we obtain the posterior energy functional that corresponds to the labeling y of all pixels in the image:

$$E(y|\mathbf{x}) = \frac{1}{T} \sum_{\{\mathrm{p,q}\} \in \mathcal{N}} B_{p,q} \cdot \delta_{y_p \neq y_q} + \sum_{\mathrm{p} \in \mathcal{P}} R_p(\mathbf{x}; y_p)$$

In the literature, this equation is usually encountered in the form:

$$E(y) = \lambda \cdot \sum_{\mathrm{p} \in \mathcal{P}} R_p(y_p) + \sum_{\{\mathrm{p,q}\} \in \mathcal{N}} B_{p,q} \cdot \delta_{y_p \neq y_q} \tag{5.6}$$

where λ represents the relative importance of the regional term R_p versus the boundary term $B_{p,q}$. The regional term defines the cost for assigning the pixel to a given label, while the boundary term represents the cost associated with image discontinuities.

The presence of both the data and the smoothness terms on the same lattice can be represented as two different MRFs (one for pixel values and one for edge values) coupled to each other via a conditional probability. In literature, this model is classified as a *coupled MRF* (CMRF) [127].

5.2.3. The graph cut algorithm

Boykov and Jolly [11] have introduced a segmentation technique that treats the image lattice as a graph and have provided a new min-cut/max-flow algorithm to segment (or cut) this graph, effectively solving the problem of minimizing the posterior energy defined in equation (5.6). The algorithm is also known as *α-expansion* and has laid out the foundation for a new class of efficient binary segmentation methods.

Each site (pixel) in the set $\mathcal{P}$ represents a vertex in the graph representation of the image. The graph edges, or *n-links*, are established between 4 (or higher)-connected neighbor pixels. Their weights represent the pair wise smoothing term in the energy functional and are defined by the cost function $B_{p,q}$. In the context of binary labeling, two terminal nodes are added, namely *Source* and *Sink*. These nodes are linked to each pixel in the image by weighted edges, called *t-links*. The relation between each terminal node and each pixel represents the unary data term R_p in the energy functional, describing how pixels fit the foreground and respectively the background color distributions. The concept is illustrated in Figure 5-2.

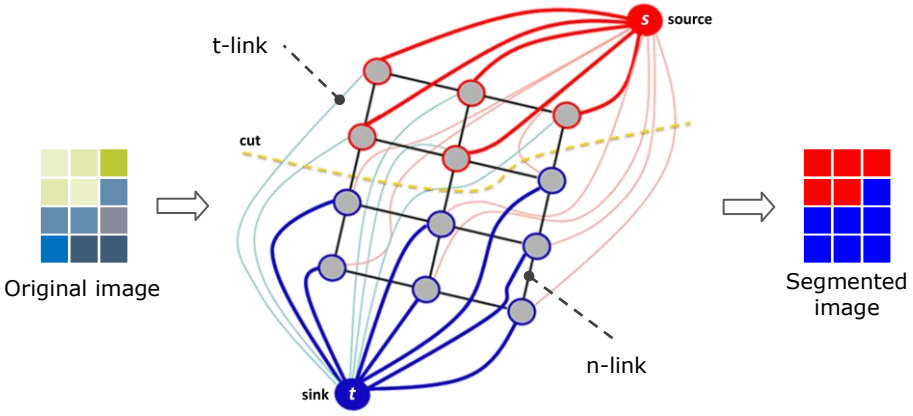


Figure 5-2. The graph representation of the image together with a graph cut

A *cut* is represented by a subset of edges (n-links and t-links) whose removal causes a complete separation of the terminal nodes. The cost of a graph cut is given by the sum of the weights of the severed edges. The severed n-links are located at the segmentation boundary, while severed t-links associate segments with a certain label. The key concept is that a minimum cost cut (min-cut) has a high probability of producing a desirable segmentation that balances boundary versus regional properties [128].

Computing the minimum cut can raise performance issues, since straightforward implementations like the max-flow algorithm [129] or the push-relabel technique [130] can be too slow for real-time applications. Boykov and Kolmogorov [53] have performed an assessment of min-cut / max-flow algorithms, focusing on achieving a higher performance. The improved max-flow algorithm proposed in their study has proven to be significantly faster than standard approaches, opening the way to real-time minimum cut computation on 2D and 3D images.

The complete description of the enhanced max-flow/min-cut algorithm developed by Boykov and Kolmogorov exceeds the scope of our present thesis. For the astute reader, the author recommends the personal web page of Professor Yuri Boykov [http://www.csd.uwo.ca/~yuri/] which provides access to all of his publications and a vast information on the subject of graph cuts. We focus next on the formulation for the t-link and n-link costs, as well as the concept of hard constraints imposed onto the graph-cut algorithm.

5.2.3.1. The smoothness term n-link

The n-link is used to model the cohesion between neighboring pixels, with the purpose of ensuring that labeling varies smoothly inside objects but change at their boundaries [131].

The $B_{p,q}$ term in the prior energy formula (5.4) exhibits large values for similar pixels and values close to zero if the pixels are very different, which signals the presence of a region boundary. Its value is expressed as:

$$B_{p,q} = e^{-\frac{||I_p - I_q||^2}{2\sigma^2}} \cdot \frac{1}{\|p,q\|} \tag{5.7}$$

where $||.||$ represents the vector Euclidian norm, I_p denotes the vector color information for pixel p across all image channels, and σ is a fixed value that estimates the video capture noise [128].

5.2.3.2. The data term t-link

The data term t-link describes regional properties of segments in the image. These regional properties can be expressed in the form of histograms [128] or GMMs [119].

In order to adhere to the real-time requirements of a videoconferencing system, an intensity histogram can be used to store the pixels' probability density of color information found in the image. Although less adapted in case of color images than a GMM, this approach improves performance and running time requirements. To reduce the impact of color variety and to improve the t-link formulation found in [128] in respect to handling color images, a regularization term can be introduced as the mean color of each label. Thus, the formulae for establishing the t-link costs become:

$$\begin{aligned} R_p(FG) &= -\frac{\ln P(I_p|FG)}{P(I_{meanFG}|FG)} \\ R_p(BG) &= -\frac{\ln P(I_p|BG)}{P(I_{meanBG}|BG)} \end{aligned} \tag{5.8}$$

where I_p is the color vector of pixel p, I_{mean} is the mean color vector of all pixels associated with a given label and $P(\cdot)$ is a probability that reflects how the color of pixel p fits the known color models associated with the labels.

5.2.3.3. Hard constraints

In simple synthetic images, the regional properties of the objects can differ enough from the background to allow for an unsupervised graph cut segmentation. Unfortunately, this does not always apply to real-world examples, where differences between FG and BG can be much more subtle. In such case, a human operator is required to intervene and direct the segmentation by offering *a priori* knowledge to the algorithm about pixels that are definitely FG or BG (as seen in Figure 5-1).

In the scenario of interactive segmentation described above, the user provides two sets of seeds on the image, $\mathcal{O}$ and $\mathcal{B}$ ($\mathcal{O} \cap \mathcal{B} = \emptyset$), that correspond to FG and respectively BG segments. These sets act as hard constraints among all possible segmentations, linking with infinite-cost t-links any pixel $p \in \mathcal{O}$ to the Source and any pixel $q \in \mathcal{B}$ to the Sink. The color information of the seeds can also be used to learn the histogram and mean color vector for the FG and BG [128], allowing to compute the probabilities used in formula (5.8).

Figure 5-3 provides an update of Figure 5-2 by introducing hard constraints.

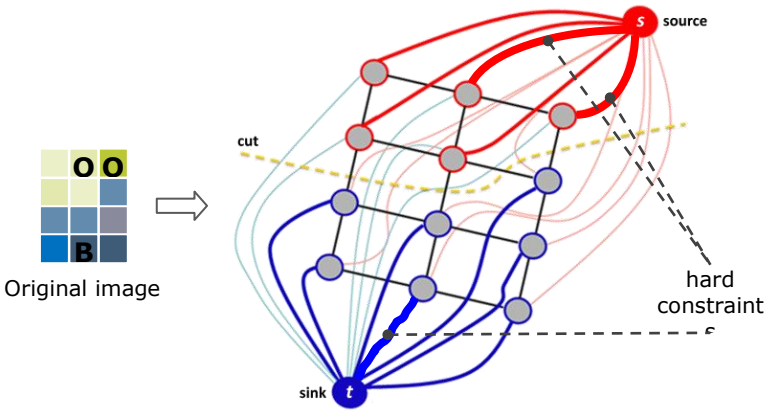


Figure 5-3. Graph cut segmentation with hard constraints

5.2.4. Applicability to our foreground extraction system

In respect to the results produced by the S-T motion-based FG extraction method described in chapters 3 and 4, the balancing of boundary versus regional properties offered by graph cut can provide the means to:

a) restore the FG regions that proved to be a challenge for the OF estimation and/or the TSM persistence algorithm (as shown in Figure 4-4, sequence *Mihai 3*);
b) achieve the pixel-accurate segmentation required in videoconferencing applications, by segmenting along the most probable boundaries between FG and BG.

Point a) is covered by the regional term in the CMRF energy functional described in formula (5.6), while point b) relies on the boundary term of this functional.

Considering the good results reported in the literature regarding the use of hard constraints for interactive segmentation, we propose an heuristic approach in which these constraints are generated automatically, based on the information provided by the TSM and the sparse OF.

The next section describes the proposed algorithm for generating hard constraints and the integration of graph cut in our FG extraction system. We also show that this approach effectively solves the problems described at points a) and b) and how the resulting segmentation can be used to correct the TSM by removing the BG leaking effect (illustrated in Figure 4-4, sequence *Green Screen 3* and *Train*).

5.3. Unsupervised graph cut segmentation using constraints generated from TSM and optic flow data

5.3.1. High-level description of the algorithm

The second part of this chapter describes the algorithm responsible of regenerating the pixel-accurate FG image starting from the coarse segmentation result represented by the TSM, in combination with additional information provided

by the sparse OF. Unlike the TSM approach, which works with image blocks (superpixels) of 4x4 pixels, the pixel-accurate segmentation labels every pixel in the input frame I_t, using the predefined label set $\Omega = \{BG, FG\}$.

From an architectural point of view, this step requires new processing blocks to be added to our FG extraction system, as illustrated in Figure 5-4.

The key to the unsupervised graph cut segmentation lies in the hard constraints generator block, which implements the heuristic algorithm for generating the sets $\mathcal{O}$ and $\mathcal{B}$ described in section 5.2.3.3. As we are about to discover next, seeds in set $\mathcal{O}$ come from Delaunay triangulation applied on a set $\mathcal{D}$ of sparse OF features that is maintained and updated in parallel with the TSM. Hard background constraints in set $\mathcal{B}$ are created using an algorithm inspired by the drape approach from [15]. At every moment t, the hard constraints are regenerated based on new evidence provided by TSM_t and $\mathcal{D}_t$ and the graph-cut algorithm from [53] is applied to produce the final segmentation of I_t.

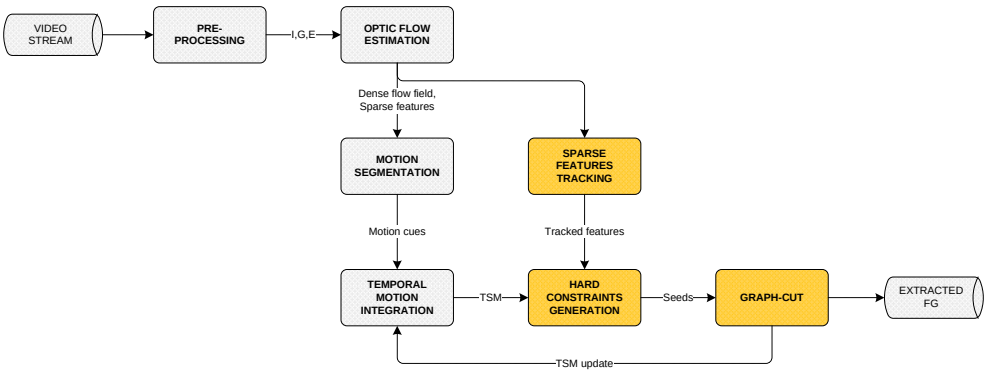


Figure 5-4. The updated block-level diagram of the FG extraction system.

5.3.2. Hard constraints generation algorithm

5.3.2.1. Foreground constraints

At a first glance, the TSM should offer sufficient information for generating the hard foreground constraints, for example by placing them along its medial axis. However, as shown in Figure 4-4, sequence *Green Screen 3*, it is possible to have BG regions leaking into the TSM. If FG seeds are placed inside such regions, the final segmentation may mislabel significant parts of the actual BG as FG. For this reason, the process of generating FG constraints starts by finding those locations in the TSM that involve a high degree of certitude in respect to their FG labeling.

5.3.2.1.1. Tracking of sparse OF features

In our previous researches [71] the subset $\varphi \subset \Phi$ of sparse OF features has been used as a confident source of control points on moving image objects. If we can track the features exposed by φ at a given moment t across a larger time window and as part of a bigger set $\mathcal{D}_t$, it is possible to obtain a series of control points that always follow the motion of exposed FG regions (see Figure 5-5).

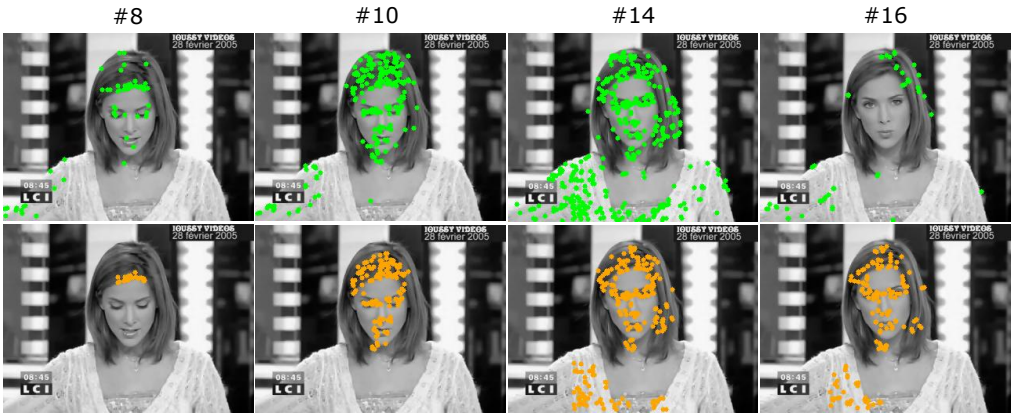


Figure 5-5. Tracking of sparse OF features on the TV show sequence.
1st row - sparse OF features, 2nd row - set $\mathcal{D}$ of tracked features

We built the set $\mathcal{D}_t$ by applying the following simple algorithm:

1. all features in φ are added as part of the new set $\mathcal{D}_t$;
2. all features in $\Phi - \varphi$ that are located within a $r_f = 3$ pixel radius from a feature in $\mathcal{D}_{t-1}$ are also added to $\mathcal{D}_t$ in order to account for features that belong on previously moving foreground regions, now stationary;
3. the features in the newly-formed set $\mathcal{D}_t$ that are not labeled as FG in TSM_t are discarded in order to avoid tracking features that may actually reside in the BG.

Formally, set $\mathcal{D}$ can be expressed as:

$$D_t = \varphi \cup \{f_i \in \Phi - \varphi | \exists f_j \in D_{t-1}, |f_i - f_j| < r_f\} - \{f_i \in \Phi | TSM_t(f_i) = BG\} \quad (5.9)$$

where $|\cdot|$ denotes the Euclidian norm. At the initial moment we consider $D_{t=0} = \emptyset$.

5.3.2.1.2. Triangulation of tracked features

Features in $\mathcal{D}_t$ are good candidates for FG hard constraints due to their origin as observed control points on moving image regions and their belonging to TSM_t.

In order to complete the set $\mathcal{O}$ of FG image seeds and ensure that as much color information from the FG is captured by the hard constraints, we apply a Delaunay triangulation [24] on set $\mathcal{D}_t$. As the reunion of all 2-simplices produced by the triangulation gives the convex hull of set $\mathcal{D}_t$, there may be facets that exceed the boundaries of TSM_t. By eliminating the 2-simplices that do not completely overlap with the TSM, we obtain a subset of triangulation facets. The edges of each facet in this subset are used to build a wireframe inside TSM_t. For every pixel on the wireframe, a seed of radius r_s is added in the set $\mathcal{O}$, where r_s has been arbitrarily set to 2 pixels. In order to avoid overextending the FG into the BG, seeds that are located too close to the TSM boundary are eliminated from set $\mathcal{O}$. This is achieved by constraining them within a mask $\widetilde{TSM}_t$ obtained as the erosion of TSM_t.

A graphical illustration of the process of generating hard foreground constraints is given in Figure 5-6 using a synthetic image from the *Avatar* sequence for better clarity.

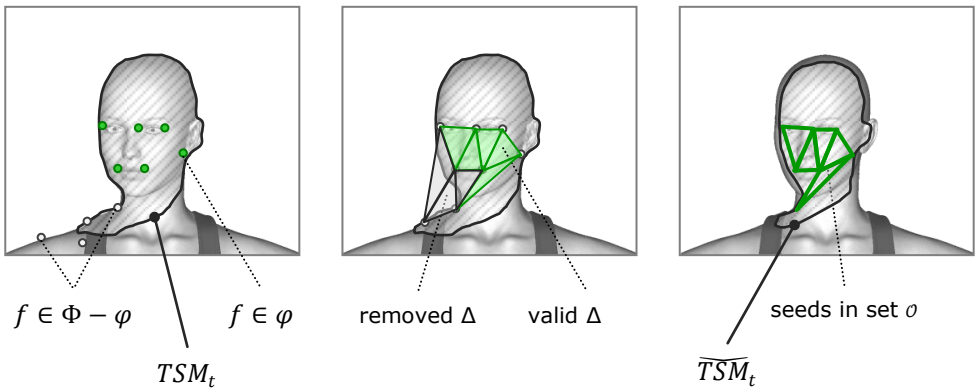


Figure 5-6. Graphical representation of the process of generating hard FG constraints

5.3.2.2. Background constraints

The background seeds are generated around the FG exposed by TSM_t, by maintaining a guard interval of $2 * r_s$, where r_s denotes the seed radius. The guard interval is required around the proximity of the TSM in order to avoid crossing into possible FG space. The described approach employs 3 types of BG constraints, each specialized on a particular aspect related to the separation between FG and BG.

5.3.2.2.1. Boundary constraints

This type of BG seeds is generated by using a drape model [15] that descends pixel by pixel from the top row of the current frame, until reaching the proximity of TSM_t or the bottom row of the frame.

The proximity mask $\widehat{TSM_t}$ is obtained by successively dilating the TSM over a number of iterations $i = 2 * r_s$, where r_s denotes the seed radius. By leaving enough space between the BG and FG seeds guarding the TSM border, the algorithm ensures that the drape does not get too close to the FG to adversely affect the accurate boundary extraction.

At each point where the drape meets $\widehat{TSM_t}$, a new BG seed of radius r_s is added to the set $\mathcal{B}$, as shown in Figure 5-7 (a). No seeds are added for drape pixels that reach the bottom row of the frame, in order to leave room for FG expansion, which is required for cases involving smooth regions (like the one shown in Figure 4-4, sequence Mihai 3 in respect to the subject's clothing).

5.3.2.2.2. BG clutter constraints

In order to account for cluttered backgrounds or BG patches with similar color distributions as the FG, every j columns a complete line of smaller seeds with radiuses of $r_s/2$ is added to $\mathcal{B}$, connecting the top row of the frame with the boundary of $\widehat{TSM_t}$ (see Figure 5-7 (b)). This increases the chances to capture as much color information from the BG as possible. Throughout our experiments we have arbitrarily set distance j to $3 * r_s$ pixels.

As in case of boundary BG constraints, these seeds are added only if the column intersects the proximity of TSM_t. This ensures that enough space is left for possible FG expansions, which in case of a videoconferencing application are likely to occur in the lower half of the image, where the subject's torso is located.

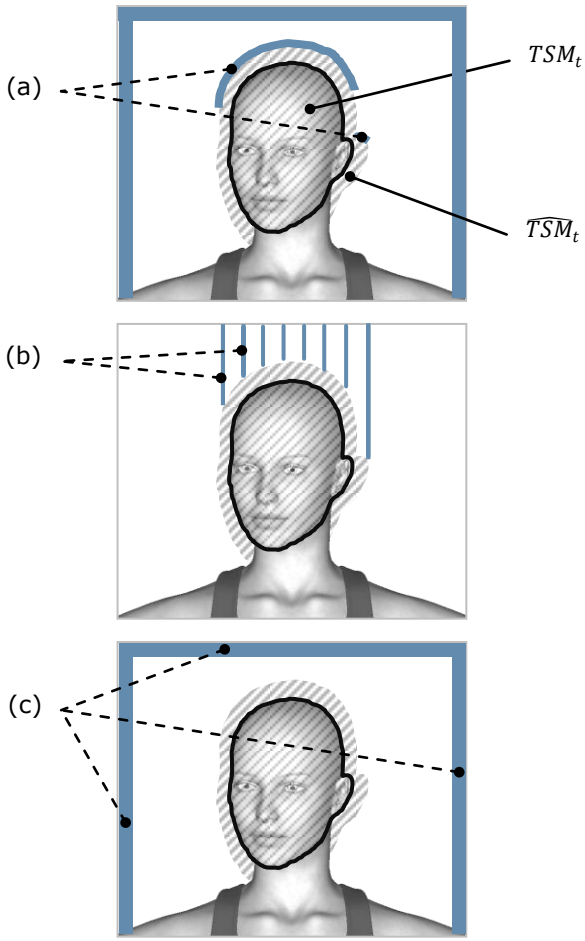


Figure 5-7. Automatic generation of BG constraints

5.3.2.2.3. Scene-related constraints

For videoconferencing applications, we exploit the fact that the FG is mostly located in the middle of the screen. In Figure 5-7 (c) the top row, the leftmost and the rightmost columns are by default bordered with BG seeds as long as those seeds do not reach $\widehat{TSM_t}$. The reason behind this approach is to constrain an eventually uncontrolled FG expansion due to similar BG/FG color distributions and to account for the initial frames in the sequence, when the TSM provides too little information about the FG.

5.3.3. Pixel-accurate segmentation

Applying the graph cut algorithm based on the hard constraints from the sets $\mathcal{O}$ and $\mathcal{B}$ will produce the most probable labeling $\hat{y}$ mentioned in paragraph 5.2.2. The FG is defined as the set of pixels in I_t that remain connected to the Source terminal, while the BG is represented by those pixels that keep their t-links with the Sink one.

Figure 5-8. Final foreground extraction results. From left to right: the corrected TSM, the set of tracked sparse OF features, the hard constraints imposed on graph-cut segmentation, and the pixel-accurate FG segmentation.

The FG connected components in $\hat{y}$ may occasionally exhibit small irregularities around their borders. In order to improve their appearance even more, the same active contour approach used in [71] is applied to increase edge smoothness. The only difference lies in the values used for the continuity, curvature and respectively edge attraction terms in the snake energy functional. We want the contour to follow closely the nearby edges without distancing too much from the original object boundary given by $\hat{y}$. Thus, we have increased stiffness, allowed for a certain amount of edge attraction and placed a negative weight on elasticity, obtaining the following empiricaly-adjusted values for the snake weighting parameters: $\alpha = -1.1$, $\beta = 50$ and $\gamma = 1.5$. The resulting contours define the final FG regions, united as part of the **pixel-accurate FG mask** $\mathcal{M}_t$ and illustrated in Figure 5-8.

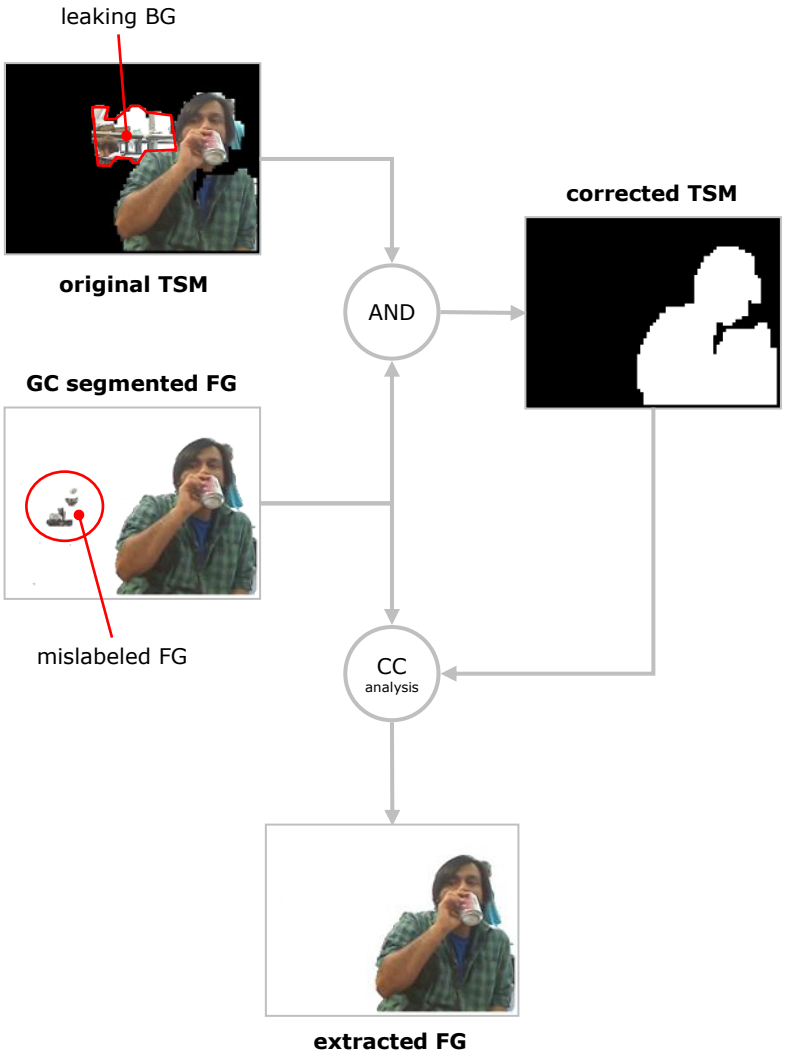


Figure 5-9. Synchronization between TSM and graph-cut segmentation

5.3.4. Temporally stable mask synchronization

As the last step in the presented FG extraction method, the information in TSM_t is synchronized with the result produced by the GC segmentation.

First, TSM_t is corrected by removing those foreground pixels that are not labeled as FG in $\mathcal{M}_t$:

$$TSM_t^* = TSM_t \wedge \mathcal{M}_t \tag{5.10}$$

This apparently simple step has a significant impact on segmentation quality, as it helps removing leaking background regions so that they are not propagated into TSM_{t+1}.

Figure 5-9 shows the difference made by the TSM correction on frame #299 of sequence *Mukul 10*. The leaking BG region found in the original TSM formulation [132] is completely eliminated after applying formula (5.10).

Second, it is possible for $\mathcal{M}_t$ to contain small, isolated blobs which are mislabeled as FG, as shown in Figure 5-9. These artifacts are caused by small BG patches with color distributions very similar to those of the FG, which are not covered by any BG seeds that would keep them connected to the Sink terminal. In order to eliminate them, the algorithm performs a connected-component analysis of $\mathcal{M}_t$ and keeps only the components that intersect with TSM_t^*.

$$\mathcal{M}_t^* = \bigcup CC_i \,|(CC_i \in \mathcal{M}_t) \wedge (CC_i \cap TSM_t^* \neq \emptyset) \tag{5.11}$$

Formula (5.11) keeps the regions that resulted from GC foreground expansion as part of the final segmentation and discards the artifacts that are not connected with TSM and originate due to mislabeled BG regions.

5.3.5. Algorithm pseudocode

After describing all stages of the unsupervised GC algorithm, it is now possible to present the complete approach in pseudocode:

```
var 𝒟;     // set of tracked sparse OF features, according to eq. (5.9)
var 𝒪;     // set of FG seeds (constraints)
var ℬ;     // set of BG seeds

procedure AddSeed(seeds, x, y, r) {
      // add a seed with radius r and coordonates (x, y) to the set of GC constraints
      for (i in [-r .. r]) {
            for (j in [-r .. r]) {
                  seeds += (x+i, y+j);
            }
      }
}

procedure TrackSparseFeatures(TSM, Φ, φ, rf) {
      // add all control features located on moving regions
      𝒟′ = φ;

      // add all previous moving features, now stationary
      foreach (fi in Φ − φ)
            foreach (fj in D)
                  if (|fi − fj| < rf) { 𝒟′ += fi; }

      // remove all features marked as BG in the TSM
      foreach (fi in Φ) if (TSM(fi) == BG) { 𝒟′ -= fi; }
```

```
      // store result
      𝒟 = 𝒟′;
}

procedure GenerateFGSeeds(TSM, 𝒟, r_s) {
      𝒪 = ∅;
      TSM' = erode(TSM, r_s);

      foreach (facet in Delaunay(𝒟)) {
            // filter out 2-simplices that are not completely covered by the TSM
            if (TSM' ∩ facet ≠ facet) continue;

            // mark FG seeds
            foreach (edge in facet)
                  foreach (p in edge)
                        AddSeed(𝒪, p.x, p.y, r_s);
      }
}

procedure GenerateBGSeeds(TSM, r_s) {
      ℬ = ∅;
      TSM' = dilate(TSM, 2r_s);

      // boundary + BG clutter constraints
      for (x in [r_s .. TSM.width – r_s])
            for (y in [r_s .. TSM.height – r_s])
                  if (TSM'(x, y) ≠ 0) {
                        // drape reached top oF dilated TSM, add boundary seed
                        AddSeed(ℬ, x, y, r_s);

                        if (x % 3r_s == 0) {
                              // add BG clutter seeds
                              for (k in [r_s .. y])
                                    AddSeed(ℬ, x, k, r_s/2);
                        }
                  }

      // scene-related constraints
      for (x in {r_s, TSM.width – r_s})
            for (y in [r_s .. TSM.height – r_s]) {
                  if (TSM'(x, y) ≠ 0) break;
                  AddSeed(ℬ, x, y, r_s);
            }
      for (x in [r_s .. TSM.width – r_s])
            if (TSM'(x, y) == 0) {
                  AddSeed(ℬ, x, y, r_s);
            }
}

procedure UnsupervisedGC(I_t, TSM_t, Φ, φ) {
      // empyrical values
      r_f = 3;
      r_s = 4;

      // update set of tracked sparse OF features
      TrackSparseFeatures(TSM_t, Φ, φ, r_f);

      // generate GC constraints
```

```
        GenerateFGSeeds(TSMt, 𝒟, rs);
        GenerateBGSeeds(TSMt, rs)

        // perform the pixel-accurate segmentation
        ŷ = GraphCut(It, 𝒪, ℬ); // apply the [53] algorithm

        // smoothen the result
        foreach (CC in ConnectedComponents(ŷ)) {
             ℳt += BodyOf(Snake(CC, α, β, γ));
        }

        // synchronize the TSM and the GC segmentation
        TSMt = TSMt and ℳt;      // eq. (5.10)
        foreach (CC in ConnectedComponents(ℳt))
             if (CC ∩ TSMt == ∅)
                   ℳt = ℳt − CC;       // eq. (5.11)
}
```

5.4. Summary

The approach to unsupervised GC introduced in this chapter represents the final stage in the proposed FG extraction method. A new heuristic algorithm has been introduced, which relies on the intermediate results obtained in the motion detection and S-T integration stages in order to automatically generate the hard constraints required by the unsupervised GC segmentation.

Experimental results (illustrated in Figure 5-8 for a subset of video sequences) have shown that the proposed approach is capable of generating an accurate FG extraction, based on the coarse foreground representation provided by the TSM. These results indicate that the addition of graph cut to the video frame processing workflow addresses several important issues reported in chapter 4.5, as follows:

- the ability to restore the smooth and non-reflecting FG areas which proved to be a challenge for the TSM algorithm;
- the capability to remove background leaking caused by cumulated TSM errors;
- the ability to restore the missing FG regions caused by occlusions.

In order to ensure that the reported results are objective and accurate, the final segmentation quality requires an evaluation based on a method-independent perceptual objective metric. In addition, it is important to ensure that the complete FG extraction system implementing the methods described in chapters 3-5 has the potential to match the real-time execution requirements of videoconferencing applications. Hence, we dedicate the following chapter to these two important aspects.

6. RESULTS AND DISCUSSION

6.1. Implementation details and performance considerations

6.1.1. Implementation details

The FG extraction method presented in this thesis has been implemented using the *C# 4.0* programming language on top of the *Microsoft .NET 4.5* application framework. Additional image processing support has been provided by the *Emgu CV 2.2.1* .NET wrappers [www.emgu.com] to the *Intel OpenCV* image processing library [24].

In particular, the underlying *OpenCV* library plays an important role as it provides, besides basic image and video stream manipulation primitives, reliable off-the-shelf implementations of several algorithms and methods employed by our approach:

- Canny edge detection [133];
- bilateral filtering [99];
- Farnebäck's dense OF estimation [80];
- Shi and Tomasi feature detector [65];
- pyramidal Lukas-Kanade sparse OF estimation [79];
- distance transform [103];
- Greedy snake [106].

In addition, the max-flow/min-cut algorithm used by the GC segmentation has been translated into *C#* and *Emgu CV* method calls based on the reference C/C++ implementation provided courtesy of Y. Boykov and V. Kolmogorov [http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html].

Development and testing of the resulting FG extraction system were carried using the *Microsoft Visual Studio 2010* environment running on *Windows XP SP3* and *Windows 7 Professional* operating systems. No GPU acceleration was used as we focused instead on exploiting the multiprocessor architectures of the host machine, leveraging the features provided by the *Parallel Extensions* of .NET Framework.

Input sequences were handled at a resolution of 320 x240 pixels, which is standard in most videoconference systems. Given the performance considerations mentioned in section 5.2.3 and further accentuated by the managed language implementation, the GC algorithm has been applied on a downscaled clone of the input frame, at 160 x 120 pixels. The same approach was taken for the dense OF estimation, where pixel-accuracy is not a major aspect.

The choice for a managed programming language like C# was not coincidental. Beside the advantage of automatic memory and object lifecycle management ensured by the presence of a garbage collector, the language and the framework supporting it have enabled us to focus on the algorithm rather than the implementation details due to its integration of advanced features such as lambda expressions, anonymous functions, delegates and language integrated query (LINQ). Furthermore, cross-platform compatibility was also attained, as the code was compiled and run on the *OS X 10.8* operating system using the open-source

Mono .NET 2.10 framework without changes. On the downside, managed languages are known to perform slightly slower than their unmanaged counterparts like C/C++. Considering that the highest priority was given to the qualitative aspects of the algorithm rather than its brute performance, this drawback has been deemed acceptable.

Finally, in order to inspect the intermediate results at various stages and to fine tune the control parameters of each stage we have build a graphical user interface frontend on top of our FG extraction system, illustrated in Figure 6-1.

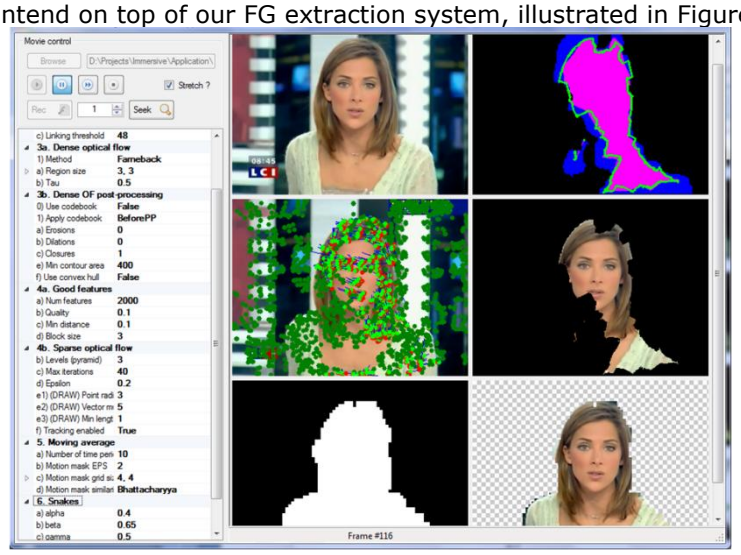


Figure 6-1. GUI frontend for algorithm visualization and tuning

6.1.2. Algorithm parallelization

Besides segmentation quality, another important requirement imposed on a FG extraction method is the capability of real-time (RT) performance. A balance must be achieved between quality and performance in order to satisfy immersive conferencing requirements, as quality alone is not sufficient to qualify the algorithm for deployment in a videoconference system.

In the present thesis the emphasis was put on the quality of the FG / BG segmentation. Nevertheless, the algorithms and techniques employed by the presented FG extraction system have been chosen with performance considerations in mind, in order to preserve its RT execution potential.

In order to prove the RT capabilities of the introduced FG extraction method, its implementation has been designed to support parallel execution by exploiting the multi-core architecture of the host machine CPUs. Parallelization options have been considered with two levels of granularity: global and local, both of them supported by the *Parallel Extensions* of the .NET Framework.

6.1.2.1. Global parallelization options

Figure 6-2 illustrates the parallel execution of tasks as implemented by the proposed FG segmentation system.

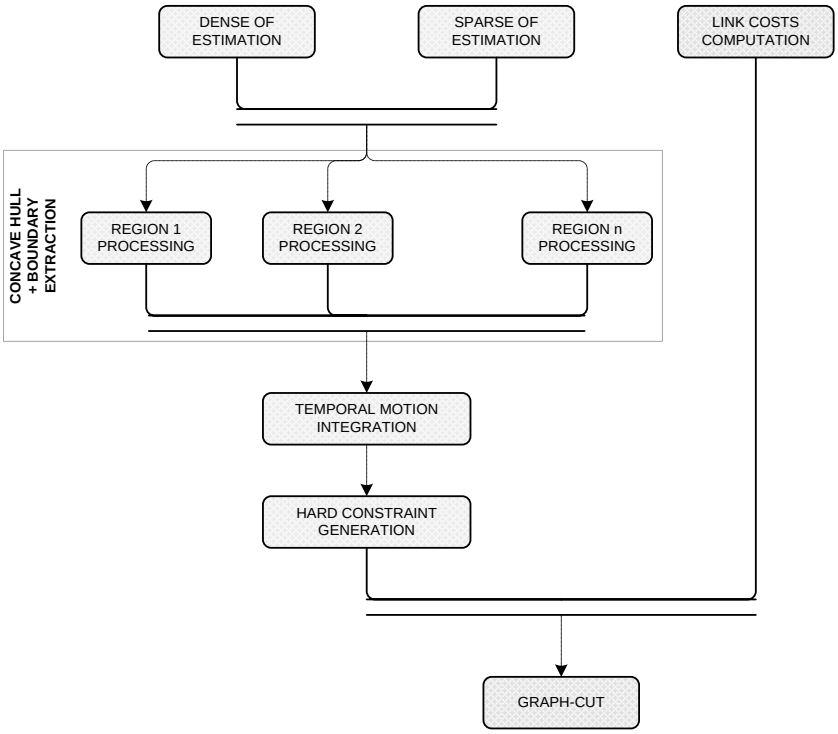


Figure 6-2. Parallelization of the complete FG extraction algorithm

In line with Figure 3-6 and following the description in [71], the motion detection parts of the FG extraction system are designed as parallel tasks with potential of running simultaneously on separate CPU cores while processing the current video frame. These tasks include:

a) the dense OF estimation;
b) the sparse OF estimation;
c) the concave hull computation and active contour extraction for each moving image region.

Tasks a) and b) are among the most computationally intensive elements in the processing pipeline and have comparable execution times. The ability to assign them to different CPU cores and compute the OF estimations in parallel provides a significant performance boost, as the execution time is reduced to half when compared to a sequential approach.

The addition of graph-cut in the frame processing pipeline may only be done after the computation of TSM data, as this information is required to build the hard constraints (see Figure 5-4). As a consequence, the max-flow/min-cut algorithm can only be executed at the end of the TSM computation, which increases frame processing time with a significant amount. However, the cost computation of n- and t-links is independent from the motion detection and persistence part and can be performed as soon as a new frame is available. Thus, a new parallel task is added in our system as:

d) cost computation of n-links and t-links for the graph-cut algorithm.

The cost associated to the CMRF site links provided by task d) becomes available by the time GC constraints generation is finished, thus saving additional time in the overall video frame processing.

6.1.2.2. Local parallelization options

The local parallelization options target the iterations employed by different stages of the FG extraction method. These iterations correspond to *foreach* statements in the pseudocode listed in chapters 4.3.1 and 5.3.5.

The processing of the elements in an iteration can be distributed among available CPU cores using the *Parallel.ForEach<>* method when the elements can be processed independently. This is applicable to the following parts in our implementation:

a) morphology part of the TSM algorithm (removal of small FG / BG connected components from the TSM);
b) processing of 2-simplices in the Delaunay triangulation (creation of FG constraints for the GC segmentation);
c) result smoothing in the pixel accurate GC segmentation (snakes applied to connected components of the resulting FG mask).

The code snippet below demonstrates the implementation in case of point a).

```
// get rid of small FG blobs
var contours = image.GetContoursList();
Parallel.ForEach<Contour<Point>>(contours, ct => {
     if (ct.Area < areaThr)
          image.Draw(ct, new Gray(bg), -1);
      else
          image.Draw(ct, new Gray(fg), -1);
});

// get rid of small BG blobs
contours = image.Not().GetContoursList();
Parallel.ForEach<Contour<Point>>(contours, ct => {
      if (ct.Area < areaThr) {
          image.Draw(ct, new Gray(fg), -1);
      }
});
```

6.1.3. Execution performance

We have measured the execution performance of the FG extraction system on different test machines, using a database of test videos recorded at a resolution of *320 x 240 pixels and 30 FPS*. These are common parameters used in videoconferencing systems, as they fit the data traffic bandwidth and video capture capabilities of both mobile devices as well as desktop computers. The same consideration applies to immersive video conferencing: the resulting virtual scene resolution can be higher even if the video streams coming from participants are at a lower resolution, as they occupy only a smaller part of the scene.

The first machine in the test setup is an older generation *Lenovo T400* laptop equipped with a dual-core *Intel Centrino P8400 CPU* and 2 GB RAM. The second, a modern desktop, has a quad-core *Intel Core i5 – 2400 CPU* and 4 GB of RAM. The third is a *MacBook Pro 13”* equipped with a dual-core *Intel Core i7 – 3520M CPU* and 4 GB of RAM running Windows 7 as a virtual machine.

Table 6-1. Algorithm execution times

CPU	Cores / Threads	CPU load	Execution time / frame [ms]	Average FPS
Intel Centrino P8400	2/2	65%	172.4	5.8
Intel Core i5 – 2400	4/4	40%	84.0	11.9
Intel Core i7 – 3520M	2/4	43%	108.7	9.2

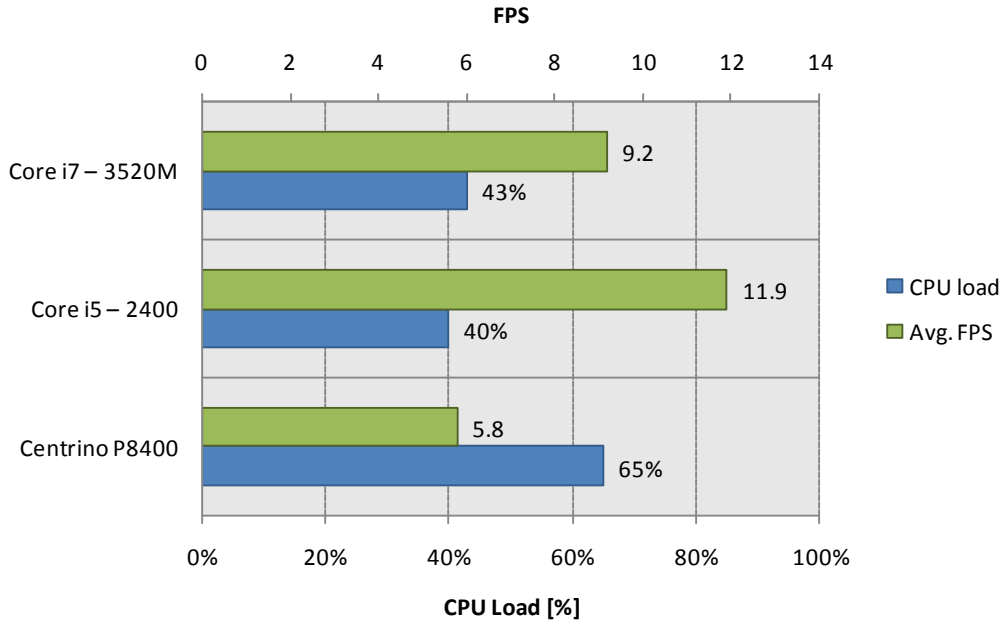


Figure 6-3. Execution performance chart

The average performance results obtained during testing are listed in Table 6-1 and plotted in Figure 6-3. The impact on execution performance caused by switching between a Gaussian-based TSM and a SSIM TSM (see chapter 4.2) has been found negligible.

As expected, the modern quad-core i5 CPU is able to exploit better the parallelization options of the proposed algorithm, achieving an average of **12 FPS**, followed by the dual-core mobile i7 whose output achieved between 9 and 10 FPS. The 4-years old Centrino CPU is the slowest performer, not being able to meet near-RT requirements. The aggregated CPU load was in inverse proportion to the number of supported CPU threads, however all machines presented sufficient margin for running other applications. It is important to note that this result is also influenced by several implementation factors, such as:

a) use of the managed C# programming language;
b) the overhead introduced by the *Parallel Extensions* context switching;
c) the use of a virtualized OS in case of the Core i7-equipped test machine.

Taking into account the above factors, the execution results prove the near-RT capabilities of the proposed FG extraction method on *current generation hardware*, considering the RT margin to be at 15 FPS. Furthermore, given that GPU implementations have already been reported in the literature for optical flow

estimators [134, 135], active contours [136] and graph-cuts [137], we can safely conclude that the algorithm has potential for significantly faster execution if implemented using video hardware acceleration.

6.2. Perceptual quality assessment of the segmentation

6.2.1. Overview

The perceived quality of FG/BG segmentation is a key point for immersive videoconferences, being the main responsible in delivering a realistic experience to the participants. Given the ill-posed nature of object segmentation [3] and the fact that perceived quality depends on subjective factors, building a *reliable and objective perceptual metric* to evaluate segmentation results is a very challenging task.

Previous researches have focused on standardizing the way segmentation results are compared between different algorithms. The most popular metrics are the *MPEG error measure* proposed by the ISO/MPEG-4 standard [138] and the derived *weighted quality metric* introduced in [139], due to their simplicity and easy implementation. More recently, Gelasca-Drelie and Ebrahimi [9] have proposed an improved perceptual objective metric that relies on formal psychophysical experiments in order to better cover the subjective component of the evaluation. According to the results presented by the authors, this new approach outperforms the MPEG error and the weighted quality metrics and furthermore, it allows parameter tuning for different types of applications in order to address the ill-posed characteristic of FG segmentation.

6.2.2. The perceptual objective evaluation metric

The segmentation results produced by our FG extraction algorithm have been evaluated using the state-of-the-art metric proposed by Gelasca and Ebrahimi [9]. This method relies on ground truth information in order to identify segmentation artifacts in the extracted FG. Next, the artifacts are classified based on their annoyance factor and weighted depending on the current application context: video compression, video surveillance or mixed reality.

Immersive videoconferencing can be considered a classical candidate for the *mixed reality* group of applications. In this context, the evaluation metric identifies the following possible segmentation artifacts (first introduced in Figure 1-3, shown in Figure 6-4 for real cases and listed below in descending order of their annoyance to a human observer):

- *flickering*, which occurs due to an abrupt change in artifact size between consecutive video frames;
- *inside holes* (HI) – regions mislabeled as BG located completely inside FG objects;
- *border holes* (HB) – regions mislabeled as BG on the border of FG objects;
- *added regions* (AR) – leaking BG regions not connected to FG objects;
- *added background* (AB) – leaking BG regions along the border of FG objects.

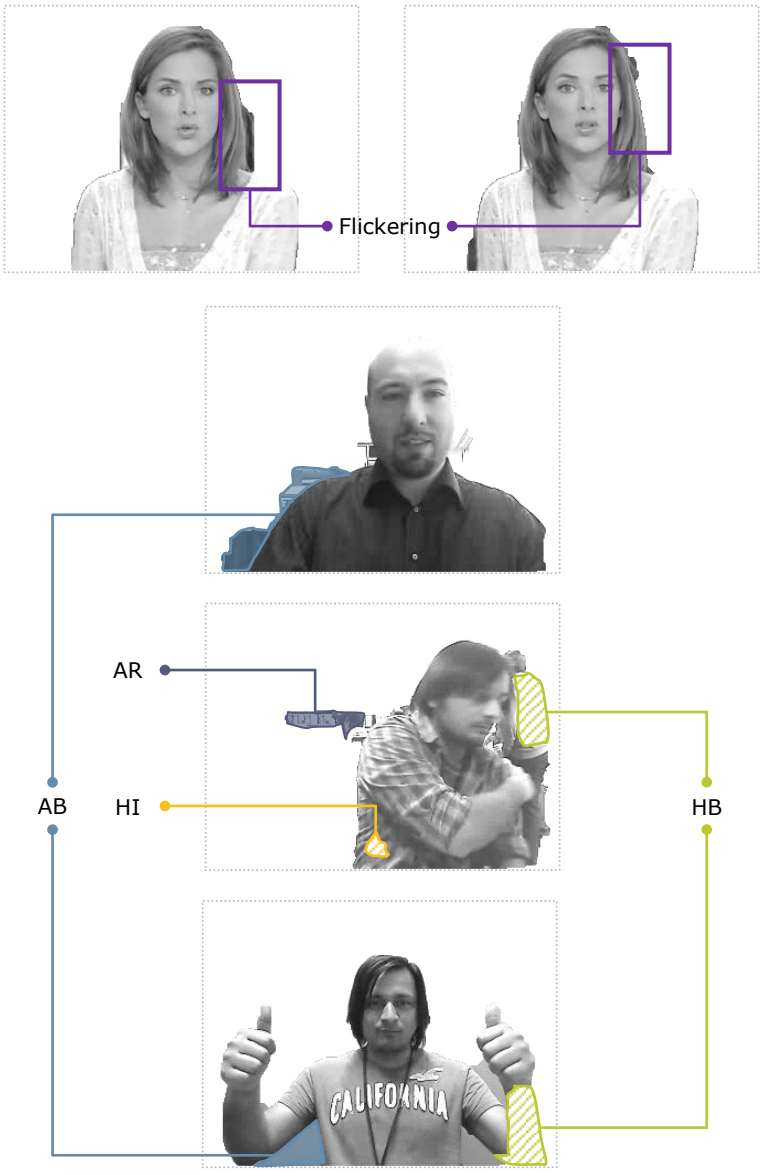


Figure 6-4. Segmentation artifacts identified during quality evaluation

Each artifact λ in the set $\Lambda = \{HI, HB, AR, AB\}$ has an associated perceptual artifact metric, denoted with PST_λ. The PST encodes the *mean annoyance value* (MAV) associated with the artifact, and takes into account the S-T characteristics of the artifact and the human memory and expectation effect over a sequence of video frames. Flickering is not modeled as a separate artifact as it is included as a temporal component into PST_λ. The detailed formulae and methods for computing the MAV for each type of artifact can be found in [9].

The complete MAV for each segmented video frame is given by the formula:

$$PST = \sum_{\lambda \in \Lambda} \omega_\lambda \cdot PST_\lambda$$

where ω_λ is a set of application-dependent weighting factors that control the relative contribution of each artifact to the overall perceived quality. Table 6-2 lists the weights proposed by Gelasca and Ebrahimi for their optimized PST metric in the context of mixed-reality applications.

Table 6-2. Weighting factors for segmentation artifacts in a mixed reality scenario

Artifact type	Weight	Value
Inside holes	ω_{HI}	12.57
Border holes	ω_{HB}	8.74
Added regions	ω_{AR}	8.31
Added background	ω_{AB}	6.71

The PST is computed over 5 second video sequences in order to prevent subjects from getting used to a certain segmentation quality and to account for the human memory and expectation effects.

6.2.3. Evaluation of the FG extraction algorithm

We have implemented an unsupervised approach to compute the perceptual objective metric (PST) and evaluate the segmentation produced by our FG extraction system.

The unsupervised evaluation has been performed on a database of test video sequences recorded using a chroma-keyed background. The assessment process, illustrated in Figure 6-5, performs the following steps:

1. Obtain the ground truth FG segmentation by removing the color keyed component. The simplest way to achieve this is by performing color substitution in the 32-bit ARGB color space. For example, in case of a video recorded using green chroma-key BG, the alpha channel of a pixel is set to 0 (transparent) if the RGB color information matches the following criteria:
   $$(G - R > \tau_R) \wedge (G - B > \tau_B)$$
   where τ_R and τ_B are thresholds that depend on the actual BG characteristics (in our particular case we determined $\tau_R = \tau_B = 15$ empyrically).
2. Perform BG substitution by overlaying the ground truth on top of a background video sequence. The use of a BG video instead of a static image ensures that lighting changes and BG motions are encountered during the segmentation process.
3. Subject the BG-substituted sequence to FG extraction using our proposed algorithm, and store the segmentation result.
4. Compare the extracted FG frame-by-frame with the ground truth obtained in step 1 in order to identify segmentation artifacts and compute their contribution to the perceptual objective metric.

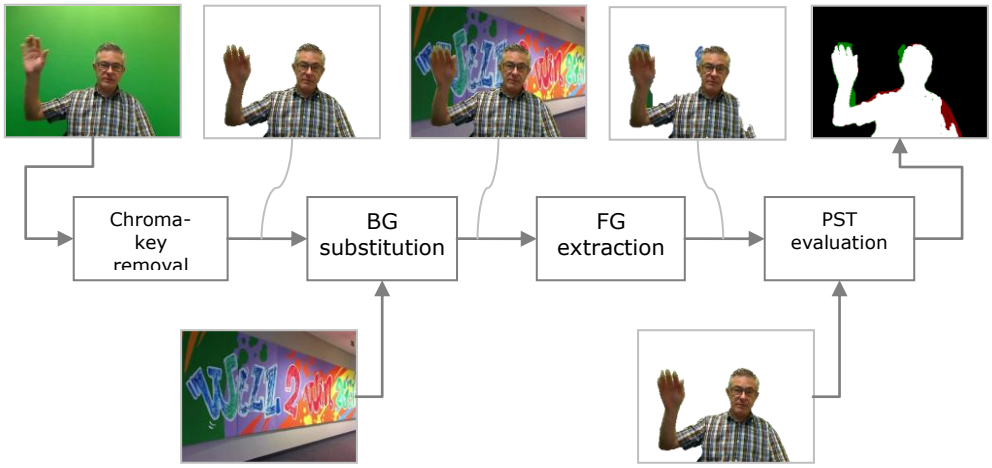


Figure 6-5. The unsupervised quality assessment process

A total of 10 background video sequences recorded in various environments and lighting conditions have been used in conjunction with 30 chroma key videos, for a total of 300 assessments. Since the TSM part of our FG extraction method comes in two flavors – Gaussian and SSIM-based – the assessments were performed independently for each of them, in order to provide the means for a comparison between the two alternatives.

6.2.4. Assessment results and discussion

The evaluation results have been summarized in Figure 6-6, using box plots individually for each type of artifact (PST_λ) and globally at PST level.

From the summarized results it can be observed that inner holes (HI), the second most annoying artifact after flickering, present a low occurrence. The same can be said about added regions (AR), which are reduced to a minimum, also due to the TSM synchronization step described in paragraph 5.3.4. This behavior comes from the way hard constraints are generated for the graph-cut algorithm, as a clear separation between FG and BG is achieved while capturing as much region color information as possible. The higher value for the border holes (HB) metric is partly due to lack of sufficient motion at the beginning of the video sequences under test. The ground truth is immediately available through chroma-key removal, but FG extraction requires a sufficient number of motion cues to reach the same level of completeness. During this phase, HB artifacts are likely to occur as false negatives, sometimes in conjunction with the flickering effect which raises the value for PST_{HB}.

The perceived quality results produced by the Gaussian-based TSM and the SSIM-based TSM are very similar. By referring to the side-by-side results in Table 6-3, we notice a marginal improvement on the Gaussian-based TSM, which produces an average MAV of **20.45** compared to the **20.83** obtained for the SSIM TSM. The Gaussian-based S-T segmentation is slightly better at handling AB and HI artifacts, while the SSIM one deals better with AR and HB artifacts. Std. deviation results show that the SSIM-based TSM behaved more consistent than the Gaussian one, which produced results spread across a slightly larger domain. From a

qualitative point of view the differences between the two TSM flavors are likely to go unnoticed by a human observer, however the quantitative analysis of the perceptual quality assessment results puts the Gaussian-based TSM ahead of the SSIM-based one.

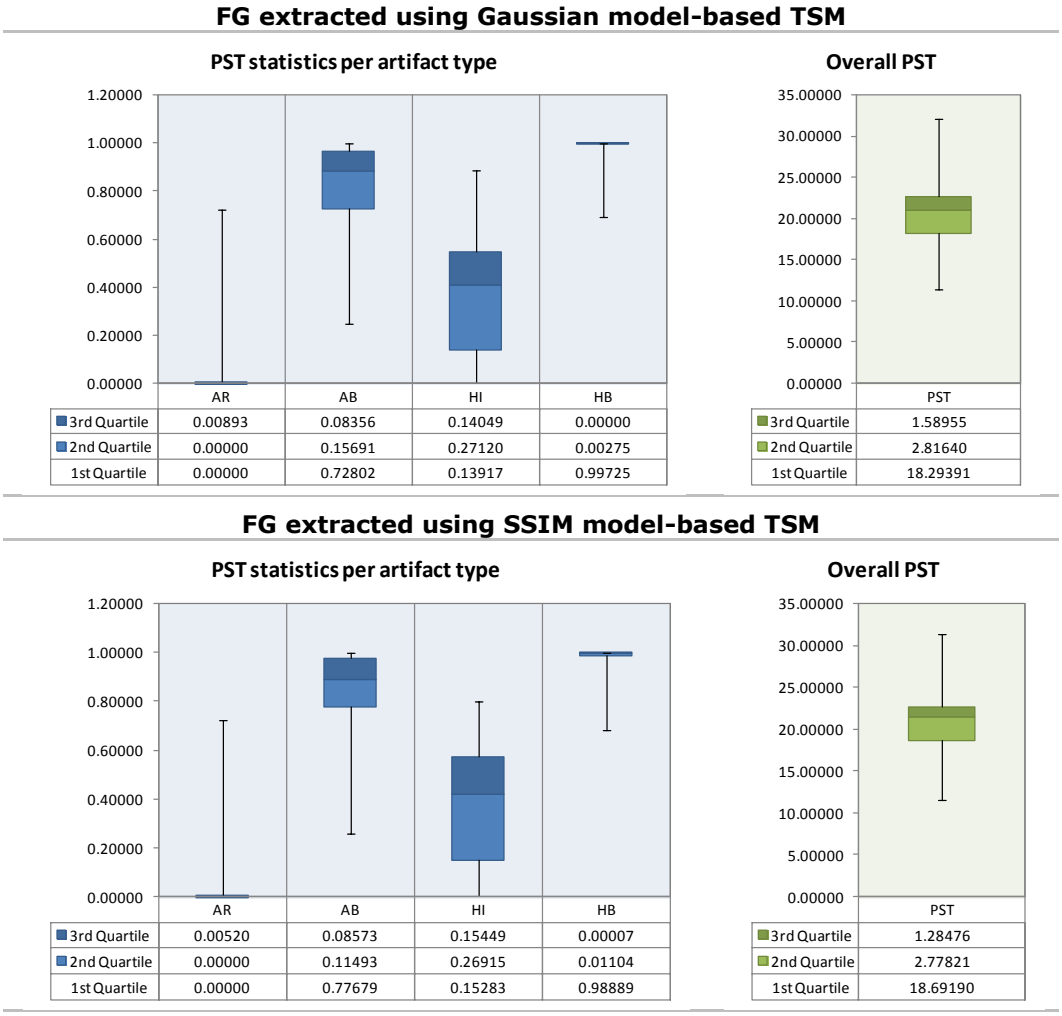


Figure 6-6. Summary charts for the quality assessment results

Table 6-3. Side-by-side comparison of Gaussian- and SSIM-based TSM results

	Average		Std. deviation	
	Gaussian TSM	**SSIM TSM**	**Gaussian TSM**	**SSIM TSM**
AR	0.04527	0.03667	0.12285	0.10985
AB	0.82568	0.84547	0.18010	0.16269
HI	0.37304	0.39731	0.23126	0.22357
HB	0.98409	0.98045	0.04577	0.04770
Overall	**20.45530**	**20.83521**	**3.56204**	**3.16174**

Compared with the results presented in [9] for the mixed reality scenario, the FG extraction method described in the present thesis performs better in terms of overall perceived quality than the top performers evaluated using the same metric [140-142]. Our method achieves an average MAV of **20.45**, versus the 23.91, 27.86 and respectively 32.18 produced by the above-mentioned methods.

6.3. Comparison with state of the art

For comparison with the state of the art we refer to the *Bilayer Segmentation of Live Video* method of Criminisi *et al.* [20] introduced in chapter 2.2.2.2. This method achieves a highly accurate segmentation comparable in quality with the output of stereoscopic methods like the one described in [10]. This is attained by relying on S-T priors that are built during an extensive training phase using manually labeled sequences and by tuning the weights of the algorithm's energy terms for the different types of video sequences that must be segmented [21].

The quality assessment results available for the state of the art method involve a simpler form of measurement than the perceptual metric applied in chapter 6.2 for our own approach. This measurement, which follows the one from [10], takes into account the number of misclassified pixels in respect to a ground-truth (GT) segmentation, reported as a percentage of the image area.

In order to compare the error rates of the proposed FG extraction method with the ones reported in [20] we have used the same video sequences and GT segmentation, available at [http://research.microsoft.com/en-us/projects/i2i/data.aspx]. The GT segmentation is provided for every 5 or 10 frames depending on the sequence, thus marking the moments at which the percentage errors (PE) can be calculated.

The comparative quality assessment results are illustrated in Figure 6-7, for the *AC*, *JM*, *MS* and *VK* sequences, respectively. The sequences present a significant color distribution similarity between FG and BG, making it sometimes difficult to distinguish the object boundaries without prior knowledge of the GT segmentation. Based on the obtained results, the following can be observed:

- The *AC* and *VK* sequences exhibit an initial burn-in period, also reported in the original paper of Criminisi *et al*. This is caused by the lack of significant motion during the first frames in the sequence, and makes it hard for both algorithms to converge to the actual object boundaries. After sufficient motion cues have been collected, both algorithms converge to an accurate FG representation. This effect is also present in the *MS* sequence for our method and is counteracted in the state of the art by the use of prior models.

- For the *AC* sequence the state of art method benefits from the trained priors in order to filter out small AB artifacts along the object boundary, which are almost identical in color to the subject's clothing. It achieves PE values close to 1%, while our method achieves a mean percentage error (MPE) of 4%. Both segmentations are accurate and comparable in quality, our method closely following the reference one.

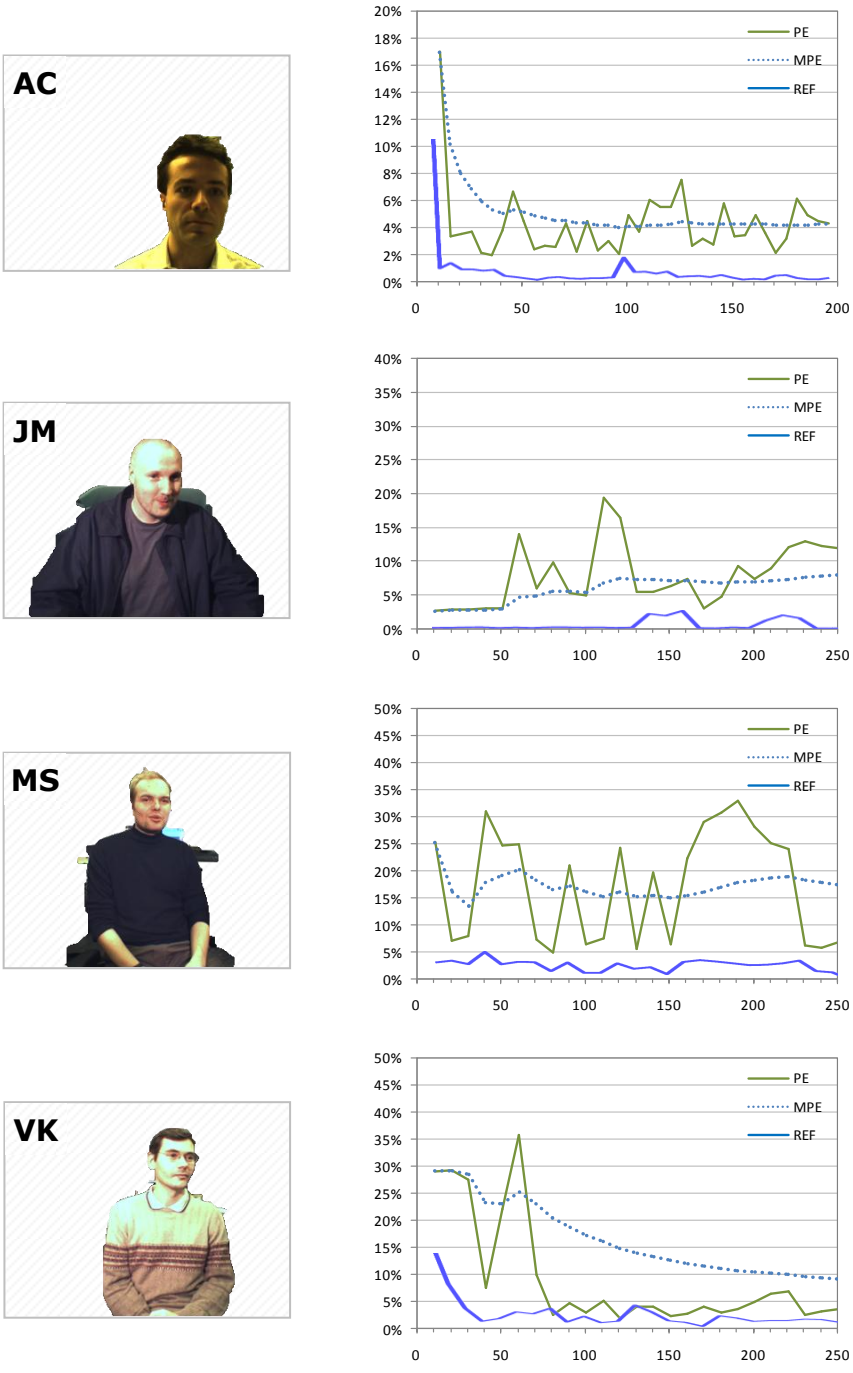


Figure 6-7. Segmentation accuracy compared with the state of the art in [20]

- The *JM* sequence poses more challenge due to presence of dark and smooth subject clothing on top of a dark background, slightly different in color. This causes occasional flickering HB artifacts in our method, which achieves an MPE of 7%, while the reference method is more stable grace to its fusion of motion, color and prior information, achieving an MPE of less than 1%.

- *MS* is the most challenging case for our approach, as the subject's upper body garments are black with smooth texture and low reflectivity. The TSM has difficulties in persisting those regions, especially due to occlusions caused by the subject's hand motion, which results in large HB and IH artifacts. As a consequence, the MPE produced by our method is situated between 15 and 20%, compared to the 3% achieved by the reference method, which uses color fusion and priors to avoid this unwanted behavior.

- In the *VK* sequence the main difference in segmentation is achieved between frames 50 and 70, where the subject performs large hand motions that cover most of the screen. In our case these motions are accompanied by larger AB artifacts, similar to those illustrated in Figure 6-4, while the state of art method is more resilient again due to the presence of trained priors. After frame 80 both algorithms achieve very accurate levels of segmentation quality, with PE values situated around 3-4%.

From the results and observations presented above, we can conclude that the proposed FG extraction method achieves a segmentation quality that is, in most of the cases, comparable to the one produced by the state of the art method of [20]. For certain sequences the reference method shows only marginal improvements and there is only one case in which the difference is significant, namely the handling of smooth and dark FG regions.

Another important aspect is that the state of the art requires prior trainings and parameter calibration depending on the sequence under test, while our approach eliminates this step and works out-of-the-box with any video sequence. Performance-wise, our approach is significantly faster when compared with the results reported in [21] for the method of Criminisi *et al.* This confirms the validity of the proposed FG segmentation method and its applicability to videoconferencing applications.

6.4. Summary

The 6th chapter of the present thesis was dedicated to the analysis of the experimental results obtained after implementing and testing the proposed approach to FG segmentation in monocular video sequences. The analysis has covered two crucial aspects related to the success of the algorithm in the context of videoconferencing applications: execution performance and segmentation quality.

The FG extraction system obtained by implementing the proposed method shows near real-time performance capabilities and is able to take advantage of multiprocessor architectures through parallel algorithm execution. The CPU-based implementation covers only a part of the algorithm's potential, as GPU implementations - not covered by our analysis - hold the promise for a further boost of execution speed.

Segmentation quality has been evaluated using a state of the art perceptual objective metric, which takes human psychology into account in order to assess the impact produced by segmentation defects. The results have proven the accuracy of the proposed method, which performs better than the top performers evaluated with the same metric [9]. In addition, the quality was also compared to the one produced by a state of the art method [20]. Our approach reaches a comparable level of accuracy, slightly below the reference method, without relying on prior trainings in respect to the video sequences being segmented.

We can safely state that the analysis results described in this chapter support the validity of the FG extraction method and offer an indication that the objectives set at the beginning of our thesis have been reached.

7. CONCLUSION AND FUTURE WORK

7.1. Conclusion

In this thesis we have presented solutions to the problem of foreground extraction in monocular video sequences, with direct applicability to immersive videoconference applications. We have introduced a new *model-less* method for FG / BG segmentation which relies exclusively on motion flow analysis and color information in order to expose the foreground objects in the observed scene. The main differentiator of this approach is the absence of any *prior* knowledge about the scene composition - in the form of assumptions or training - which makes it suitable for immediate use on any video sequence.

More specifically, we have investigated the different algorithms and techniques applicable to motion flow analysis and have shown how sparse and dense optic flow estimations can be aggregated in order to increase the accuracy and robustness of motion segmentation and to reveal the presence of FG objects. We have also demonstrated how color and contrast information can be used in conjunction with motion data in order to build a temporally stable FG representation. By further integrating a GC algorithm into the frame processing workflow together with a heuristic approach to unsupervised hard constraints generation, we have obtained a system capable of producing a pixel-accurate object segmentation.

The performance of our FG extraction system has been evaluated from the perspective of execution speed and segmentation quality, respectively. In terms of speed, we have achieved near-RT execution times using algorithm parallelization techniques that are supported by the vast majority of modern CPU architectures. The performance results have proven that the proposed approach is capable of meeting the temporal and CPU usage requirements of videoconferencing systems. Quality-wise, segmentation results have been assessed using a state-of-the-art perceptual objective evaluation metric, which has validated the accuracy of our method by placing it among the top performers in the field.

The binary segmentation method introduced in this thesis approaches the quality of state-of-the-art techniques without sacrificing execution performance or requiring complex training or initialization phases. Based on the current state of research it is possible to identify new directions to further increase the accuracy and reliability of the presented FG extraction system on the path towards obtaining the best possible segmentation for videoconference applications.

7.2. Summary of contributions

The present thesis brings a series of contributions to the field of foreground extraction in monocular video sequences. They are as follows:

- A *novel algorithm for accurate motion segmentation* between pairs of consecutive video frames [71], introduced in chapter 3.
    - Starting from the over-smoothed, noise-sensitive dense OF field and the data obtained from tracking sparse flow features located on image edges, the algorithm identifies the concave hull of

moving scene elements, providing an accurate solution to an ill-posed problem.
    - The blend of dense and sparse OF techniques and the use of strong image gradients ensures resilience to noise, compression artifacts and smooth illumination changes.
    - According to our knowledge, this is the first attempt to fusing dense and sparse OF estimations for such purpose.
- A technique for building *temporally stable masks* as images of FG exposed through motion, presented in chapter 4.
    - With the TSM technique we introduce a lightweight approach to model-less S-T segmentation. The cues generated by the motion detection phase are treated as positive priors and support the labeling of FG regions in the TSM. Labeling persistence is then ensured until significant change occurs in the underlying pixel color distribution, as revealed by negative priors computed from image statistics.
    - Compared to other algorithms, our technique can accommodate different ways of exploiting image color information, as proven using both Gaussian and SSIM-based statistical models.
- An *heuristic algorithm for generating the hard constraints for an unsupervised graph cut segmentation*, described in chapter 5.
    - This approach leverages the results obtained from the TSM and the sparse OF in order to generate the seeds that direct the max-flow/min-cut algorithm through obtaining the most probable binary segmentation for an input frame.
    - The algorithm uses elements from computational geometry like Delaunay triangulation and drape models to define the set of hard constraints, which makes it very fast compared to existing approaches.
    - The inputs consist of a set of point features and a mask that provides coarse FG cues. They can be obtained through any method, making the algorithm potentially attractive for other applications that rely on GC for object segmentation.
- The *experimental validation of the introduced concepts and methods*, detailed and discussed in chapter 6.
    - The segmentation quality produced by the FG extraction system is assessed using a perceptual objective evaluation metric. This approach is better suited for evaluating the impact produced by the results on a human observer and, consequently, for analyzing the applicability of the introduced method to immersive video conferencing.
    - In parallel, the results are compared to those produced by one of the best performers in the state of the art by using the same video sequences and quality metrics.
    - The assessment results prove that our method is comparable with the state of the art in terms of quality and execution performance with the added benefit of completely removing the need of prior initialization or training phases.

7.3. Open points

Although not 100% accurate, our method has produced a perceptual assessment score that places it above top performers evaluated using the same metric. During the experimental phase and the evaluation of segmentation results, we have identified several open points that have a negative impact on the accuracy of the foreground extraction process.

7.3.1. Handling of dark and smooth FG regions

The most important open point is related to the handling of dark and smooth FG regions. This aspect has been previously mentioned in chapter 4.5 and had an adverse effect on the segmentation of the *MS* and *JM* sequences, as discussed in chapter 6.3. The problem stems from the fact that small changes in the color distribution of dark regions (which exhibit RGB values closer to zero than other regions) are more likely to exceed the similarity threshold introduced in equation (4.3) than for other regions and to trigger a FG → BG label change in the TSM.

The under-segmentation effect observed on dark and smooth image regions is further accentuated by the low reliability of the optic flow estimation on these image areas, as discussed in section 3.2.4.1. The state-of-the-art methods discussed in chapter 2.2 rely mostly on learned priors [20; 22] or assumptions about the scene contents [23] in order to overcome this behavior, but such an approach would conflict with the objectives set at the beginning of our research (model-less FG extraction).

7.3.2. Absence of significant motion

As stated in the title of this thesis, motion is the keystone of our approach to binary FG/BG segmentation. The first stage in the processing pipeline of the FG extraction system is dedicated to motion detection and segmentation in order to generate cues about FG elements based on the observed motion in the scene.

As expected, the absence of motion in the video sequence will turn the algorithm blind, as there are no other indications about the location of FG objects. Thus, an important open point is related to bootstrapping the algorithm when lacking significant motion information.

7.3.3. Occlusion handling

The negative influence of occlusions has been observed when persisting FG information in the TSM and has been described in chapter 4.5 and illustrated in Figure 4-5. The phenomenon can be frequently encountered in videoconferences, when participants use gestures as part of non-verbal communication. Combined with dark and textureless clothing, occlusions caused by hand motions can affect the stability of the TSM, as seen on sequence *MS* in chapter 6.3.

The negative effects of occlusion have been alleviated by the introduction of the GC algorithm in the last stage of the frame processing pipeline. The smoothness term of the GC energy formulation is capable of restoring most of the FG regions lost due to occlusion. However, this restoration does not propagate into the TSM,

which brings another open question on how this could be achieved as part of the TSM synchronization step detailed in section 5.3.4.

7.3.4. Camera instability

The FG extraction algorithm introduced in this thesis works based on the assumption that the video capture hardware remains in the same stable position during the streaming / recording of the video sequence. Failure to meet this requirement would introduce false motion in an otherwise stationary BG, causing an amount of false positives directly proportional with the camera displacement magnitude and the level of clutter in the background. Handling this scenario is still an open point of our present approach.

7.4. Future research perspectives

Based on the obtained experimental results and previous discussions, we finalize the thesis with an outline of future work and research directions.

The most significant and immediately appealing direction is towards advanced statistical modeling in the S-T motion integration algorithm. Addressing the instability of dark and smooth FG regions in the TSM may follow the approach described in [143] using the HSV color space combined with dynamic background learning. Another possibility is to substitute the Gaussian and SSIM models with MoG-based models as described in [35, 144] or Local Binary Patterns [145, 146].

Algorithm initialization (also known as bootstrapping) represents the next important research direction, especially useful for scenarios characterized by absence of significant motion. Video chat applications are a prime beneficiary, since motion may not occur immediately after a person joins the conference. Foreground information may be collected from other sources than motion, by using face / body detection models [67] or skin color distributions [147], with the important remark that such approaches would transform the proposed method into a model-based one.

The third perspective is oriented towards an improved algorithm for synchronizing the TSM and the GC segmentation. The main goal is to increase the method's resiliency to occlusions, by identifying the occluded FG areas that are restored by the GC segmentation in order to propagate them back into the TSM.

Methods to detect and compensate for unwanted camera motion (camera shake) are part of the last research direction. The camera shake and compensation algorithm can take as input the OF estimation in either sparse on dense form, as proposed in [148] and respectively [149, 150]. Such approaches can be implemented with low penalty as part of the existing method, which already computes and processes both OF flavors, and can also be augmented in order to take advantage of the available FG information contained in the TSM.

REFERENCES

[1] E. Prakash, J. Wood, B. Li, M. Clarke, G. Smith, and K. Yates, "Games Technology: Console Architectures, Game Engines and Invisible Interaction", in *Proc. of the 4th Annual International Conference on Computer Games, Multimedia and Allied Technology (CGAT 2011)*, 103-108, 2011.

[2] A. Bruhn, J. Weickert, and C. Schnorr, "Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods", in *International Journal of Computer Vision*, 61(3):211–231, 2005.

[3] L. Grady, M. P. Jolly and A. Seitz, "Segmentation from a Box", in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, 367-374, 2011.

[4] C. Gao, K. Huang, and T. Tan, "Improvements on MID Based Foreground Segmentation Using Optical Flow", in *Second CJK Joint Workshop on Pattern Recognition (CJKPR 2010)*, November 2010.

[5] P. Kauff, and O. Schreer, "An immersive 3D video-conferencing system using shared virtual team user environments", in *Proc. of the 4th International Conference on Collaborative Virtual Environments*, 105-112, 2002.

[6] E. Woyke, "The Business Case For Immersive Communications", *Forbes*, March 2010, available: http://www.forbes.com/sites/elizabethwoyke/2010/12/03/the-business-case-for-immersive-communications.

[7] A. DeVicenzi, L. Yao, H. Ishii, and R. Raskar, "Kinected conference: augmenting video imaging with calibrated depth and audio", in *Proc. of the ACM 2011 conference on Computer supported cooperative work*, 621-624, 2011.

[8] B. Choudhury and S. Chandran, "A Survey of Image-based Relighting Techniques", *Journal of Virtual Reality and Broadcasting*, 4(7), 2007.

[9] E.D. Gelasca, and T. Ebrahimi, "On Evaluating Video Object Segmentation Quality: A Perceptually Driven Objective Metric", in IEEE Journal of Selected Topics in Signal Processing, 3(2):319-335, 2009.

[10] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross and C. Rother, "Bi-Layer Segmentation of Binocular Stereo Video", in *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, 407-414, 2005.

[11] Y. Boykov and M. P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images", in *Proc. of the International Conference on Computer Vision*, 1:105-112, 2001.

[12] J.-M. Gottfried, J. Fehr and C. S. Garbe, "Computing range flow from multi-modal Kinect data", in *Proc. of the 7th international conference on Advances in visual computing (ISVC'11)*, 1:758-767, 2011.

[13] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison and A. W. Fitzgibbon, "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera", in *Proc. ACM Symposium on User Interface Software and Technology*, 559-568, 2011.

[14] M. Piccardi, "Background subtraction techniques: a review," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, 2004.

[15] P. Kumar, K. Sengupta, and S. Ranganath, "Real Time Detection and Recognition of Human Profiles Using Inexpensive Desktop Cameras", in *Proc. 15th International Conference on Pattern Recognition*, 1096-1099, 2000.

[16] H. Kim, R. Sakamoto, I. Kitahara, T. Toriyama, and K. Kogure, "Robust Foreground Segmentation from Color Video Sequences using Background Subtraction with Multiple Thresholds", *Technical report of IEICE. PRMU*, 106(376):135-140, 2006.

[17] J. Sun, W. Zhang, X. Tang and H.-Y. Shum, "Background cut", in *Proc. European Conference on Computer Vision*, 2:628-641, 2006.

[18] Q. Liu, H. Li and K. N. Ngan, "Automatic body segmentation with graph cut and self-adaptive initialization level set (SAILS)", in *J. Vis. Comun. Image Represent.*, 22:367-377, 2011.

[19] H. Li, K. N. Ngan and Q. Liu, "FaceSeg: Automatic Face Segmentation for Real-time Video", in *IEEE Transactions on Multimedia*, 11:77-88, 2009.

[20] A. Criminisi, G. Cross, A. Blake and V. Kolmogorov, "Bilayer Segmentation of Live Video", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 53-60, 2006.

[21] Y. Wang, "Foreground-Background Segmentation of Video Sequences", Technical report, California Institute of Technology, United States, 2006.

[22] P.Yin, A. Criminisi, J. Winn, and I. Essa, "Bilayer Segmentation of Webcam Videos Using Tree-Based Classifiers", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):30-42, January 2011.

[23] J. Kim, H.J. Lee, T.H. Lee, M. Cho, and J.B. Lee, "Hardware/Software Partitioned Implementation of Real-time Object-oriented Camera for Arbitrary-shaped MPEG-4 Contents", in Proc. of the 2006 IEEE/ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia, 7-12, 2006.

[24] G. Bradski, and A. Kaehler, Learning OpenCV, 1st edition, O'Reilly Media, Sebastopol, California, September 2008.

[25] J. Kim, J. Zhu, and H.J. Lee, "Block-level Processing of a Video Object Segmentation Algorithm for Real-time Systems", in *IEEE International Conference on Multimedia and Expo*, 2066-2069, 2007.

[26] K. Y. Wong, and M. E. Spetsakis, "Motion Segmentation and Tracking", in *Proc. 15th International Conference on Vision Interface*, 80–88, 2002.

[27] D. Hendriksen, "Segmentation and Tracking of Deformable Objects in Video", Technical report, University of Delft, The Netherlands, August 2005.

[28] O. Barnich, and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences", in *IEEE Transactions on Image Processing*, 20(6):1709-1724, 2011.

[29] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: A systematic survey," *IEEE Transactions on Image Processing*, 14:294–307, 2005.

[30] Y. Benezeth, P. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *IEEE International Conference on Pattern Recognition (ICPR)*, 1–4, 2008.

[31] T. Bouwmans, F. El Baf, and B. Vachon, "Statistical background modeling for foreground detection: A survey," in Handbook of Pattern Recognition and Computer Vision (volume 4), ch. 3, pp. 181–199, World Scientific Publishing, January 2010.

[32] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Realtime tracking of the human body," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785, July 1997.

[33] C. Stauffer, and W. Grimson, "Adaptive background mixture models for real-time tracking," *Computer Vision and Pattern Recognition*, 2:246-252,1999.

[34] J. Cezar, C. Rosito, and S. Musse, "A background subtraction model adapted to illumination changes," *in IEEE International Conference on Image Processing (ICIP)*, 1817–1820, 2006.

[35] T. Bouwmans, F. El Baf, and B. Vachon, "Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey", *Recent Patents on Computer Science*, 1(3):219-237, 2008.

[36] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground–background segmentation using codebook model", in *Real-Time Imaging*, 11(3):172-185, 2005.

[37] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm", *Journal of Royal Statistical Society*, 39:1-38, 1977.

[38] S. Borman, "The Expectation Maximization Algorithm. A short tutorial", Technical report, University of Notre Dame, July 2004.

[39] T.K. Moon, "The expectation-maximization algorithm", in *IEEE Signal Processing Magazine*, 13(6):47–60, 1996.

[40] M. Figueirdeo, and A. Jain, "Unsupervised Learning of Finite Mixture Models," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[41] J. G. Brankov, N. P. Galatsanos, Y. Yang, and M. N. Wernick, "Similarity based clustering using the expectation maximization algorithm", in *Proc. of IEEE Int'l Conf. on Image Proc.*, 97-100, 2002.

[42] H. Yalcin, M. Black, and R. Fablet, "The dense estimation of motion and appearance in layers", *Second IEEE Workshop on Image and Video Registration (IVR'04)*, 2004.

[43] W. Abd-Almageed, A. El-Osery, and C. Smith, "Non-Parametric Expectation Maximization: A Learning Automata Approach", in *IEEE International Conference on Systems, Man and Cybernetics*, 3:2996-3001, 2003.

[44] B.K.P. Horn, and B.G. Schunck, "Determining optical flow.", *Artificial Intelligence*, 17:185-203, 1981.

[45] B.D. Lucas, and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", in *Proc. 7th Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 674-679, 1981.

[46] D. Zhang, and G. Lu, "Segmentation of Moving Objects in Image Sequence: A Review", in *Circuits, Systems, and Signal Processing*, 20(2):143-183, March 2001.

[47] L. Zappella, X. Llado, and J. Salvi, "Motion Segmentation: a Review", in *Proc. 2008 Conference on Artificial Intelligence Research and Development*, 398-407, 2008.

[48] A. Utasi, and L. Czuni, "Reducing the Foreground Aperture Problem in Mixture of Gaussians Based Motion Detection", in *Proc. 14th International Workshop on Systems, Signals and Image Processing (SSIP)*, pp. 157-160, June 2007.

[49] G. Eibl and N. Brandle, "Evaluation of clustering methods for finding dominant optical flow fields in crowded scenes", in *Proc. 19th International Conference on Pattern Recognition (ICPR 2008)*, 1-4, 2008.

[50] D. Comaniciu, and P. Meer, "Mean shift: A robust approach toward feature space analysis", in *IEEE Trans. Patt. Analy. Mach. Intell.,* 24:603–619, 2002.

[51] A. Yilmaz, X. Li, and M. Shah, "Contour based object tracking with occlusion handling in video acquired using mobile cameras", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, 2004.

[52] J. Shi, and J. Malik, "Normalized cuts and image segmentation", in *IEEE Trans. Patt. Analy. Mach. Intell.,* 22(8):888–905, 2000.

[53] Y. Boykov and V. Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision", in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(9):1124-1137, 2004.

[54] M. Yokohama, and T. Poggio, "A Contour-Based Moving Object Detection and Tracking", in *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 271-276, 2005.

[55] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey", in *ACM Computing Surveys*, 38(4):Article 13, December 2006.

[56] R. Mech, and M. Wollborn, "A noise robust method for 2D shape estimation of moving objects in video sequences considering a moving camera", *Signal Processing* 66(2):203-217, 1998.

[57] R. Stolkin, A. Greig, M. Hodgetts, and J. Gilby, "An em/e-mrf algorithm for adaptive model based tracking in extremely poor visibility." *Image and Vision Computing*, 26(4):480–495, 2008.

[58] D. Cremers, and S. Soatto, "Motion competition: A variational approach to piecewise parametric motion segmentation," *International Journal of Computer Vision*, 62(3):249–265, 2005.

[59] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J. Buhmann, "Topology free hidden markov models: Application to background modeling", in *IEEE International Conference on Computer Vision (ICCV)*, 294–301, 2001.

[60] D. Comaniciu, V. Ramesh, and P. Andmeer, "Kernel-based object tracking", in *IEEE Trans. Patt. Analy. Mach. Intell.* 25:564–575, 2003.

[61] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance", in *IEEE International Conference on Computer Vision (ICCV)*, 734–741, 2003.

[62] M. Tkalcic, and J. F. Tasic, "Colour spaces - perceptual, historical and applicational background", in *Proc. EUROCON 2003. Computer as a tool*, 304-308, 2003.

[63] M. Fagadar-Cosma, "Extragerea automată a contururilor din hărţi raster stocate în format digital", Technical Report, "Politehnica" University of Timisoara, Romania, July 2008.

[64] S. Haddad, "Texture Measures for Segmentation", M.Sc. dissertation, University of Cape Town, South Africa, April 2007.

[65] J. Shi, and C. Tomasi, "Good features to track", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 593–600, 1994.

[66] D. Lowe, "Distinctive image features from scale-invariant keypoints", *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[67] P. Viola, and M. Jones, "Robust Real-time Object Detection", in *International Journal of Computer Vision*, pp. 1-25, 2001.

[68] J.G. Allen, R.Y.D. Xu, and J.S. Jin, "Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces", in *Proceedings of the Pan-Sydney area workshop on Visual information processing (VIP'05)*, 2004.

[69] B. Li, R. Chellappa, Q. Zheng, and S. Der, "Model-based temporal object verification using video", in *IEEE Trans. Image Process.*, 10(6):897–908, 2001.

[70] Y. Chen, Y. Rui, and T. Huang, "Jpdaf based hmm for real-time contour tracking", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 543–550, 2001.

[71] M. Fagadar-Cosma, V. I. Cretu and M. V. Micea, "Dense and Sparse Optic Flows Aggregation for Accurate Motion Segmentation in Monocular Video Sequences", in *Proc. 9th International Conference on Image Analysis and Recognition (ICIAR 2012)*, Aveiro, Portugal. Vol. Aurelio Campilho and Mohamed Kamel (Eds.): Part I, LNCS 7324:208-215, 2012.

[72] M. J. Tarr, and M. Black, "A computational and evolutionary perspective on the role of representation in computer vision", Technical Report YALEU/DCS/RR-899, Yale University, October 1991.

[73] M.J. Black, "Robust Incremental Optical Flow", PhD thesis YALEU/CSD/RR-923, Yale University, September 1992.

[74] B. K. P. Horn, Robot Vision, The MIT Press, Cambridge, Massachusetts, 1986.

[75] S. Baker, D. Scharstein, J.P. Lewis, S. Roth, M.J. Black, and R. Szeliski, "A Database and Evaluation Methodology for Optical Flow", *International Journal of Computer Vision*, 92: 1–31, 2011.

[76] D. J. Fleet, and Y. Weiss, "Optical flow estimation". In N. Paragios, Y. Chen, and O. Faugeras, editors, Handbook of Mathematical Models in Computer Vision, chapter 15, pages 239–258. Springer, 2006.

[77] S. Baker, and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework", *International Journal of Computer Vision*, 56(3):221–255, 2004.

[78] B.D. Lucas, "Generalized Image Matching by the Method of Differences", Ph.D. thesis, Carnegie-Mellon University, United States, July 1984.

[79] J. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm," OpenCV Document, Intel, Microprocessor Research Labs, 2000.

[80] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion", in *Proc. of the 13th Scandinavian Conference on Image Analysis*, 363-370, 2003.

[81] G. Farnebäck , "Polynomial Expansion for Orientation and Motion Estimation", Linkoping Studies in Science and Technology, Dissertation No. 790, Linkoping University, Sweden, November 2002.

[82] C. Tomasi, and T. Kanade, "Detection and Tracking of Point Features", Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[83] W. Cheney, and D.R. Kincaid, Numerical mathematics and computing, 6th edition, Cengage Learning, Belmont, California, 2007.

[84] K.G. Derpanis, "The Harris Corner Detector", Tecnhical report, York University, Canada, October 2004.

[85] C. Harris, and M.J. Stephens, "A combined corner and edge detector", *in Alvey Vision Conference*, pp. 147–152, 1988.

[86] J. K. Suhr, "Kanade-Lucas-Tomasi (KLT) Feature Tracker", Course notes, Yonsei University, Korea, 2009.

[87] O. Mac Aodha, G.J. Brostow, and M. Pollefeys, "Segmenting Video Into Classes of Algorithm-Suitability", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1054-1061, 2010.

[88] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, "Performance of Optical Flow Techniques", *International Journal of Computer Vision*, 12(1):43–77, 1994.

[89] T. Portz, L. Zhang and H. Jiang, "Optical Flow in the Presence of Spatially-Varying Motion Blur", in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[90] J. Molnar, D. Chetverikov, and S. Fazekas, "Illumination-robust variational optical flow using cross-correlation", in *Journal of Computer Vision and Image Understanding*, 114(10):1104-1114, 2010.

[91] X. Yang, and R. Klette, "Evaluation of Motion Analysis on Synthetic and Real-World Image Sequences", Technical report MItech-TR-58, University of Auckland, New Zealand, 2010.

[92] O. Al-Shaykh, R. Neff, D. Taubman, and A. Zakhor, "Video Sequence Compression", in V.K. Madisetti, editor, The Digital Signal Processing Handbook, 3rd Edition, chapter 20, CRC Press, 2009.

[93] T. Stich, and M. Magnor, "Image Morphing for Space-Time Interpolation", Technical report 2007-4-2, Technical University of Braunschweig, Germany, April 2007.

[94] T. Brox, B. Rosenhahn, J. Gall, and D. Cremers, "Combined Region and Motion-Based 3D Tracking of Rigid and Articulated Objects", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):402-415, 2010.

[95] S. Waldherr, S. Thrun and R. Romero, "A Gesture Based Interface for Human-Robot Interaction", in *Autonomous Robots*, 9(2):151-173, 2000.

[96] M. Hwangbo, J.S. Kim, and T. Kanade, "Inertial-Aided KLT Feature Tracking for a Moving Camera", in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1909-1916, October 2009.

[97] A. Sturm, "Geometric Approximations in Two- and Three Dimensional Space", Ph.D. thesis, Freie Universitat Berlin, Germany, 2009.

[98] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, "A Gentle Introduction to Bilateral Filtering and its Applications", in *Proc. ACM SIGGRAPH'08*, 2008.

[99] C. Tomasi, and R. Manduchi, "Bilateral Filtering for Gray and Color Images", in *IEEE International Conference on Computer Vision (ICCV)*, 839-846, 1998.

[100] M. Fagadar-Cosma, M.V. Micea, and V. Cretu, "Obtaining Highly Localized Edges using Phase Congruency and Ridge Detection", in *Proc. of International Conference of Computational Cybernetics and Technical Informatics (ICCC-CONTI)*, 33-342, 2010.

[101] J.F. Rivest, P. Soille, and S. Beucher, "Morphological gradients", in *Journal of Electronic Imaging*, 326(2), 1993.

[102] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High Accuracy Optical Flow Estimation Based on a Theory for Warping", in *Proc. 8th European Conference on Computer Vision*, 4:25-36, 2004.

[103] R. Fabbri, L.F. Costa, J.C. Torelli, and O.M. Bruno, "2D Euclidean Distance Transform Algorithms: A Comparative Survey", in *ACM Computing Surveys*, 40(1), 2008.

[104] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models", *International Journal of Computer Vision*, 18:321-331, 1988.

[105] M. O. Berger, "Snake growing", *Lecture Notes in ComputerScience*, 427:387-396, 1990.

[106] D. J. Williams, and M. Shah, "A Fast Algorithm for Active Contours and Curvature Estimation", *CVGIP: Image Understanding*, 55(1): 14-26, 1992.

[107] C. Xu and J. L. Prince, "Snakes, Shapes and Gradient Vector Flow", in *IEEE Trans. of Image Processing*, 7(3):359-369, 1998.

[108] L. Zhang, "Active Contour Model, Snake", Technical report, University of Nevada, United States, 2001.

[109] W. Kasprzak, and P. Skrzynski, "Hand image interpretation based on double active contour tracking", in *ROMANSY 16. Robot Design, Dynamics, And Control.*, *CISM Courses and Lectures,* 487:439-446, Springer, 2006.

[110] E. Kienel, M. Vanco and G. Brunett, "Speeding up snakes", in *Proc. VISAPP '06*, 1:323-330, 2006.

[111] M.S. Nikulin, "Hellinger distance", in M. Hazewinkel (Eds.), *Encyclopedia of Mathematics*, Springer, 2001.

[112] K.G. Derpanis, "The Bhattacharyya Measure", Technical report, York University, Canada, March 2008.

[113] M.S. Khalid, M.U. Ilyas, M.S. Sarfaraz, and M.A. Ajaz, "Bhattacharyya Coefficient in Correlation of Gray-Scale Objects", in *Journal of Multimedia*, 1(1):56-61, 2006.

[114] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity", in *IEEE Trans. on Image Processing*, 13(4):600-612, 2004.

[115] Z. Wang and A. C. Bovik, "A universal image quality index," in *IEEE Signal Processing Letters*, 9:81–84, 2002.

[116] S. Ince, and J. Konrad, "Occlusion-Aware Optical Flow Estimation", in *IEEE Transactions on Image Processing*, 17(8):1443-1451, August 2008.

[117] Y. Boykov and G. Funka-Lea, "Optimal object extraction via constrained graph-cuts", in *International Journal of Computer Vision (IJCV)*, 2004.

[118] Y. Li, J. Sun, C.-K. Tang and H.-Y. Shum, "Lazy snapping", in *ACM Trans. Graph.*, 23(3):303-308, 2004.

[119] C. Rother, V. Kolmogorov and A. Blake, "Grabcut:Interactive Foreground Extraction Using Iterated Graph Cuts", in *ACM Transactions on Graphics*, 23:309–314, 2004.

[120] J.-S. Kim and K.-S. Hong, "Color-texture segmentation using unsupervised graph cuts", *J. Pattern Recognition*, 42(5):735-750, 2009.

[121] K. T. Park, J. H. Lee and Y. S. Moon, "Unsupervised foreground segmentation using background elimination and graph cut techniques", *Electronic Letters*, 45(20):1025-1027, 2009.

[122] Y. Boykov, O. Veksler and R. Zabih, "Markov Random Fields with Efficient Approximations", in *Proc. IEEE conference on "Computer Vision and Pattern Recognition" (CVPR)*, 648-655, 1998.

[123] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[124] S. Z. Li, "A Markov Random Field Model for Object Matching under Contextual Constraints", in *Proc. IEEE conference on "Computer Vision and Pattern Recognition" (CVPR)*, 866-869, 1994.

[125] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Belief Propagation for Early Vision", in *Int. J. Comp. Vis.*, 70, 2006.

[126] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen and C. Rother, "A comparativestudy of energy minimization methods for Markov random fields", in *Proc. European Conference on Computer Vision*, 2:16–29, 2006.

[127] S. Z. Li, "Markov Random Field Modeling in Computer Vision", 1st edition, Springer-Verlag, New York, 1995.

[128] Y. Boykov and G. Funka-Lea, "Graph Cuts and Efficient N-D Image Segmentation", in *International Journal of Computer Vision (IJCV)*, 70(2):109-131, 2006.

[129] L. Ford and D. Fulkerson, "Flows in Networks", Princeton University Press, 1962.

[130] A. Goldberg and R. Tarjan, "A new approach to the maximum flow problem", in *J. ACM*, 35(4):921–940, 1988.

[131] Y. Boykov, O. Veksler and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts", in *IEEE Transactions on PAMI*, 23(11):1222-1239, 2001.

[132] M. Fagadar-Cosma, "Aggregation of Dense and Sparse Optical Flows for Foreground Identification and Persistence in Videoconferencing Systems", Ph.D. report, "Politehnica" University of Timisoara, Romania, 2012.

[133] J. Canny, "A computational approach to edge detection", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[134] J. Chase, B. Nelson, J. Bodily, Z. Wei and D. J. Lee, "Real-Time Optical Flow Calculations on FPGA and GPU Architectures: A Comparison Study", in *Proc. IEEE Symposium on FPGAs for Custom Computing Machines (FCCM '08)*, 173-182, 2008.

[135] N. Sundaram, T. Brox and K. Keutzer, "Dense Point Trajectories by GPU-Accelerated Large Displacement Optical Flow", in *Proc. 11th European Conference on Computer Vision*, 438-451, 2010.

[136] Z. He and F. Kuester, "GPU-Based Active Contour Segmentation Using Gradient Vector Flow", in *Proc. ISVC 2006*, 191-201, 2006.

[137] V. Vineet and P. J. Narayanan, "CUDA cuts: Fast graph cuts on the GPU", in *IEEE CS Conf. on Computer Vision and Pattern Recognition Workshops*, 1-8, 2008.

[138] M. Wollborn and R. Mech, "Refined procedure for objective evaluation of video object generation algorithms", in *Proc. 43rd MPEG Meeting*, Tokyo, Japan, 1998, ISO/IECJTCI/SC29/WG11 M3448.

[139] P. Villegas and X. Marichal, "Perceptually-weighted evaluation criteria for segmentation masks in video sequences", in *IEEE Trans. Image Process.*, 13(8):1092–1103, 2004.

[140] T. Horprasert, D. Harwood and L. S. Davis, "A statistical approach for real-time robust background substraction and shadow detection", in *Proc. IEEE ICCV Frame Rate Workshop*, 1–19, 1999.

[141] S. Jabri, Z. Duric, H. Wechsler and A. Rosenfeld, "Detection and location of people in video images using adaptive fusion of color and edge information", in *Proc. Int. Conf. Pattern Recognition (ICPR)*, 627–630, 2000.

[142] J. Shen, "Motion detection in color image sequence and shadow elimination", in *Vis. Commun. Image Process.*, 731–740, 2004.

[143] A. R. J. François and G. G. Medioni, "Adaptive Color Background Modeling for Real-time Segmentation of Video Streams," in *Proc. of the International Conference on Imaging Science, Systems, and Technology*, 227-232, 1999.

[144] X. H. Fang, W. Xiong, B. J. Hu and L. T. Wang, "A Moving Object Detection Algorithm Based on Color Information", in *Journal of Physics: Conference Series*, 48:384-387, 2006.

[145] M. Heikkila, M. Pietikainen, and J. Heikkila, "A texture-based method for detecting moving objects," *British Machine Vision Conference*, pp. 187-196, 2004.

[146] J. Yang, J. Wang, and H. Lu, "A Hierarchical Approach for Background Modeling and Moving Objects Detection", in *International Journal of Control, Automation, and Systems*, 8(5):940-947, 2010.

[147] G. Gomez and E. F. Morales, "Automatic Feature Construction and a Simple Rule Induction Algorithm for Skin Detection", in *Proc. of the ICML Workshop on Machine Learning in Computer Vision*, 31-38, 2002.

[148] P. O'Donovan, "Using Optical Flow for Stabilizing Image Sequences", Technical Report 502425, University of Toronto, Canada, April 2005.

[149] J.-Y. Chang, W.-F. Hu, M.-H. Cheng and B.-S. Chang, "Digital Image Translational and Rotational Motion Stabilization using Optical Flow Technique", in *IEEE Transactions on Consumer Electronics*, 48(1): 108-115, 2002.

[150] S. Baker, E.Bennett, S.B. Kang, and R. Szeliski, "Removing Rolling ShutterWobble", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 2392—2399, 2010.

ANNEX A. LIST OF PUBLISHED PAPERS

[1] M. Fagadar-Cosma, L. D. Faniciu and M. V. Micea, "TETHRA: TDM over Ethernet Router and Analyzer", Proceedings of the 4th International Conference on Microelectronics and Computer Science, ICMCS-05, Chisinau, Republic of Moldova, vol. 2, pp. 36-39, 2005.

[2] V. Bocan and M. Fagadar-Cosma, "Towards DoS-resistant Single Sign-On Systems", Proceedings of the International Conference on "Computer as a Tool", EUROCON 2005, Belgrade, Serbia, vol. 1, pp. 668-671, 2005.

[3] V. Bocan and M. Fagadar-Cosma, "Adaptive Threshold Puzzles", Proceedings of the International Conference on "Computer as a Tool", EUROCON 2005, Belgrade, Serbia, vol. 1, pp. 644-647, 2005.

[4] V. Bocan and M. Fagadar-Cosma, "Scalable and Secure Architecture for Digital Content Distribution", Proceedings of the International Conference on Software in Telecommunications and Computer Networks, SoftCOM 2006, Split, Croatia, pp. 182-187, 2006.

[5] M. Fagadar-Cosma, L. D. Faniciu and M. V. Micea, "Data Routing and Remote Protocol Analysis using the TETHRA System", Proceedings of the 32nd Annual Conference of the IEEE Industrial Electronics Society, IECON'06, Paris, France, pp. 603-608, 2006.

[6] M. Fagadar-Cosma, M. V. Micea and V. I. Cretu, "Obtaining Highly Localized Edges using Phase Congruency and Ridge Detection", Proceedings of the International Joint Conference on Computational Cybernetics and Technical Informatics, ICCC-CONTI, Timisoara, Romania, pp. 339-342, 2010.

[7] V. Bocan and M. Fagadar-Cosma, "Denial of Service Resilience of Authentication Systems", in Digital Identity and Access Management: Technologies and Frameworks, Sharman R., Das Smith S. and Gupta M. Eds., IGI Global, New York, ch. 11, pp. 188-208, 2011.

[8] M. Fagadar-Cosma and M. Fadili, "Method for recognizing gestures and gesture detector", Alcatel-Lucent, Application no. 11290561.7, European Patent Office, 2011.

[9] M. Fagadar-Cosma, "Aggregation of Dense and Sparse Optical Flows for Foreground Identification and Persistence in Videoconferencing Systems", Ph.D. report, "Politehnica" University of Timisoara, Romania, 2012, unpublished.

[10] M. Fagadar-Cosma and M. Casas-Sanchez, "Method for control of a video interface, method for operation of a video interface, face orientation detector, and video conferencing server", Alcatel-Lucent, Application no. 12290086.3, European Patent Office, 2012.

[11] M. Fadili and M. Fagadar-Cosma, "A method for generating an immersive video of a plurality of persons", Alcatel-Lucent, Application no. 12172112.0, European Patent Office, 2012.

[12] M. Fagadar-Cosma, "Accurate Foreground Extraction in Videoconference Systems by using Sparse Optical Flow and Graph Cut Segmentation", Ph.D. report, "Politehnica" University of Timisoara, Romania, 2012, unpublished.

[13] M. Fagadar-Cosma, V. I. Cretu and M. V. Micea, "Dense and Sparse Optic Flows Aggregation for Accurate Motion Segmentation in Monocular Video Sequences", Proceedings of the 9th International Conference on Image Analysis and Recognition, ICIAR 2012, Aveiro, Portugal, Springer-Verlag, vol. LNCS 7324, pp. 208-215, 2012.

[14] M. Fagadar-Cosma, M. Nouri, V. I. Cretu and M. V. Micea, "A Combined Optical Flow and Graph Cut Approach for Foreground Extraction in Videoconference Applications", J. Stud. Inform. Control, ICI Bucharest, Romania, vol. 21, no. 4, 2012, in press.

Disclaimer: The video sequences illustrated in the present thesis are either a) freely available (sequences AC, JM, MS, VK) or b) have been used with the permission of the depicted subjects (sequences Mihai *, Hendrik *, Green Screen *, Mukul *) or c) are part of public broadcasts and have been used according to *fair use* terms (TV Show sequence). The author mentions that no material benefits of any nature have been obtained as a result of this usage.