Motion Detection & Segmentation for Audio-Visual Source Separation

Oscar Divorra Escoda

FINAL DEGREE THESIS

Director: Dr. Pierre Vandergheynst

Lausanne, 31st August 2000

LTS, EPFL

Detecció de Moviment i Segmentació per a Separació de Fonts Audiovisuals

Oscar Divorra Escoda

PROJECTE FINAL DE CARRERA

Director: Dr. Pierre Vandergheynst

Lausanne, 31 d'agost del 2000

LTS, EPFL

(Suïssa)

En els darrers anys, les aplicacions multimèdia han millorat considerablement. Els avanços en sistemes de computació, comunicacions i teoria del senyal, han aportat un entorn propici per a desenvolupar i integrar solucions per permetre a la gent compartir informació i comunicar-se.

Els humans viuen en societat, són una mena d'éssers als quals els cal la comunicació, relacionar-se els uns amb els altres. Des que hi ha gent al Món, sempre han cercat una manera d'expressar els seus pensaments, de compartir les idees. Els primers passos foren l'establiment del llenguatge, seguit de l'escriptura. Des de llavors, s'ha corregut un llarg camí. El Telègraf, el Telèfon, les Radiocomunicacions, les Comunicacions Digitals... Ara, la gent es pot comunicar per tot el Món, immediatament i amb velocitat. Les distàncies no existeixen amb el cable, la fibra o l'aire.

La distància entre dos locutors ha estat reduïda al límit mitjançant la videoconferència. Aquesta permet una plena comunicació audiovisual, aportant a aquells que l'usen per parlar amb el millor comfort possible. La videoconferència possibilita l'existència d'un sol event, únic en temps, en raó, en group de gent, pero divers en espai. Així desapareix la necessitat de que la gent hagi de trobar-se en el mateix lloc físic per poder dur, per exemple, una reunió d'empresa. Cadascú pot restar a la seva oficina o sucursal, i dur a terme un diàleg sense limitacions.

Aquest treball està desenvolupat en el context d'un sistema multiusuari de video-conferència multi-usuari, en el que en cada terminal hi ha més d'una persona participant. Normalment, cal un càmera i un tècnic de so en cada extrem del

sistema de video-conferència. Aquests s'encarreguen d'enfocar la càmera i localitzar la font de so en la persona que està parlant o necessita l'atenció de la camera. Seria desitjable, doncs, de disposar d'un sistema més independent, automàtic, pràctic i barat.

Alguns sistemes han començat a aparèixer, sistemes capaços de fixa l'enfoc de la càmera cap a la direcció de so predominant. Aquest és un bon comença-ment, però de tota manera és encara massa rígid. Si el locutor para de parlar i es desplaça per mostrar alguna cosa en un plafó, la càmera restarà aturada fins que el locutor torni a parlar. A més, la precisió dénfoc dependrà força en la precisio i complexitat de l'array de receptor de so. Caldria també poder tenir en compte el sistema de zoom per a que el sistema sigui una eina flexible.

Un sistema de video-conferència intel·ligent es pot entendre com un sistema capaç d'utilitzar tota la informació de léntorn per controlar-se a si mateix. En un sistema així, és cert que el so dóna molta informació, però com algú digué una vegada: *Una imatge val més que mil paraules*. Per tant, és normal pensar en treure el màxim profit de la font de video.

A la figura podem veure una idea del que podria ser el sistema de visió intel·ligent. La idea no queda limitada a una aplicació de video-conferència. La idea, és aplicable a la robòtica en general i la intel·ligència artificial en el marc de la Visió per ordinador.

El sistema de visió artificial hauria de poder gaudir de totes aquelles característiques que beneficien als humans per a poder-s'hi veure. Caldria que tingués una anàlisi de baix nivell (sense comprensió de contingut) de la imatge, capaç de reaccionar ràpidament en front de canvis o situacions. També caldria que tingués una part capaç d'aportar la part suficient de comprensió de la imatge per poder-ne assegurar un funcionament robust.

La primera part de l'anàlisi correspondria al que sánomena *"Focus of Attention Finding"* (o *Recerca del Centre d'atenció*). Aquesta faria ús de la informació provinent del so, de les regions que són detectades com a regions que es mouen i si hi ha la presència d'algun color important (entre d'altres característiques). Combinant aquestes informacions, podem saber on de la imatge cal fer la major èmfasis per a l'anàlisi. Un cop sabem *on?*, ens caldria saber *què?* per tal de permetre al sistema de decidir què cal fer. Aquí és on entra el concepte de segmentació. Cal una descomposició o divisió per tal d'analitzar la imatge. El nostre interés es centra en aquelles regions que representen objectes amb significat, per tal de ser capaços d'analitzar-los i també d'analitzar l'escena en general.

Els segments de la imatge poden ser trobats a diferents nivells i utilitzant diferents característiques per analitzar la imatge. El Sistema de Visió Humà (HVS) realitza el mateix, utilitza moltes característiques per analitzar-los. Molts científics pensen que el HVS realitza un processament i anàlisi de la imatge pas-baix i pas-

banda. El primer tipus d'anàlisi s'utilitzaria per extreure la estructura de la imatge, i el segon per analitzar els detalls i complementar el primer. Tenint en compte el rendiment i qualitat de l'HVS, no és mala idea provar de prendre'l com a model, i mirar si es possible assemblar-s'hi.

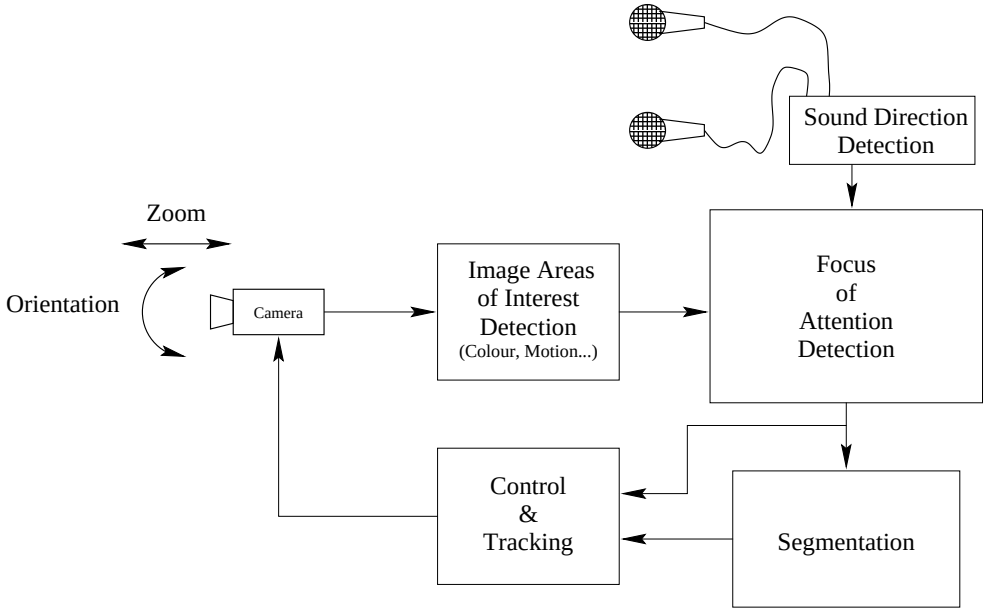


Figure 1: Sistema de Video-conferència Intel·ligent

En aquest treball, s'ha fet èmfasis principalment en dues de les parts del Sistema Multimèdia Intel·ligent. Els principals punts són dos dels que afecten al Sistema Visual.

En Capítol 2, s'ha treballat en un *"front-end"* no adaptat per a lánàlisi de la imatge. Aquest *"front-end"* està basat en una tècnica de segmentació que es recolza en la descomposició de la imatge a través de de l'escala i l'anàlisi de la seva estructura.

L'altre camp de treball ha estat en la part de *"Focus of Attention"*. Aques-ta pot ser trobada al Capítol 3. En aquest, s'ha estudiat les possibilitats de dirigir l'atenció de la càmera. De fet, en podríem dir els actes reflexos de la càmera. L'anàlisi del moviment a baix nivell de l'informació de moviment de la imatge, permet de determinar on cal parar atenció i on cal realment segmentar, simplificant d'aquesta manera la tasca d'anàlisi que realitza la part de segmentació.

Acknowledgements

Abans de res, jo voldria agrair als meus pares la confiança que han tingut en mi i l'esforç per donar-me uns estudis i tot el que m'ha calgut. A més vull agrair al meu pare l'haver-me motivat desde sempre en el camp de la tecnologia.

També voldria donar les gràcies a la Rosa, que ha estat al meu costat en tot moment i m'ha donat un cop de mà quan em feia més falta.

Vull agrair a tots els professors de l'ETSETB que han sabut motivar-me i animar-me en l'estudi d'aquesta carrera. No és una tasca fàcil i no tothom en té el do, per tant a tots aquets professors moltes gràcies. Gràcies Ferran a més, pel teu consell i bona voluntat.

No voldria pas que quedessin sense anomenar tots els amics de Telecomunicacions (Xavi, Xevi, Eduard, Albert, Helena, Víctor, Ferran, Marc, Jaume, Ivan, Lluís, Àlex, Francesc, Dani, Rosa), amb els quals he passat molt bones estones i bons sopars (a veure quan tornem a la *Fonda*), i a més hi he passat mija vida aquests anys d'estudis (esment especial als meus companys de pis Eduard, Francesc, Ferran i l'Alfons, tot un mestre de les "Endos"). I els meus amics d'infància (Èric, Xavi, Javier ...), que tot i que no hi passo tanta estona no els oblido pas.

Je voudrais bien remercier mon directeur de projet Prof. Pierre Vandergheynst pour son conseil et aide en tout moment. Merci Pierre.

Gràcies per la vostra companyia, personatges Lausannoises (Akira, Albert I, Albert II, Dani, Meri, Francesc, Edu, Xavi, Cesar ...). Al Sergi, vingut de terra d'avellaners per posar-se el nom de *Madam Pastis*, home sense igual, no el deixeu pas anar a la mili. A la Rosa un altre cop, ja que per ella mai ni ha prou. Grazie al Gran Scopatore Matteo, sempre pronto con il suo radar, e abile pastasciuttaro. Grazie Riccardo per suoi Xeyes. And thanks to every body in the LTS(EPFL) Lab.

Contents

List of Figures

Resum

En els darrers anys, nous sistemes multimèdia han estat desenvolupats per a videoconferència. Aquests sistemes, cada dia millors, tenen encara el problema d'estar controlats manualment en part o en la seva totalitat.

En aquest treball, es tracta d'introduir el concepte de *Sistema de Videoconferència intel·ligent*, el qual hauria de ser capaç de cercar automàticament el locutor, enfocar-lo i seguir-lo. Aquesta idea requereix un sistema intel·ligent de visió artificial per ordinador, capaç de trobar les àrees o regions d'interés (aquelles on alguna cosa o algú pot demanar de ser enfocat), entendre què hi ha en aquella regió, i actuar en conseqüència.

El nostre estudi es pot estructurar en dos punts principals. L'un correspon a la part de cerca de la regió d'interés (Recerca del Focus d'Atenció) i l'altre correspon a una anàlisi de baix nivell del contingut de la imatge. En primer lloc, hem estudiat una tècnica per detectar regions en moviment basada en un test estadístic χ^2 aplicat a un fluxe òptic dens, obtingut mitjançan l'algoritme de "Lucas & Kanade per Simoncelly". Aquest permetrà de detectar on són les regions on hi transcorre l'acció o hi ha moviment. En segon lloc, l'anàlisi de baix nivell de la imatge és un front de visió no adaptat. Una tècnica de segmentació basada en el que s'anomena *Scale-Space*, un principi que es fonamenta en l'anàlisi de la descomposició de la imatge en diverses versions d'ella mateixa a diferent escala (diferents versions pas-baix de la imatge). D'acord amb el que proposen bastans científics, aixo és, en certa manera, una part del que realitza el sistema visual humà (HVS) per poder-hi veure. A més, s'ha afegit informació pas-banda al procés per mirar de millorar la segmentació.

Abstract

In the last years, many multimedia systems have been developed for video-conferencing. These systems, every day better, have still the problem of being manually controlled, or at least partially.

In the present work, we try to introduce an approach to a Smart Video-conferencing System, which one should be capable to automatically search who is speaking, focus him and track him. This idea, requires an intelligent computer vision system, capable to find the areas or regions of interest (where something or somebody could be suitable to focus in), understand what is in that region, and act consequently.

We have studied two main subjects here. One concerns to the part of finding the region of interest (Focus of attention finding) and the other corresponding to a low level image analysis. In the first one, a moving region detection based on a statistical χ^2 test detection on a dense optical flow generated by "Lucas & Kanade by Simoncelly" has been studied. It will permit to detect where are the regions where is the action, or the movement. In second place, the low level image analysis is an uncommitted visual front-end. A segmentation technique that will look for meaningful object regions. This segmentation technique is based on Scale-Space, a principle that relies in the analysis of the decomposition of the image in several versions of the image itself but at different scales. According to many scientists, this is, in some way, a part of what the Human Visual System (HVS) performs to see. In addition, some band-pass information will be added in the process, to improve the segmentation.

Chapter 1

Introduction

1.1 Situation

In the last years, multimedia applications have improved considerably. The advances in computers, communications and information theory have brought an excellent field to develop complex and integrated solutions to allow people share information and communicate.

Humans live in society, they are a kind of beings that need to deal with others. Since people are wandering around the world, they have always looked for a way to express what is in their mind and to share their ideas. First steps in this were the language, and then the writing. Since that, a long way has been already run. Telegraph, Telephone, Radio-communications, Digital communications... Now, people can communicate all over the World, immediately and fast. Distances are non existent into the cable, the fiber or the air.

Distance between too speakers has been reduced to the limit with video-conferencing. It permits a full audiovisual communication, bringing to those who are in communication the best possible comfort. Video-conferencing allow the existence of an only event, unique in time, in reason, in group of people, but diverse in space. It is not necessary anymore for people to meet in the same place for an enterprise meeting. Everybody can stay in their city office, and carry out a the full discussion without limitations.

1.2 Context

This work is developed in the context of a multiuser video-conferencing system, where in every video-conference terminal there are many people participating. Normally, in each end of the video-conferencing system, there is the necessity of a cameraman and a sound technician. They must take care to focus the camera

and the sound source on the person who is speaking or needs the attention of the camera. A more independent and automatic system would be desirable, cheaper and practical.

Different systems have begun to appear. Systems capable of setting the focus of the camera to the sound source. This is a good beginning, but anyway still too rigid. If the speaker stops talking and moves to show something on a board, the camera will stay still until the speaker talks again. In addition, precision on the focusing will depend much on the precision and complexity of the sound receiver array. The zooming of the system, should be also taken into account to have a flexible tool.

A smart video-conferencing system can be conceived as a system using all the possible information in the environment to have control of itself. In a system like that, of course sound gives a lot of information, but as somebody said once: *An image tells more than a thousand words*. So, it is natural to think in taking much profit from the video source.

In Fig. 1.1 we can get roughly an idea of what such a Smart Vision system could be. The idea is not limited to the application of video-conferencing. It is useful and applicable to the general robotics and artificial intelligence in the part of Computer Vision.

The vision system should benefit from those characteristics that benefit humans in their task of visualizing. It should have a low level (without content understanding) analysis of the image capable to react fast in front of changes or situations. And another part capable to apport the sufficient understanding of the image to ensure a robust operation.

The first part of analysis would correspond to the *Focus of Attention Finding*. This should take profit from the information about the direction of the incoming sound, where moving regions are detected and if there is the presence of any important color (among the main features). Combining this information, we can know where the major emphasis is needed for the content understanding. Once we know *where?*, we should know *what?* in order to be able to decide what the system must do. Here is when we come to the segmentation concept. A decomposition or division is necessary to analyze the image. We are interested in those regions that represent meaningful objects to be able to analyze them and the general scene.

Image segments, can be found at many different levels and using many different features to analyze the image. Human visual system (HVS) performs the same, it uses many features to analyze them. Many researchers think that the HVS performs low-pass and band-pass analysis of the image. The first used to extract the structure, and the second to analyze the details and complement the first one. Taking into account the performance of the HVS, it is not a bad idea trying to take it as a model, and look if the suppositions about it are possible.

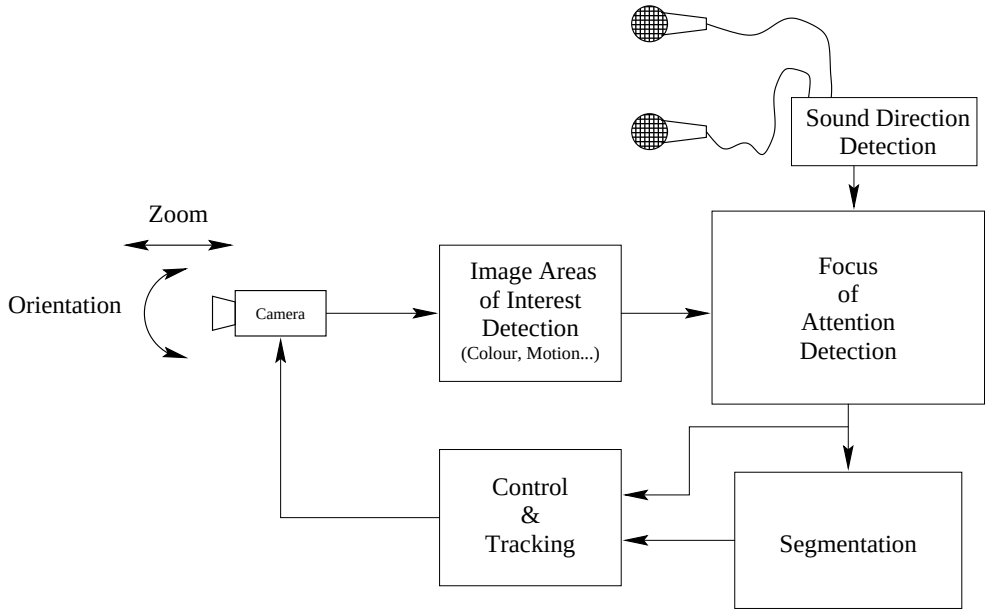


Figure 1.1: Smart Video-conferencing System

1.3 Organization

In the present work, we have focused on two of the parts of the Smart Multimedia System. The main subjects dealt here are two of the ones concerning to the Visual system.

In **Chapter 2**, we have worked on an uncommitted front-end for the analysis of the image. This uncommitted front end is based on a segmentation technique that relies on decomposition of the image through scale and the analysis of its structure.

The other field of work has been in the *Focus of attention* part. It can be found at **Chapter 3**. In this, we have studied the possibilities to manage the attention of the camera. In fact, we could call it the reflex acts of the camera. An analysis of the movement at a very low level of the motion information, permits to determine where the attention is more required and where we must really segment, simplifying in this way the analysis task of the segmentation part.

Chapter 2

Scale-Space Segmentation

2.1 Introduction

One of the most important subfields of Image Processing is image segmentation. It is of crucial importance in visual information analysis in order to make possible the work with images of different natures. The use of different image segmentation techniques can be found in a large number of applications, whose review could take a large number of dissertations.

Recently, engineers have begun to be interested in certain techniques that try to resemble or imitate what they think Nature does. Many of these techniques have demonstrated their efficiency like, among others, genetic algorithms or neural networks. In the artificial vision system presented in this work, we follow what some people think might be an approximation of the Human Visual System, and propose a scale-space based segmentation technique.

Scale-Space permits a segmentation scheme based on a hierarchical representation of the input data. Other hierarchical segmentation algorithm can be found in the literature, like the one presented by *Ziliani and Jensen* [36] or the previous works in the same field by *Burt et al.* [5]. These last examples, basically multifeature segmentation algorithms, try to combine different features in a weighted way in order to correctly link up in the hierarchic structure. In contrast, what is proposed here is trying to rely only on the image intensity structure (is from which the human brain extracts the main amount of information for object extraction). Such structure is obtained as said before from the spread in scale of the original image in gray levels.

Many previous work based on scale-space can be found. Well known dissertations on this subject are from *Lindeberg* [20] [19], which presents scale-space theory and introduces a segmentation algorithm that deals with the properties of extrema points. Unfortunately, this method only permits a coarse blob-like object

segmentation. He extended this work with an edge detection at different scales and this led to an improved version of the technique. Other Extensions dealing with extrema can be found in the work of *Lifshitz* [18] and *Henkel* [13, 14].

In the present work, another way of dealing with the scale-space is used. Instead of relying only with extrema, all the pixels are used to look for the image structure. *Vincken* proposed this idea in [32], where he applied this method to medical images and performed a special variation and extension for $3D$ images.

2.2 Scale-Space Approach

The idea of this segmentation relies on the information that can be extracted from the Scale-Space representation of the original image. In the real world, objects take place in a finite set of scales. This is because details of the same object are visible at some resolutions (not necessarily the same, depending on their size) and a general representation of the object is found at other resolutions (necessarily bigger than the ones where details can be found). The Scale-Space representation is composed by the stack formed of successive versions of the original image at coarser scales. It is supposed then, the bigger the scale, the less information referred to local characteristics of the objects will appear, but the information corresponding to the general object area will remain. Taking that into account, it is reasonable to think that the local and high resolution description of the image at low scales can be related to the general and low resolution contained in upper scales. This relation should permit to link up pixels at one level to the pixel that represents all of them at the immediate superior scale. This relation should lead, at a certain depth of the scale, to the convergence of all pixels that form an object into a single pixel, which is the representation of such an object at that scale.

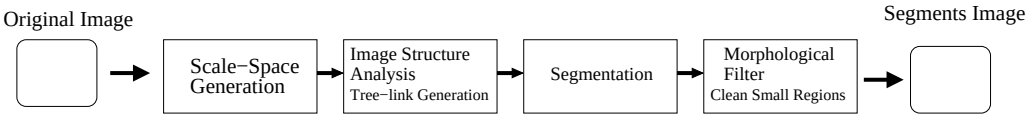


Figure 2.1: Segmentation Schemme.

A scale-space is an ordered set of derived signals/images intended to represent the original signal/image at various levels of scale. This is how *Lindeberg* defines it in [19], where a detailed explanation of linear scale-space can be found.

Each of the signals/images in the multi-resolution stack is associated with a given value which describes the scale level. This parameter in theory could be considered to be discrete or continuous, leading to the possibility of both versions in discrete signals, discrete scale-spaces or continuous scale-spaces of discrete

signals. But since such a thing has to be implementable and computable, we will only deal with the part of the theory that focuses on Linear Discrete Scale-Space of Discrete Signals. *Lindeberg* did in both cases the following assumptions:

- All scale representation should be generated by convolution of the original image with a kernel.
- An increasing value of the scale parameter (the one that describes the scale level) should correspond to coarser levels of scale and signals with less structure.
- All signals should be real-valued functions: $\mathbb{Z} \rightarrow \mathbb{R}$ on the same grid size (theoretically and in ideal conditions a infinite grid), which means that no decimation in coarser resolution levels will be used to obtain a pyramidal representation.

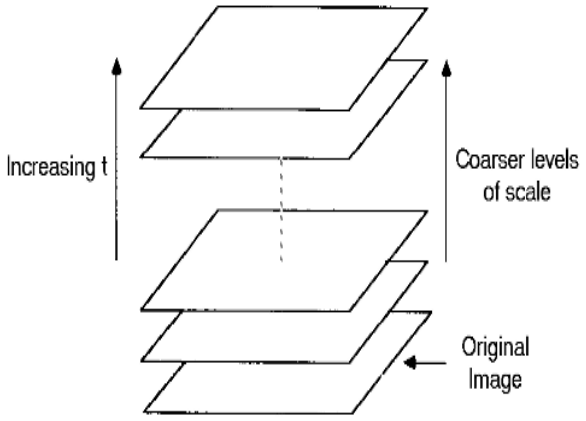


Figure 2.2: Scale-Space stack.

The most interesting characteristic is that the coarser the scale the less structure is present. In fact, no new structure creation would be desired in the multi-resolution stack generation. To obtain these characteristics it has been found that in the continuous case a family of Gaussian kernels is the only solution. In order to check the property of smoothness increment through scale (another manner to say that structure decrements through scale), a good way to measure is looking at the number of local extrema present in each level. There must be not increment when going towards bigger resolutions.

2.2.1 Scale-Space Generation

Techniques Classification

A general comparison and explanation of different scale-space techniques can be found at [32] [12]. Here is a general view of the most relevant techniques. A genuine classification could be:

Linear Scale-Space is a one parameter family of images generated by convolution of the input signal with a Gaussian kernel of increasing width σ. The unique kernel of the linear Scale-Space is the Gaussian. *Koenderink* derived the Gaussian kernel [17], from the linear diffusion equation:

$$\frac{\partial I\left(\vec{x},t\right)}{\partial t} = \Delta I\left(\vec{x},t\right), \tag{2.1}$$

where I is a scalar function (an Image) and t is the scale parameter.

Non-Linear Scale-Spaces has appeared in recent years to bring solutions to drawbacks from the linear approach. These kind of Scale-Spaces relax the constraint of having to process all the information in the same way, but keeps the main properties of scale-space. We can find:

1. **Luminance conserving scale-spaces** are those which preserve the luminance relations between points over scale. In them, can be established relations through levels based on their grey value. Linear Scale-Space is a conserving scale-space, but many others, like the present ones can be found. Among them there are:
   - *Gradient dependent diffusion* [27].
   - *Tensor dependent diffusion* [33].
2. **Geometric flows** are an approach to nonlinear scale-spaces where the evolution of *curves and surfaces* are considered as a function of their geometry. In images, curves are the isophotes, i.e. curves of points of equal luminance [26].
3. **Morphological scale-spaces** In mathematical morphology a scale-space is represented by the stack of images eroded or dilated with a structuring element of increasing size. It has been shown that dilations and erosions with a convex structuring element satisfy the differential semi-group property, and as a particular case, that normal motion flow is formally equivalent to erosion with a circular element [15].

The use of one technique or another depends on the application. Each of them has a particular quality and in front of the necessities is more appropriated or not.

Non-linear Scale-Spaces are a flexible framework to tune the evolution process to the task. When using them, the objective is to take profit from some special characteristic from the original image. That means that we know a priory which kind of image we are going to find (i. e. all kind of medical images).

Contrary, if we do not know what kind of image or situation we are going to find, we can not impose any constraint. We can not select any kind of aperture to generate Scale-Space because we are interested in a particular advantage. In that situation, we should use a uncommitted visual front-end, whose axioms or postulates are:

- linearity (no knowledge, no model, no memory)
- spatial shift invariance (no preferred location)
- isotropy (no preferred orientation)
- scale invariance (no preferred size, or scale)

Since no particularization or emphasis in any feature (at the moment) can be done, **Linear Scale-Space** will be used to simplify the image through scale. Afterwards, on the basis of some criteria, relations will be established over scale.

2.2.2 Linear Scale-Space

Linear Scale-Space, as said before, is the solution to generate Scale-Space with no previous knowledge about the image.

The basic assumptions of *Lindeberg* [19] ask for a kernel to convolve with the original image. Many approaches have been done to find an appropriate solution.

It was first realized by *Koenderink* [17] that the generating equation of a linear scale-space is the linear *diffusion equation*:

$$\begin{aligned} \tfrac{\partial I(\vec{x},t)}{\partial t} &= \vec{\nabla} \cdot \vec{\nabla} I\left(\vec{x},t\right) = \Delta I\left(\vec{x},t\right) = \\ &= \tfrac{\partial^2}{\partial_{xx}} I\left(\vec{x},t\right) + \tfrac{\partial^2}{\partial_{yy}} I\left(\vec{x},t\right), \end{aligned} \tag{2.2}$$

(for $2D$, stating that the scale derivative equals the divergence of the gradient of the luminance function, which is the Laplacian, the sum of the second partial derivatives.

The Gaussian is the Green function of the diffusion equation (see app. A). For an infinite domain, the general solution to Eq. 2.2 is:

$$I(\vec{x},t) = \int_D G(\vec{x}',\vec{x},t) \cdot I(\vec{x}',0), \tag{2.3}$$

where G is the Green function of Eq. 2.2 (so, G is a Gaussian):

$$G(x, y, t) = \frac{1}{2\pi t} e^{\left(-\frac{x^2+y^2}{4t}\right)}. \tag{2.4}$$

Note that the t in the diffusion equation as in the Gaussian has the dimension of the *squared* spatial dimension. The relation to the standard deviation is: $\sigma^2 = 2t$.

Some interesting properties of this representation are:

- Causality: coarser scales can only be caused by what happened at finer scales.
- Maximum principle: at any scale change the maximal luminance at coarser scale is always lower then the maximum intensity at the finer scale, the minimum is always larger.
- No new extrema at larger scales: this holds **only** for one-dimensional signals [19].
- Physics of luminance diffusion: the decay of the luminance with scale is equal to the divergence of a flow.

All the previous formulations, have the problem of being done in the continuous domain. Since the only way to work and compute is in the discrete domain, it is sure that all the assumptions will not be necessarily true. In addition, another problem appears: in computer vision these ideas are impossible to apply to an infinite image, which may lead to finite size effects.

Gaussian derivatives and regularization

The Gaussian kernel is now established as the unique scale-space operator to change scale. There is an important additional result: one of the most useful results in linear scale-space theory is that the spatial derivatives of the Gaussian are solutions of the diffusion equation too, and together with the zeroth order Gaussian they form a complete family of differential operators.

We can see:

$$\frac{\partial}{\partial x}(I * G) = I * \frac{\partial G}{\partial x}. \tag{2.5}$$

This means that the derivative is given at a given scale: this operator is called *scaled derivative operator or Gaussian derivative operator*. Differentiation and

scaling are intrinsically connected: it is impossible to differentiate discrete data without increasing the inner scale, i.e. blurring a little. This is consequence of a property of the family of differential operators from Eq. above: *regularization* of the differentiation.

Gaussian derivatives, are also a solution of Eq. (2.2). This introduces that derivative analysis can be performed at different scales, and if with the fundamental solution (Gaussian) we can establish the basis of the image structure, with the remaining derivatives we can obtain description of the original image helping to follow up the structure.

Discrete Scale-Space Approximation

Tony Lindeberg wrote a complete dissertation about Scale-Space, its fundamentals and possibilities [21]. As said before, he postulates that the Linear Scale-Space should be generated by convolution with a one-parameter family of kernels, i.e. $L(\vec{x}, 0) = I(\vec{x})$ and for $t > 0$.

$$L(\vec{x}, t) = \sum_{\vec{x}' \in \mathcal{Z}^N} G(\vec{x}', t) I(\vec{x} - \vec{x}'). \tag{2.6}$$

He defines the Scale-Space Family of Kernels $G : \mathbb{Z}^N \times \mathbb{R}_+ \rightarrow \mathbb{R}$ as satisfying:

- $G(\cdot, 0) = \delta(\cdot)$,
- *the semi-group property* $G(\cdot, s) * G(\cdot, t) = G(\cdot, s + t)$,
- *the symmetry properties* and
- *the continuity requirement* $||G(\cdot, t) - \delta(\cdot)||_1 \rightarrow 0$ *when* $t \downarrow 0$.

and a Scale-Space representation:
Let $I : \mathbb{Z}^N \rightarrow \mathbb{R}$ *be a discrete signal and let* $G : \mathbb{Z}^N \times \mathbb{R}_+ \rightarrow \mathbb{R}$ *be a scale-space family of kernels. Then, the one-parameter family of signals* $L : \mathbb{Z}^N \times \mathbb{R} \rightarrow \mathbb{R}$ *given by Eq. 2.6 is said to be the scale-space representation of* I *generated by* G.

If we want to approximate the discrete Scale-Space kernel, by a discrete Gaussian kernel, first of all we find two main problems.

1. Gaussian is defined over an infinite domain, which means that for a practical implementation a truncated version of it will be necessary.
2. Discrete Gaussian approximation differs much from the continuous Gaussian function at lower scales.

Discrete Gaussian kernel, is the fundamental solution for the discrete diffusion equation. Unfortunately, it lacks of the property of the possibility to generate one level of the scale-space from the level below with an iterative kernel. Because of that, the property of *semi-group* is failing. That is one of the reasons that impulsed *Lindeberg* to work directly on a kernel approximating the discrete version of the diffusion equation [19].

In this work we will make use of a discrete Gaussian for simplicity and we will tune our kernel to introduce the less possible distortion.

2.3 Scale-Space Application

2.3.1 Scale-Space Generation

Several manners of generating the multi-resolution stack have been taken into account. This is because of the great importance of this step into the segmentation process. The quality of the posterior segmentation will be influenced by its accuracy

Gaussian Blurring

As it has been said before, one of the most common and used way to generate the Scale-Space is the convolution of the original image with a Gaussian kernel, whose width determines the resolution of the resultant level in scale. In addition, in [17, 10] it is shown that the only linear scale-space constructor is the Gaussian function. It can be formulated in the continuous domain as a vectorial isotropic Gaussian function:

$$g(\vec{x},\sigma) = \frac{1}{(\sigma\sqrt{2\pi})^d} e^{(-\frac{||\vec{x}||^2}{2\sigma^2})}, \tag{2.7}$$

where d denotes the dimension of the input domain (in our case $d = 2$), and σ will determine the blurring in the resultant image.

Since we are working in the discrete domain, the discrete version of Eq. 2.7 is:

$$g(n,m,\sigma) = \frac{1}{2\pi\sigma^2} e^{(-\frac{n^2+m^2}{2\sigma^2})}. \tag{2.8}$$

The kernel can not be infinite, and it has to be once more approximated by a truncated version. Defining the Gaussian matrix as a NxN matrix, if we take

$N = 2K_d\sigma$ (or equivalently truncating the Gaussian at radius $r_g = K_d\sigma$), K_d defines the accuracy factor used in the approximation. In the implementation the desired factor will be to take at least $K_n = 3$.

Performing convolution in the **Spatial Domain** has its advantages and disadvantages. In one hand there is the possibility to easily set the width of the Gaussian in order to keep the desired accuracy of the Gaussian kernel. In addition it is possible to control exactly the region which is going to be affected by the border effect (as it has been said in section 2.2, ideally the image should be unlimited, in fact the theory is based on this supposition). But in the other hand, there is the big problem of convolution in spatial domain: computing time. Although some fast computing can be implemented thanks to the separable nature of the Gaussian (see [32]).

In the **Fourier Domain**, convolution turns into a simple product between both transformed functions, furthermore speed is higher than in spatial convolution for big kernels, and in some way boundary problems "get solved" by the implicit periodization of the image in the discrete Fourier domain. Unfortunately, one problem arises. It is not sure, according to the literature [32, 18], that periodization is the best solution to the boundary problem, in fact it has been qualified of including "undesirable artifacts" at large scales. Another variation would be also the introduction of mirroring instead of the simple periodization, and although it has also been qualified of introducing "undesirable artifacts" it would eliminate the discontinuity in the image boundaries. The problem with both solutions is that a spatial aliasing is introduced and it could introduce distortion in structure at large scales, when the Gaussian kernel grows enough.

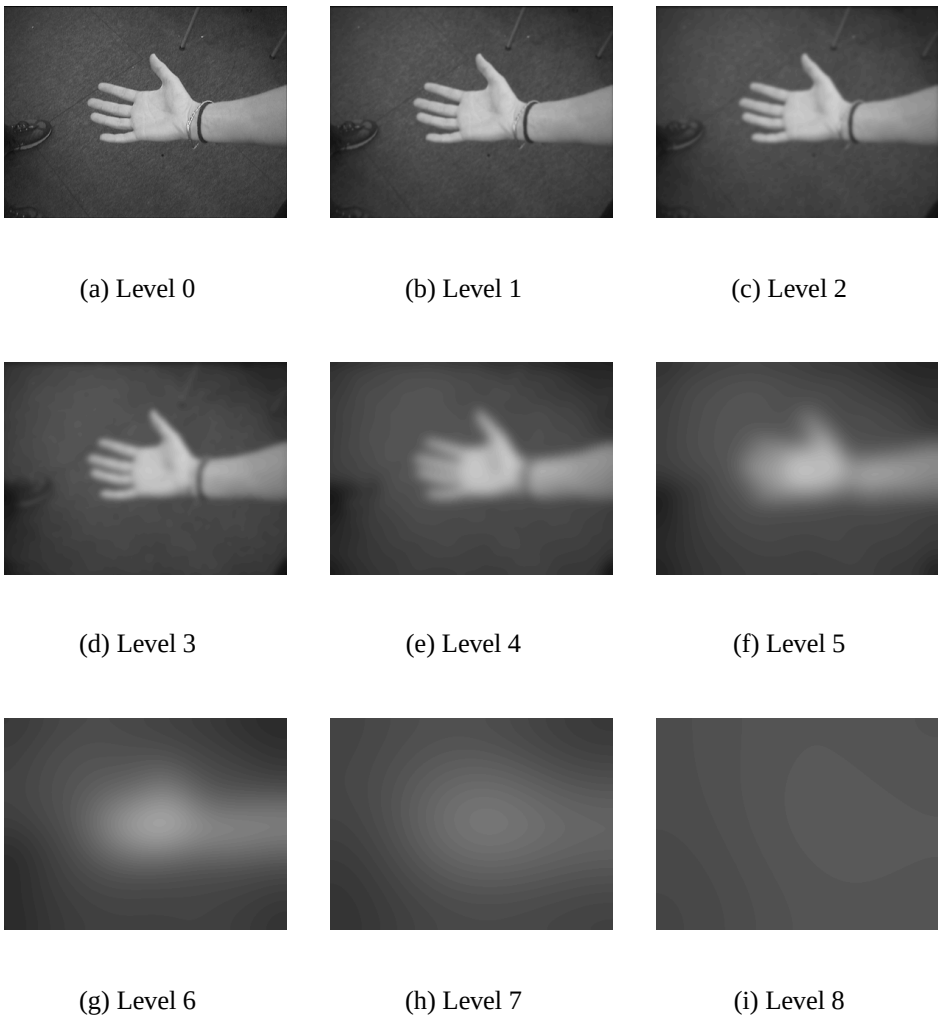
The expression for the Gaussian kernel in the discrete Fourier domain, obtained of applying the DFT to (2.8), is simply:

$$G(k,l,\sigma) = e^{(-(\left(\frac{k}{N}\right)^2+\left(\frac{l}{M}\right)^2)2\pi^2\sigma^2)} \tag{2.9}$$

where N and M specifies the Fourier transform dimensions.

Scale Sampling

Ideally, in order to be able to correctly follow the Scale-Space up, it would be great to dispose of a continuous version of it. But again, since it has to be computable, only discrete set of scale-space slides will be available. The final segmentation result will be strongly dependent on the quality of the multi-resolution stack. That quality refers to the selected slides density of the scale-space, in such a way



(a) Level 0 (b) Level 1 (c) Level 2

(d) Level 3 (e) Level 4 (f) Level 5

(g) Level 6 (h) Level 7 (i) Level 8

Figure 2.3: Scale-Space representation by Gaussian blurring of image (a) with 1 sample per octave in scale.

that, if not enough slides are taken, it will not be possible for the linking to follow back correctly the scale-space structure, leading to wrong and undesired effects in the segmentation, as over-segmentation in some regions and under-segmentation in some others.

An adequate sampling rate should be found. As in sec. 2.2 the standard deviation of the Gaussian σ_n is shown to be the determining parameter for the scale level. This is, in consequence, the value that will be discrete to obtain the selection of scale levels.

To obtain a uniform sampling in the scale direction, and relate it linearly with

a parameter δ_τ, the std. deviation σ_n of the Gaussian must have the form:

$$\sigma_n = \varepsilon e^{(\tau_0 + n \cdot \delta_\tau)} \tag{2.10}$$

where ε is a parameter which specifies the initial scale for $n = 0$, τ_0 is just a possible offset of the first level in Scale-Space, and δ_τ specifies the scale sampling.

It is shown that pyramids accomplice the rule:

$$N_{n+1} = 2^{-d} N_n \tag{2.11}$$

so:

$$N_n = 2^{-nd} \tag{2.12}$$

where N specifies the side dimension of a pyramid level.

To get a pyramid from the values of (2.10) one must necessarily set $\delta_\tau = d \ln 2$. This yields $\delta_\tau = \ln 2$ for the octave spacing, but this is too coarse to allow the successfully convergence of the pixels in their regions. It has been found experimentally that $\delta_\tau = \frac{\ln 2}{4}$ is a good compromise in the present application, *Vincken* in [32] applies and recommends a factor $\delta_\tau = \frac{\ln 2}{2}$ in $3D$ medical applications, but in this work images are not composed of such uniform surfaces. Consequently, since each object to segment includes more heterogeneous surfaces, more levels in scale space are needed to follow better the structure of the image. Nevertheless, common sense recommends to sample as fine as possible (in the end that will be reflected in the computing time needed) let the linking procedure follow the adequate paths in scale.

2.3.2 Linking Levels

Once the multi-resolution stack is available, the linking between pixels of different levels must be carried out. It is the most critical step in the segmentation process. An incorrect linking will lead to a wrong segmentation, merging areas which should not merge or split them. The decreasing information in the scale-space stack gives rise to a graph-like structure of singularities and catastrophes. The structure of this tree may be exploited to link pixels according to certain criteria to create segments.

In the following application, the idea found in [32] has been applied. It enables the hierarchical analysis of image structure: the number of required segments determines at what level in scale-space the down projection of the tree should start.

The main idea in the linking is to follow the iso-intensity paths through Scale-Space. The linking process is performed in a bottom-up way. It creates linkages

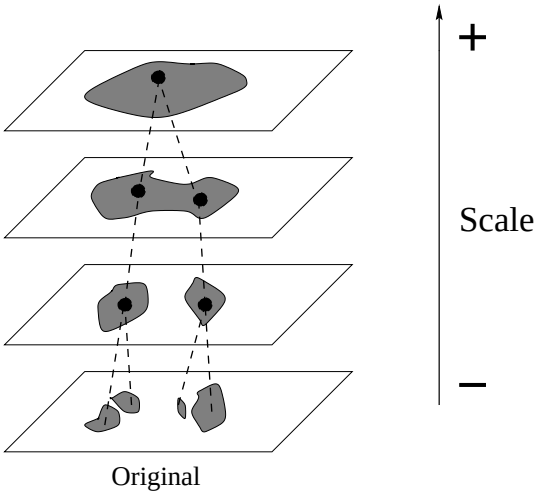


Figure 2.4: Hierarchical analysis of the image structure linking pixels through levels.

between pixels in two adjacent levels (n and $n+1$). Beginning from the lowest level in the stack (the one with more information), we search in the following level for the most suitable pixel parent (according to some criteria). This search for the most suitable parent from the next level above is realized recursively from the 1st level to the top of the stack.

Only those pixels that have at least a child will be linked up again. In this way, and thanks to blurring, a convergence is ensured on the top level in one pixel (only if we have represented the image deep enough into scale).

The search area

When looking for a parent pixel, it is sure that an exhaustive search in the next level is not necessary. We are trying to follow the iso-intensity paths. In theory, these are continuous in a continuous Scale-Space, that means that in the continuous case it would be only a differential displacement around the child pixel. In our reality, we must work on a discrete Scale-Space, with a scale sampling. That means that the continuation of the iso-intensity path in the next level will be farther than just a differential. Anyway, since we know the distance between scale levels, we know which is the relative diffusion from one level to next one. Consequently, the search can be reduced to a certain area, the radius $r_{n,n+1}$ of which (since the blurring is isotropic, a circular search window must be considered) will be related to the relative scale difference $\sigma_{n,n+1}$ between levels n and $n+1$.

Radius $r_{n,n+1}$ is defined as:

$$r_{n,n+1} = k_n \cdot \sigma_{n,n+1}, \qquad k_n \geq k_{n,min}. \tag{2.13}$$

where k_n (the value can depend on the level, but here is taken constant) is a constant which determines the compromise between accuracy and computational complexity.

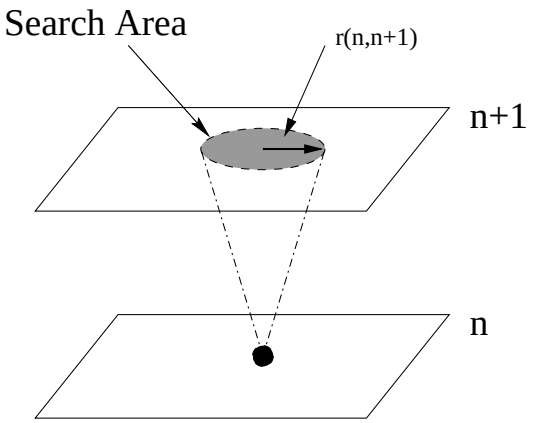


Figure 2.5: Search area for a parent pixel.

k_n must be at least such that $r_{n,n+1}$ is as big as the inner scale of the parent pixels level:

$$k_{n,min} \cdot \sigma_{n,n+1} = \sigma_{n+1}. \tag{2.14}$$

An influence factor γ (such that $k_n \leq \sqrt{-2\ln(\gamma)}, 0 \leq \gamma \leq \gamma_{max}$) can be defined and a relation between γ and k_n is established. Fig. 2.6 shows the relation between them.

Depending on the scale sampling, a maximum value for γ is necessary: $\gamma_{max} = e^{(-\frac{k^2_{n,min}}{2})}$.

Linkage criteria

In the task of searching a parent, the selection has to be done according to some criteria. The main objective is to find the nearest pixel with the nearest gray level. Apart from that, and in order to help and increase the robustness of the selection, some other characteristics can be added.

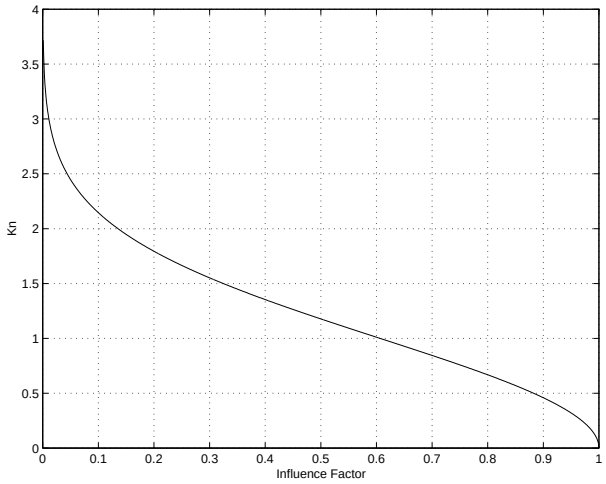


Figure 2.6: Relation between γ and k_n

Gray Level Difference is the main component and the most important. In an ideal situation should be the only one necessary to follow. The problems arise when appears the distortion due to approximations. The normalized factor is:

$$\mathcal{C}_{\mathcal{I}} = 1 - \frac{|I_p - I_c|}{\Delta I_{max}}, \tag{2.15}$$

where I_p and I_c are the pixel intensity from the parent and the child respectively, ΔI_max is the maximum intensity in the original image.

Ground Surface is the normalized number of pixels from the original image that the parent represents. So a pixel will be more likely to be related with a parent with many children than with another with many few. The utility of this factor is to avoid pixels that keep linking with themselves through scale due to scale-space approximation imperfections. It acts like a kind of gravity. The normalized factor is:

$$\mathcal{C}_{\mathcal{G}} = \frac{SG_p}{SG_{max}}, \tag{2.16}$$

where SG_p is the parent Ground Surface (or number of pixel from the ground level he represents) and SG_{max} is the max Ground Surface found for a parent in the same parent's level.

Ground Surface Mean intensity is the normalized difference between the mean intensity value of the pixels from the original image represented by the par-

ent and the ones represented by the child. The normalized factor is:

$$\mathcal{C}_{\mathcal{M}} = 1 - \frac{|M_p - M_c|}{\Delta I_{max}}, \tag{2.17}$$

where M_p and M_c are the mean for the parent and the child respectively. This is another auxiliary characteristic to help the linking up. If a parent and a child have a very similar mean is natural to merge.

The two last characteristics will have to be included in an iterative version of the affinity computation between a parent and a children. In fact, the first time we try to find the links between two levels, no knowledge about Ground Surfaces and Ground pixels representation is known for the parents' level. So, after a first initialization of links computed just with the gray level difference, it will be calculated an arbitrary number of times with the additional parameters till stability of the result.

To compute the general affinity value, a pondered summation of the parameters is done. In addition, the hole pondered summation, is also pondered by a factor that depends on the distance between possible parent and the child. The further, the less probable to be related both pixels.

$$\mathcal{A} = \mathcal{D} \cdot \frac{\sum_{i=1}^{N} w_i \cdot \mathcal{C}_i}{\sum_{i=1}^{N} w_i}, \qquad with\, \mathcal{C}_i \in [0,1], \tag{2.18}$$

where $\mathcal{D}$ is the distance factor, and w_i are pondering values.

Since Affinity decreases with the distance, and Gaussian blurring is being used, a based Gaussian shaped function is employed to ponder affinity between pixels according to their distance.

$$D(d_{p,c}) = e^{\left(-\frac{d_{c,p}^2}{2\cdot(\sigma_p^2 - \sigma_c^2)}\right)}, \tag{2.19}$$

where $d_{c,p} = ||\vec{x}_p - \vec{x}_c||$, and σ_p and σ_c are the scale of the parents level and the children level respectively. And, because of in each level there is a certain inner scale, into this, all the pixels are equally possible candidates to be parent, so the following modification is done:

$$\mathcal{D} = \begin{cases} 1 & \text{if } d_{c,p} \leq 0.5\sigma_p \\ \frac{D(d_{c,p})}{D(0.5\sigma_p)} & \text{if } d_{c,p} > 0.5\sigma_p \end{cases} \tag{2.20}$$

Object Contour Supervised Linking

In addition to the presented criteria, an other factor has been added to reduce incorrect linking of pixels in regions close to edges. Since human visual system is known to work as well with band-pass representation of images, edge detection has been added to help into the linking up process.

Objects are delimited by edges. Segmentation in scale-space tries to find objects, or at least meaningful regions. Because of that, linking of pixels from a region to an other must be avoided, otherwise it will lead to incorrect segmentation. To avoid that in case of failure of the correct following of the iso-intensity path through scale, a simply rule can be used. If when computing the affinity between two pixels the link goes through an edge, it will be not taken into account when comparing to find the best parent.

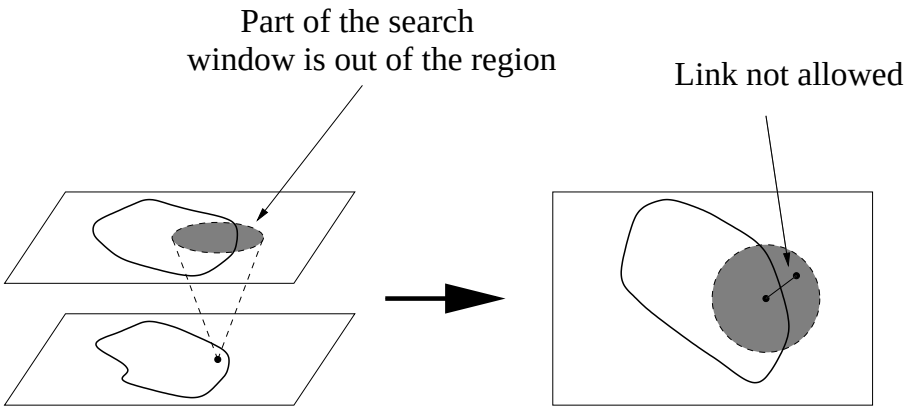


Figure 2.7: Wrong linkage problem.

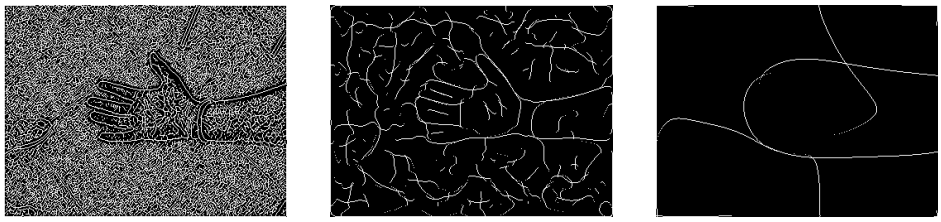
In parallel to the scale-space representation of the image, an edge representation through scale can also be performed. In the scale-space representation, image structure is kept through scale. Edges can also be detected through scale, and a relation between levels could be established. In fact, only the most representative edges will be kept when increasing scale. With the edge information in every scale level, we will find the edges of the uniform areas. Those areas should correspond to the information relative to the contained image details at that scale level.

Edge detection through scale relies on the application of Gaussian derivatives (see sec. 2.2.2). Two different approaches have been studied. The use of the first derivative (spatial gradient) and the second derivative (the Laplacian of the Gaussian).

First derivative is the spatial gradient of the scale-space levels extracted in sec. 2.3.1. All the existent ridges in all the levels are extracted. There is no selection of the most important edges, since these will persist through scale. The module of the gradient is performed and an estimation of the ridges is done. A morphological procedure using a directional dilation with reconstruction is used (see app. G).

$$|\vec{\nabla} I(\vec{r},t)| = \sqrt{\left(\frac{\partial I(\vec{r},t)}{\partial x}\right)^2 + \left(\frac{\partial I(\vec{r},t)}{\partial y}\right)^2}, \tag{2.21}$$

where t represents the scale.



(a) Level 1 (b) Level 4 (c) Level 7

Figure 2.8: Edge representation through scale using the gradient. 1 sample per 3 octaves (first sample on the first octave)

Second derivative is the Laplacian of the gaussian of the scale-space levels. Instead of using directly the second derivative, an approximation is used. We use the Difference of Gaussians (DOG) approximation, which has been recently proved to be the response of the receptive field in the cats retina [6]. To detect edges in scale, the difference between two consecutive levels of the scale-space is computed and then, a zero-crossing detection is performed.

$$\mathcal{DOG}(x) = A_1 e^{\left(-\frac{x-\mu}{2\sigma_1}^2\right)} - A_2 e^{\left(-\frac{x-\mu}{2\sigma_2}^2\right)}, \qquad \sigma_1 > \sigma_2. \tag{2.22}$$

I.e., being two Gaussians A and B of scale 6 and 5 octaves respectively, the $\mathcal{DOG}$ will be the equivalent to the second derivative of a gaussian of scale 5.5 octaves.

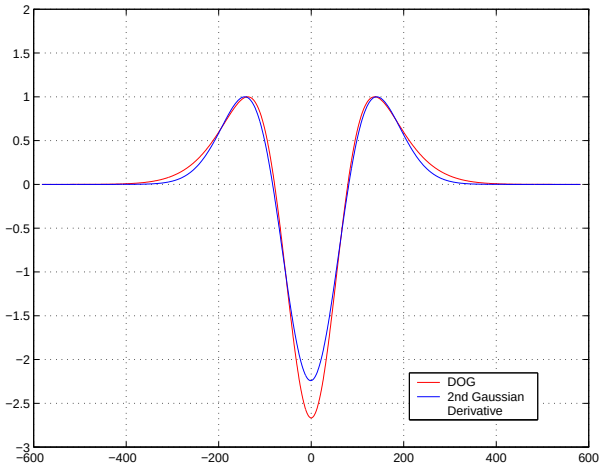
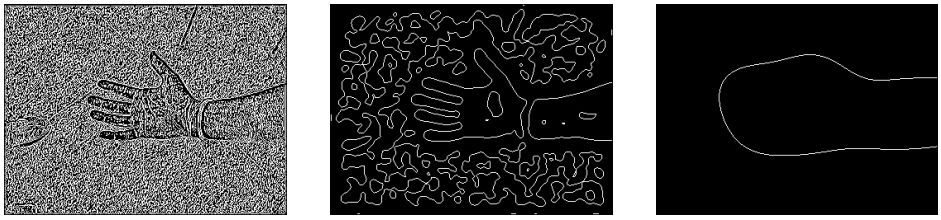


Figure 2.9: Comparison between the Laplacian of a Gaussian and the DOG approximation



(a) Level 1 (b) Level 4 (c) Level 7

Figure 2.10: Edge representation through scale using DOG. 1 sample per 3 octaves (first sample on the first octave)

In the images from Fig. 2.10 and 2.8, we can see how edges of the most important structure are kept through scale.

If the contrary is not said, we will use the Laplacian (DOG approximation) to supervise the linking process, instead of the gradient. The obtained are much cleaner.

2.3.3 Segmentation

Once the linking is done, the estimation of image structure is ready to be interpreted. All the pixels in the ground level are linked up in the tree structure. The only remaining step is to obtain the image segments. The basic idea is that to generate a segment, a pixel is taken in one of the scale levels, afterwards the tree of

links is followed down till the ground level and all the pixels are assigned to the segment.

The segmentation part is mainly composed by two steps:

Root Labelling is the part where the pixels in the scale levels are chosen to represent the segments. In the ideal case, these segments should represent meaningful objects. To select the pixels, two criteria can be taken into account:

1. The selection of the desired biggest scale level, implies automatically that all the parent pixels comprised into that level are labelled as Root pixels. These pixels are supposed to represent the segments of a certain size related to the inner scale of the selected level.

   The selection of a scale level as a beginning for the Downward Projection is necessary. If we are interested in finding objects of determinate size, a higher level selection will merge them and a lower level selection will give an over-segmented representation.

2. Normally, in segmentation, a wide rage of region sizes is desired to recognize. So in addition to the exposed in the last point, some kind of Seeding Rule should be used. It would determine which pixels from the levels below the selected could be also labelled as Root pixels. And in this way permit the distinction of the smaller significant objects.

   To carry out this task in a simple way, a threshold could be imposed to the result of Eq. (2.18). Then, if the nearest parent is over the threshold, do not link it and label it as Root.

In the present work, we have not included a Seeding rule, since we are only interested in general on a narrow range of object sizes. Only the initial level will be selected.

Downward Projection This is the last step. It remains tto follow the image structure down from the selected Root pixels at the different levels of scale. Each Root pixel will represent a segment, and all the pixels from the ground level connected to it will be marked as belonging to that segment.

2.4 Results

2.4.1 Scale-Space Generation Details

The step of Generation, is determinant in the quality of the segmentation. We are approximating a principle, valid for the continuous domain, in the Discrete

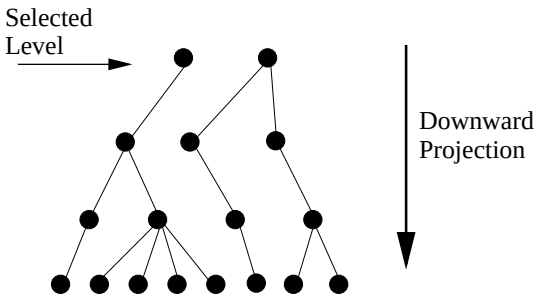
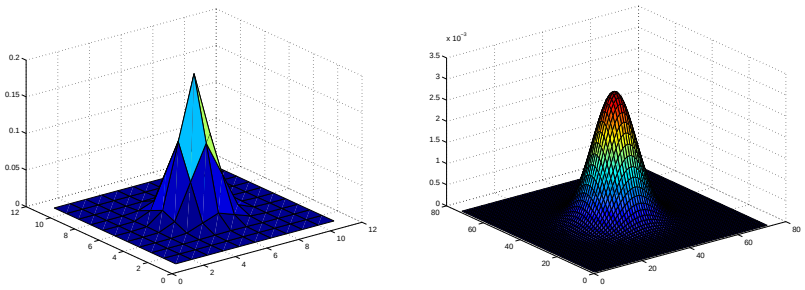


Figure 2.11: Scanning of the image structure to obtain the segments

domain. The main problems that we have found are in terms of the approximation of the Gaussian that generates the Scale-Space.



(a) Discrete Gaussian kernel to generate the 1 octave distance image.

(b) Discrete Gaussian kernel to generate the 7.25 octave distance image.

Figure 2.12: The problem of the approximation of the continuum with discrete.

In the spatial domain, as we see in Fig. 2.12 (a), for small scales, we find that the approximation of the Gaussian differs much from a continuous one. The sampling rate is too slow compared to the kernel size, which implies aliasing and kernel distortion.

In Fig. 2.12 it is seen that for bigger scales the approximation is much more good.

In prevision of the boundary effect, and in order to speed-up the filtering step, we have chosen to perform it in the Fourier domain. In the literature we find the possibilities of mirroring the image, or simply the implicitly periodization of the Discrete Fourier domain. We find as well the idea of just padding all around the image with the mean value of this. Since at large scales (when the kernel goes out from the image) the values of the image tent to the mean.

The real problem we have found, is the frequency resolution. Since we are using the discrete transform, we are applying the same value to a whole band. In addition, in high scale filtering, we find the approximation problem of the continuous kernel by the discrete kernel. Again, the sampling of the Gaussian kernel is too low and loses similarity with the continuous. That is reflected as spatial aliasing of the Gaussian, also known as the boundary problem.

To obtain an accurate result, since we have been working with small images (115x153), we had to use bigger sizes for the FFT. We found in the 512x512 transform to be adequate to perform the tests. The rest of the square outside the image has been padded with the mean of the image (as interpolation) or a mirroring of the image. In the case of mirroring the FFT has been performed on the size of the mirrored image.

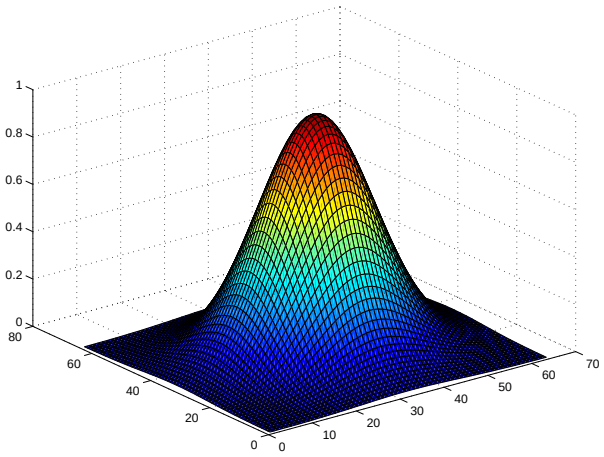


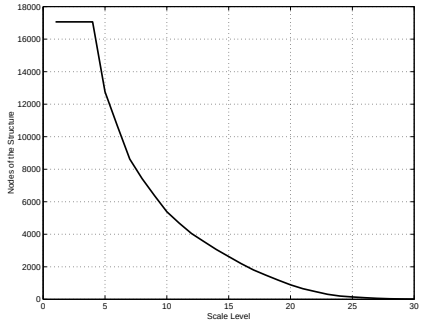
Figure 2.13: Gaussian kernel in the Fourier Domain for a $\varepsilon = 0.455$ in the scale factor. First used kernel in every Scale-Space Generation.

According to Eq. 2.10, we define $\varepsilon = 0.455$. This is to get the first kernel for the spacing of 1 sample per octave with a σ_n around the value of 2. The Gaussian that corresponds to this characteristics is seen in Fig. 2.13. Notice, the kernel is very near from the border. To avoid distortion in the kernel, we force all the other sampling rates in scale (1,$\frac{1}{2}$,$\frac{1}{4}$,...) to have their first sample in this scale. For that, we define τ_0 as $\tau_0 = \frac{(d-1)\ln(2)}{d}$, where d is the number of slices per scale.

2.4.2 Scale-Space Analysis Details

Once the Scale-Space is generated, the structure of the image must be analyzed. To ensure the analysis, from one level to another, a convergence of the pixels must be present.

In Fig. 2.14, we can see the evolution of the number of parent pixels in every level of the scale. This graph and the followings have been done using the image *Rosa's HAND* (see below).



(a) Number of nodes (parent pixels) in every level for the analysis of the image structure only with $\mathcal{C}_i$ component.

(b) Rosa's HAND

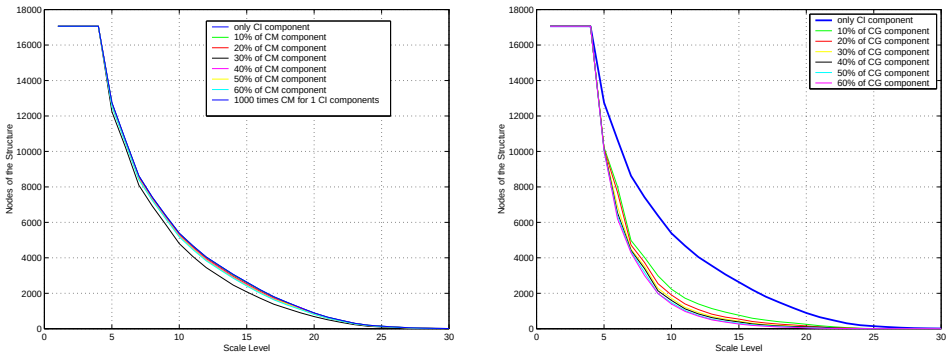
Figure 2.14: Number of nodes (parent pixels) in every level and the image used to generate the values.

First of all, notice that for the first four levels, there are he same number of nodes. This is due to the fact that all the pixels from the image link up with themselves. Non of them converge yet. This is due to the image, is also spatially non continuous, and pixels do not converge till diffusion takes a bigger range. In this case, they do not begin to converge till σ_n is around 4.

In the graph we can also see the fast convergence at the beginning. This is due to the fact that all the small details disappear fastly at the beginning and pixels converge. After, convergence goes slower because regions with components at higher scales disappear more slowly.

In Fig. 2.15 we can see the behavior of the convergence rate, on the same image as Fig. 2.14, applying the modifying factors to help. This help is needed since we have not a continuous scale-space. In fact, if we could work on really high resolution images, the approximation to the continuous scale-space would be better, and also the results using only the $\mathcal{C}_i$ components. Notice that, since it is a discrete scale, many pixels can get lost when trying to follow up the iso-intensity curves through scale.

In (a), component $\mathcal{C}_m$ shows that it does not change much the convergence rate. It helps in merging uniform gray level areas, and splitting those which ar not so uniform. This factor helps to refine some details on the contours of the segmented



(a) Number of nodes (parent pixels) in every level for the analysis of the image structure with $\mathcal{C}_i$ and some % of the $\mathcal{C}_m$ component (gray level difference).

(b) Number of nodes (parent pixels) in every level for the analysis of the image structure with $\mathcal{C}_i$ and some % of the $\mathcal{C}_g$ component (gray level difference).

Figure 2.15: Number of nodes (parent pixels) in every level for $\mathcal{C}_m$ and $\mathcal{C}_g$

regions. It helps also to merge those pixels which are very similar, but do not merge due to the problem of the spatial and scale distance of the sampling. It has also some drawbacks, it can split some regions which should. This component, in some way, can break the tracking of the iso-intensity path, and consequently lead to an undesired over-segmentation.

In (b), component $\mathcal{C}_g$ show that it really changes the convergence rate. It forces pixels to get merged to some parent instead of another if the difference between both components $\mathcal{C}_i$ is not very low, and the importance in area of one parent is much more than the other. This factor is much helping in those cases where due to imperfections of the scale-space, there are pixels that get lost from the iso-intensity curve and keep on linking with themselves through scales, reaching the top level alone, into a region of 1 or 2 pixels. If a to high value is set, it may force the building up of the structure to collapse in one segment too early.

In Fig. 2.16 if compared with 2.14, we see that it does not modify the convergence rate. It is normal if we consider that both characteristics (edges and $\mathcal{C}_i$) have the same scale evolution. Any way, it helps in correcting some mistakes introduced by the component $\mathcal{C}_m$ or $\mathcal{C}_g$, when it merges two unconnected regions that should not due to they are on two different objects.

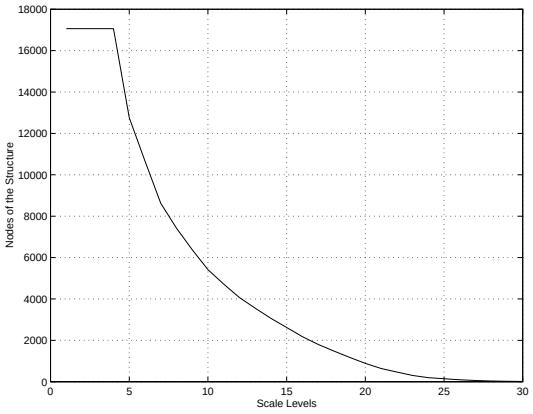


Figure 2.16: Number of nodes (parent pixels) in every level for the analysis of the image structure with $\mathcal{C}_i$ and the application of the edge detection criteria. In this test the edges where computed using the Laplacian of the Gaussian.

2.4.3 Segmentation Results

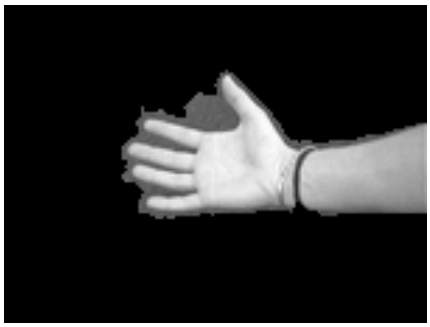
Influence of the secondary features

In the following, we illustrate the effect of the application of the secondary features in the linking process of the structure of the image.

In Fig. 2.17 can be seen the result of the segmentation (a) with using the difference gray level between parent and children only to link. The level to start the downward projection at $\sigma_n = 50$ pixels was selected, in order to not to have regions corresponding to smaller details than the most prominent object in the image, the hand. In fact, as we can see in the right image of the figure (b), we success in obtaining the region corresponding to the hand. Anyway, it is not so accurate (see the region of the fingers), a part of the background is merged with the general area of the hand. In addition, in Fig. 2.17 (a) we see a clear example of the problem due to discreteness in the scale-space. Many pixels get lost in their following of the iso-intensity path, and consequently they reach the last selected level of scale. Leading to appear many small regions that shouldn't be at this level.

In Fig. 2.18, we apply now the influence of the two additional features described in sec. 2.3.2. The selection of valid parents in every scale level continues being exclusive of the gray difference, but in the following iterations of the algorithm, the children get reassigned on basis to the three component weighted addition. The main component is still the gray level difference, and the others try to help in those pixels where the difference is not clearly determining. In the figure, we can appreciate the improvement in both details: the precision of the

(a) Segmentation of the image Rosa's hand, using only the $\mathcal{C}_i$ component and filtered with the morphological filter. Level of segmentation: $\sigma_n = 50$ pixels



(b) Extraction of a meaningful object from the segmentation in (a).

Figure 2.17: Segmentation of Rosa's hand using only $\mathcal{C}_i$.

contour and the disappearance of the little region areas. In (b) the extraction of the hand from the segmentation can be seen.

(a) Segmentation of the image Rosa's hand, using the three components with weights $W_{\mathcal{C}_i} = 1.0$, $W_{\mathcal{C}_m} = 0.4$, $W_{\mathcal{C}_g} = 0.4$ and filtered with the morphological filter. Level of segmentation: $\sigma_n = 50$ pixels

(b) Extraction of a meaningful object from the segmentation in (a).

Figure 2.18: Segmentation of Rosa's hand using the three components.

From that one question arises, which must be the weight of the values? We

have found to give good results for values between 0.1-0.4, without much variation. Normally, we have tried to keep equal the value for the *Ground Mean* component as for the *Ground Surface* component. Since the first tents to split and the second one to merge, in some way strengths get compensated.

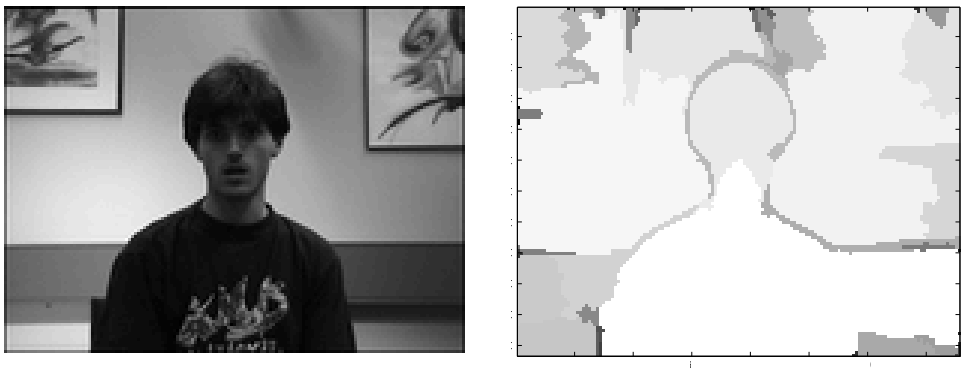
If increasing very much the weight value of *Ground Surface* we obtain that the regions collapse on a single one, if increasing very much the weight value of the *Ground Mean* component the result of Fig. 2.19 is obtained. The result is very satisfactory, due to the fact that when using this high weight value, it performs very well on uniform gray-level surfaces. It can be very useful for example on medical images, where the task there is to segment more or less uniform gray level surfaces. The parents selection continues being based on the $\mathcal{C}_i$ component, but the refinement is completely based on the component $\mathcal{C}_m$.

Figure 2.19: Segmentation of the image Rosa's hand, using two of the three components: $\mathcal{C}_i$ and $\mathcal{C}_m$, with 1.0 and 1000 as weight values respectively.

The Contrast Problem

In this segmentation technique, we have been working relying only on the information available on the intensity image. The main and unique feature is the gray level, with the additional feature of the edge detection. The problem that here arises is that in neighbor regions, although being of different color, they can have the same or similar intensity. That brings the problem of two neighbor regions badly contrasted can merge if only a meaningful object was. It is clear that sooner or later regions merge in the structure, but bad contrast between regions make them merge before they should.

This is the example that can be seen in Fig. 2.20. We can observe, how the algorithm success in segmenting the body, and the head of the subject. Anyway, a part of the wall merges with the body due to the relative low contrast. This is because when building the structure, the most suitable link connects both regions



(a) (b)

Figure 2.20: Segmentation (a) of the image Oscar (b) with parameter weights $W_{\mathcal{C}_i} = 1.0$, $W_{\mathcal{C}_m} = 0.4$, $W_{\mathcal{C}_g} = 0.4$ with edge supervision using the Laplacian version and filtered with the morphological filter. Level of segmentation $\sigma_n = 35$

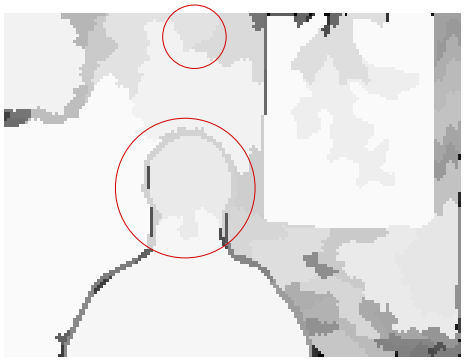
and the edge estimation can do nothing since it is also affected by the low contrast and the edge at the corresponding scale is also not found.

The edges effect

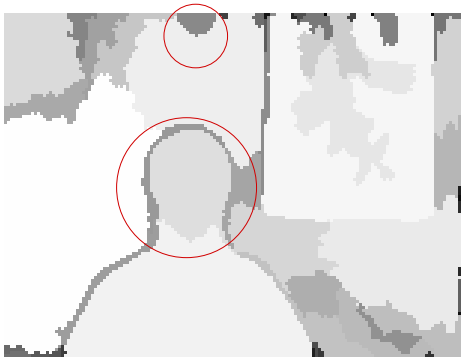
The edge detection is intended to avoid incorrect linking between different regions separated by an edge. In Fig. 2.21 it can be seen the effect of the use or no use of edges. Both segmentations are computed using the same parameters, and are segmented on the basis of the same scale level. The only difference is in the use of edges to supervise the correct linking.

It is easy to see the improvement in the quality of the segmentation. In the images the most relevant details are signaled where the use of edges are more influent. In the figure (b) we see how part of the head is merged to the body, and next to the picture on the wall, there is a little box on the wall, which does not appear defined in the version without edges. In (c), since we use the edges at each scale, we keep from linking through them, and we success in avoiding the incorrect linking of the head, improving the definition of the contours. We can also observe that here the region that defines the box on the world is kept, and not wrongly merged.

(a) Image Sergi.



(b) Segmentation of the image Sergi with parameter weights $W_{\mathcal{C}_i} = 1.0$, $W_{\mathcal{C}_m} = 0.4$, $W_{\mathcal{C}_g} = 0.4$ without edge supervision and filtered with the morphological filter. Level of segmentation: $\sigma_n = 25$ pixels.



(c) Segmentation of the image Sergi with parameter weights $W_{\mathcal{C}_i} = 1.0$, $W_{\mathcal{C}_m} = 0.4$, $W_{\mathcal{C}_g} = 0.4$ with edge supervision using the Laplacian version and filtered with the morphological filter. Level of segmentation: $\sigma_n = 25$ pixels.

Figure 2.21: Comparison of the effect of edge detection on the segmentation.

(a) Image Rosa.

(b) Segmentation of the image Rosa

(c) Extraction of a meaningful object from the segmentation in (b)

(d) Extraction of a meaningful object from the segmentation in (b)

Figure 2.22: Segmentation of the image Rosa with parameter weights $W_{\mathcal{C}_i} = 1.0$, $W_{\mathcal{C}_m} = 0.4$, $W_{\mathcal{C}_g} = 0.4$ with edge using the Laplacian version and filtered with the morphological filter. Level of segmentation: $\sigma_n = 30$ pixels.

More Results

In the Fig. 2.22 we can see how it is segmented the whole head. Notice the detail of the hair, it is well included, all together, with the face in the same object. It is well contrasted with the surrounding regions, and that allows the correct segmentation. The blouse is also correctly segmented. As we can see, the arms are not very well contrasted with their surrounding area. That makes them to be already merged at this scale with the regions of the wall. The supervision same happens with the table. In the Fig. 2.23, an example of the behavior of scale-space segmentation, and the selection of the scale level to perform the segmentation is seen. Depending of the selected scale level, we obtain bigger or smaller segmented objects. In the figure, we see how selecting the scale $\sigma_n = 30$ pixels, the segmentation of the whole Sergi is obtained. In the selection of scale $\sigma_n = 25$ pixels, we hope to find segments of smaller regions corresponding to details of smaller scale. Indeed we find the segments corresponding to the head and the body of Sergi. In addition, the boundaries of the segments are quite well defined, thanks to the edge supervision.

2.5 Conclusions

In this chapter, we have applied the scale-space segmentation to natural images, obtained with a camera from a possible video-conferencing environment. From the obtained results, we see that meaningful objects can be segmented from the scene.

In scale-space segmentation, we have seen, that we can not obtain all the possible segments in an image at the same time. We have to chose between bigger or smaller details. Then, according to the choice, we will obtain the segments of objects that are approximately belonging at that scale. Depending on the application, this can be a problem, or an advantage. In the present one, this can help in discriminate implicitly small details. In this way, the task reserved for the following step in the system (segments analysis for image understanding) will be easier.

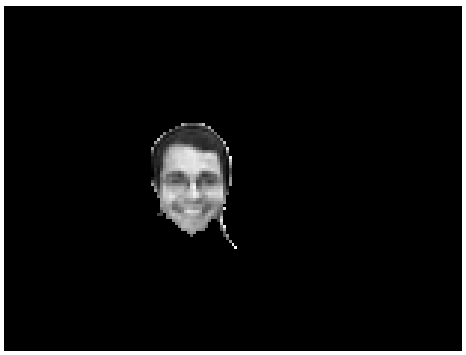
The technique proposed, has shown to be very sensitive to the quality of the generated scale-space. Much care must be taken in its generation, avoiding as much as possible coarse approximations. One of the most important factors in the quality of scale-space has shown to be the frequency resolution of the Fourier transform. The other limitations are the spatial resolution (which limits also the use of finer scale sampling rates), the boundary problems (which are more or less solved with the image mirroring or the image mean padding around), and the scale sampling rate.

(a) Image Sergi.

(b) Segmentation of the image Sergi with parameter weights $W_{\mathcal{C}_i} = 1.0$, $W_{\mathcal{C}_m} = 0.4$, $W_{\mathcal{C}_g} = 0.4$ with edge supervision using the Laplacian version and filtered with the morphological filter. Level of segmentation: $\sigma_n = 30$ pixels.

(c) Segmentation of the image Sergi with parameter weights $W_{\mathcal{C}_i} = 1.0$, $W_{\mathcal{C}_m} = 0.4$, $W_{\mathcal{C}_g} = 0.4$ with edge supervision using the Laplacian version and filtered with the morphological filter. Level of segmentation: $\sigma_n = 25$ pixels.

(d) Extraction of a meaningful object from the segmentation in (b).



(e) Extraction of a meaningful object from the segmentation in (c)

Figure 2.23: Obtention of meaningful objects using the Scale-Space segmentation.

Different aspects have been dealed in the refering to the analysis of the image structure. In what refers to the linking criteria, the difference of gray level between scale levels is used to create the links and find the parents at the upper level. Parents are found relying only in this criteria. The links of children with the found parents, is refined after changing some children to one or other father depending on the criteria. This does not affect the selection of the parents at the level, but it will affect the convergence through scale. If the parameters are well selected, the help in the convergence. The values of the parameters seemed to be independent of the image. Any way, no tests have been performed in different spatial resolutions of the same image to see if the parameters are related with the spatial resolution.

The inclusion of edge detection is the band-pass part of the image analysis in the procedure. As the human eye, we want to integrate in the same procedure low-pass and band-pass processing of the image. Using the low-pass as a basis to extract the structure and the band-pass as correcting feature, has shown to improve the performance.

The tests have shown that the segmentation technique has some problems when two objects or regions not very well contrasted are neighbors. If the segmentation scale is not very well chosen, then the regions can merge. Another problem can be when the optimal scale to begin the segmentation is between two scale samples, and it is not available.

Very complicated scenes, can have segmentation problems. If they are very complicated, then it means that there are many details. Small details are more difficult to segment due to they have less scale levels to converge. In addition, is possible that some other band-pass analysis should be necessary for their correct analysis, or some non-linear scale-space analysis.

We can conclude that it is an interesting segmentation technique for automatic computer vision systems. If we accept the supposition that once the secondary criteria is tuned for a certain image resolution and size, the only parameter to set is the scale level from which we will begin the segmentation. This scale level can be related with the camera zoom and some computed or assumed distance of the object in front.

Chapter 3

Focus of Attention Finding

3.1 Introduction

Human visual system, apart from performing an excellent segmentation of images and object selection, is also improved in such a way that it selects where and at which scale the image analysis must be performed. This improvement is due to different features that allow the visual system to focus its attention on what it is really interested in. Consequently it spends all the effort analyzing the selected area. An analysis of this phenomenon can be found in *Active Vision* in [4, 30]. There are many different stimuli and factors that drive the focus of attention in human beings. These stimuli are from very different natures, they can come from all the possible stimuli that can attract the attention of a Subject. Attention attraction of someone can be due to several reasons, and for several objectives. In the most basic levels of survival, animals can center they attention due to a stimulus because it can represent something good for their development like food, the male or female (for reproduction purposes) or something that puts in danger their existence, like the presence of their predator. In all these situations, the payment of attention could be induced because of some flavor, sound, touch or a visual stimulus (not a detailed analysis of the scene, in fact the purpose of focusing attention is to perform a posteriori a detailed examination). The first three kind of stimulus can contain a lot of information, due to the fact that the brain has other mechanisms to analyze the possible source independently of the visual system, relying on the kind of flavor, a known sound or a characteristic texture on a surface. On the other hand, visual system stimuli are at a lower level, they take effect before any further analysis of the scene. The head turns towards the point or region where some kind of phenomena that induces a stimulus has been observed . Any way, sound can also be considered a low level focus attractor if we only take into account the capability to detect the direction of a sound.

From the point of view of a multimedia system or application like the present one, the most interesting information sources to set and find the right focus of attention may be: Sound and Visual information.

Sound can carry a lot of information about who or what has emitted it, in order to warn the brain of what is the most likely thing that is going to be found by the detailed image analysis. It can be used also at a lower level. In fact a very useful characteristic of the human hearing system is that in reality it is an array of two sensors which can locate with certain precision the origin of a sound source.

Visual information is one of the most relied data by the brain to keep knowledge of the surrounding world. It is the main tool used to realize a fine an precise observation of the environment, but before such a task is performed, other information can be extracted from the image without a detailed examination. Visual system is sensitive to stimuli and in consequence focus of attention is attracted by the presence of determinate colors (colors with special meaning like red, black, yellow... or colors that the brain is looking for), light intensity or motion. All these informations are treated fastest by the visual system, and may control directly actions even before any high level understanding of the scene has been performed.

In the framework of computer vision, a simulation of this biological behavior is very useful in terms of giving the capacity of *Active Vision* to an automatic system. The application of that would bring the possibility of a camera rotating and zooming in order to concentrate its attention on some specific area of the outside world.

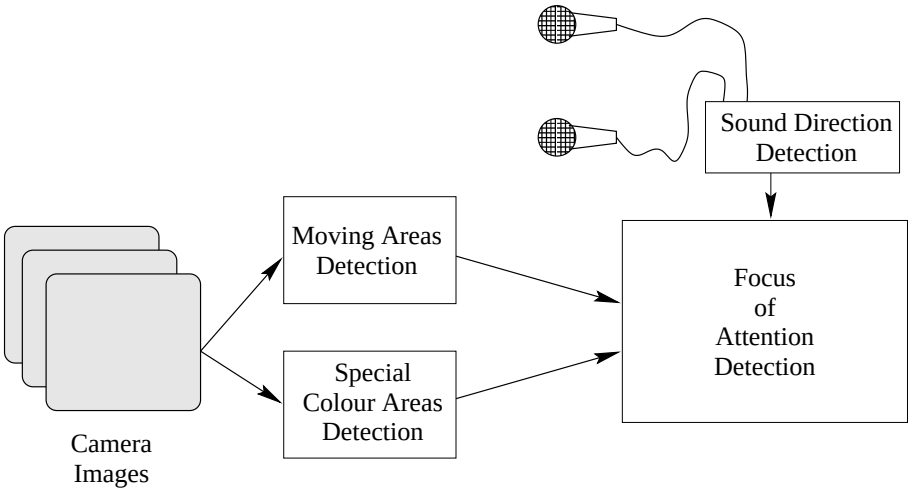


Figure 3.1: Representation of the different factors and possible information that can be used to find the Focus of Attention.

3.2 Motion Source

3.2.1 Motion Estimation

Motion Estimation is a very well known concept in Image Processing. Since images are a $2D$ representation of the $3D$ reality, it is not always possible to get from a sequence of images (a $3D$ representation) the real motion field of an object into a scene (the $4D$ reality). Motion Field is a purely geometric concept, this means that the motion field is the real movement of the structure of the body or object. In sequences indeed, what it is seen, is the Optical Flow, which is the distribution of apparent velocities that can be associated with the variation of brightness in the image. Let us remark the previous expression "apparent velocities". Since they are computed from the variation of gray intensity, they reflect variations due to (among others):

- Variation due to movement of pixels in the image that belong to objects.
- Variation because of changes in the illumination distribution (either in position or intensity).
- Variation because of camera noise.
- Variation due to objects shape.

Although it would be desirable to obtain the motion field, from the images it is only possible to obtain the Optical Flow and hope it to be in good relation with the motion field.

$$\underbrace{(X',Y',Z') \rightarrow (x',y')|_{t-\Delta t} \qquad (X,Y,Z) \rightarrow (x,y)|_{t}}$$
$$\downarrow$$
$$I(x,y,t) = I(x',y',t-\Delta t) = I(x-d_x, y-d_y, t-\Delta t), \tag{3.1}$$

$$I(\vec{r},t) = I(\vec{r} - D(\vec{r}), t-\Delta t), \tag{3.2}$$

Hypothesis: Intensity changes are only due to movement.

In the equations, X, Y, Z are the spatial coordinates of an object in a scene, x, y the spatial coordinates of the $2D$ representation in the image, and as normally t indicates time. I in this case is the image function, and $D(\vec{r})$ represents the estimated displacement vector. In fact $D(\vec{r}) = \vec{v}(\vec{r}) \cdot \Delta t$. Where $\vec{v}(\vec{r})$ is the

Optical Flow.

In a controlled environment, it is possible to take over the illumination, fixing it and setting a properly (ideally uniform) distribution. In this way, the change of illumination problem could be avoided in an application that concerns an automatic video-conferencing system in a suitable room. Once this is fixed, another problem arises from the fact of proposing an automatic system to track people in video-conferencing. Thinking of tracking people implies camera movement, and consequently the addition of such a variation on the Optical Flow estimation. The influence of this movement, external to the scene, will add a component of velocity to the whole image. This component will correspond to a model of movement or a parametrizable movement in the image, since it is due to a known displacement of the camera focus. The basic effects that can be modeled, will be the possible movements of the camera:

- Addition on the optical flow of the effect of a Zoom (In/Out).
- Addition on the optical flow of the effect of vertical axis rotation.
- Addition on the optical flow of the effect of horizontal axis rotation.
- Addition of an effects composition of the previous ones.

From the idea of the possibility of easy modeling how the camera movement affects the calculus of Optical Flow in a sequence, it follows that it a compensation of its influence should be realizable. In fact, if (i. e.) two frames are used to compute the optical flow, and in that precise instant the camera is turning, the known motion of the camera could be associated to a model and the second image only be predicted. Comparing both images (the expected one, without any other motion in it but the camera, and the real recorded one in second place) the motion of the camera would be compensated in the second one. In this way, what would remain should mostly be the motion corresponding to the objects in the scene.

In motion estimation, other problems appear. It is an intrinsic problem of the Optical Flow calculation when pixels disappear of the image or appear in it. The reason of such an effect, is due to the appearance or disappearance of objects in the scene, this is known as the Occlusion problem. In these cases, no reliable information about direction or velocity will be available.

Optical Flow computation techniques:

Despite its simplicity, Eq. (3.2) is a non-linear equation, and it is not evident how it must be solved. There are many linear approaches and several techniques

that give an estimation of the optical flow. The classification is not unique, and many reviews and surveys are written [2, 3, 28]. A synthesis of motion estimation can be found in [8, 23] as well.

Among the diversity of classifications, a general differentiation can be done. We can classify methods in:

- **Parametric Methods:** In this, Optical Flow is parametrized over an homogeneous movement area. It means that Optical Flow is approximated by a parametric model, and consequently, the possible movements or Optical Flow distributions are constrained to the ones that can be represented by the parametric function.

- **Non-Parametric Methods:** In this, Optical Flow is not constrained by any parametric function over a surface. An independent displacement vector on every pixel is thus obtained . Such techniques provide of a dense motion field.

The choice between both depends basically on the necessities in each case. In some cases, the initial constraint of a parametrization over a determinate area is interesting (mostly, because the image has been previously divided in regions, and the interest is about the general behavior of the whole region, not the pixel) [23]. In some other cases, what is interesting, is not having any kind of constraint, because there is no previous region or because what is needed is the independent approximation of the Optical Flow in every pixel [8].

In the text above, a differentiation in terms of utilization of parameters is done. Any way, it is not sufficient in itself since it tells nothing about the processes used to calculate the Optical Flow. From the scope of the used technique *Barron et al.* did the following classification:

- **Differential techniques** are also referred to as **gradient-based** techniques and are based on the assumption of *conservation of image intensity*.

- **Correlation (or Region)-based techniques** usually work on two successive images of the sequence: for each pixel in the second image a matching pixel in the first frame is searched, typically by analyzing the correlation of their neighborhoods.

- **Energy-based techniques** are based on the Spatio-temporal frequency characteristics of a moving visual stimulus.

- **Phase-based methods** define velocity in terms of the phase behavior of band-pass filter outputs.

3.2.2 Motion Estimation Application (*Lucas & Kanade* Algorithm)

Application of motion information, to Focus of Attention detection, needs a source of motion data which relies on no other previous information but the sequence of images. The desired effect would be the simulation of visual stimuli in front of sporadic movement, that means that this stimuli will happen before any other kind of image interpretation (like segmentation). A method independent of pre-set areas and capable to give a dense Optical Flow field is then required. This method must be unconstrained to any region, being able to give any evidence of displacement of any kind of area in the visual field. It follows that a non-parametric algorithm must be used.

The most used approaches to the Optical Flow calculation, are the Gradient-Based and the Region-Based methods. Both methods are widely used, but mostly on different scopes. The use of Gradient-Based methods allows to obtain the desired dense Optical Flow with pixel resolution, in contrast, Region-Based methods like *Block Matching* may give better precision in Optical Flow direction computation. Region-Based methods have the drawback of having less spatial resolution, and they also tend to be computationally complex (any way, to reduce it, fast suboptimal non-exhaustive search can be carried out).

In our application, the precision in the computation of the speed vector is not so important. We will be interested in the existence or non existence (with a necessary minimum coherence on the direction over a neighborhood) of movement. Taking into account all the previous assumptions, it seems reasonable to think about using a Gradient-Based technique.

The algorithm proposed by *Lucas & Kanade* [22] is a Gradient-Based algorithm. We choose to work on a variation proposed by *Simoncelli et al.* [29]. That algorithm introduces probabilistic extensions of gradient techniques which compute two-dimensional optical flow distributions. It includes an automatic gain control mechanism and provides (two-dimensional) flow vector confidence information, allowing later stages of processing to weight their use of the vectors accordingly.

Lucas & Kanade gradient algorithm

Eq. (3.2) being non-linear, a linear approximation would be useful for obtaining an analytical solution of the motion estimation problem. To that end, a first order-Taylor series expansion of the right-hand term in (3.2) can be derived. This is known as *"Optical Flow constraint equation"*:

$$\vec{v}(\vec{r}) \cdot \vec{\nabla} I(\vec{r}, t) + \frac{\partial I(\vec{r}, t)}{\partial t} = 0, \tag{3.3}$$

where $\vec{\nabla}$ is the spatial gradient operator, and $\vec{v}$ is the Optical Flow. This approximation is not valid for high velocities.

In this formula, it is assumed that changes in the image intensity (*hypothesis of intensity conservation*) are due only to translation of the local image intensity (motion is uniform and translational in a small region) and not to changes in lighting, reflectance, etc. Furthermore, by formulating the constraint only in terms of first derivative, it is implicitly approximating the image intensity as a planar function (it assumes the *hypothesis of motion spatial continuity*).

One could write a squared error function as follows:

$$E(\vec{v}) = \left(\vec{v}(\vec{r}) \cdot \vec{\nabla} I(\vec{r}, t) + \frac{\partial I(\vec{r}, t)}{\partial t} \right)^2. \tag{3.4}$$

But since it has to be minimized, the gradient (with respect to $\vec{v}$) equals zero:

$$\nabla_v E(\vec{v}) = \vec{0} \qquad \longrightarrow \qquad \nabla E(\vec{v}) = \mathbf{M} \cdot \vec{v} + \vec{b} = \vec{0}, \tag{3.5}$$

where

$$\mathbf{M} = \vec{\nabla} I(\vec{r}, t) \cdot \vec{\nabla} I(\vec{r}, t)^T = \begin{pmatrix} \frac{\partial I}{\partial x}^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \\ \frac{\partial I}{\partial y}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}^2 \end{pmatrix}, \qquad \vec{b} = \begin{pmatrix} \frac{\partial I}{\partial x}\frac{\partial I}{\partial t} \\ \\ \frac{\partial I}{\partial y}\frac{\partial I}{\partial t} \end{pmatrix}. \tag{3.6}$$

Matrix $\mathbf{M}$ in Eq. (3.6) is singular, due to the fact that the solution is based on a planar approximation to the image surface at only one point (the region of analysis for every computation is one pixel), and suffers from *the aperture problem* (it is only possible to identify the motion component parallel to $\vec{\nabla} I$ in the region of analysis). To avoid this problem an "intersection of constraints" rule is used. Instead of relying on just the derivatives in one point, the most consistent velocity in some small region will be computed. *The aperture problem*, is then in some way solved, since a bigger "window" is used to find the Optical Flow. In the other hand, since it is assumed that the velocity vector is constant in the region, some smoothness in the computation is introduced. This means some lose of spatial resolution, but the possibility to solve the Eq. (3.5).

It follows from the last paragraph and from (3.4) that

$$E(\vec{v}) = \sum_i \left(W_i \left(\vec{v} \cdot \vec{\nabla} I(\vec{r}_i, t) + \frac{\partial I(\vec{r}_i, t)}{\partial t} \right) \right)^2, \tag{3.7}$$

where $\mathbf{W}$ is the weighting window, and $i \in \{1, 2, \ldots n\}$ indexes the n elements of the region patch.

Joining (3.5) and (3.7), $\vec{v}$ can be solved:

$$\vec{v} = -\left(\sum_i W_i \cdot \mathbf{M_i} \right)^{-1} \cdot \left(\sum_i W_i \cdot \vec{b}_i \right). \tag{3.8}$$

In Eq. (3.8) although it is much more likely to be solved, $\mathbf{M}$ may still be singular (despite the blurring).

The Bayesian perspective of *Simoncelli et al.*

On the basis of the work of *Lucas & Kanade*, a probabilistic scope of the Optical Flow computation problem was taken. This point of view was qualified of being advantageous because it would take into account the inherent uncertainty due to image noise, lighting changes, low contrast regions, the aperture problem, and multiple motions in a single localized region.

Considering the total derivative constraint in equation (3.3), in practice, there will be errors in the derivative computations due to:

- Noise.
- Aliasing.
- Imprecision in the derivative filters.

all them due to the camera and the quantization process. Such errors, can be considered as a noise affecting the Optical Flow vector, and in the following they will be described by a Gaussian noise term $\vec{n}_2$.

In the case where those measurements are error-free, the constraint in (3.3) may fail to be satisfied because of:

- Changes in lighting.
- Reflectance.
- Presence of multiple motions.

and in consequence the constraint of the equality to $\vec{0}$ is broken. As it was modeled before, these errors will be considered again as a Gaussian noise, described by the term $\vec{n}_1$.

We thus obtain a new version of (3.3):

$$(\vec{v}(\vec{r}) - \vec{n}_1) \cdot \vec{\nabla} I(\vec{r}, t) + \frac{\partial I(\vec{r}, t)}{\partial t} = n_2, \qquad n_i \sim N(0, \Lambda_i). \tag{3.9}$$

In the beginning of this section we pointed out that a probabilistic scope is taken. The idea is to look for the probability function of $\vec{v}$, and since it is a linear function of independent additive Gaussian noise terms, its probability distribution will be Gaussian as well. The Maximum Likelihood Estimate (MLE) is simply the mean of the Gaussian, and consequently the probability distribution mean will be the most reliable velocity value.

The probability distribution is a conditional probability based on the image gradient, $\vec{\nabla} I$:

$$\mathbf{P}\left(\vec{v} \mid \vec{\nabla} I\right). \tag{3.10}$$

Decomposing the last expression using Bayes' rule, and following a development similar to that of (3.8) (see in Appendix C) we obtain:

$$\begin{aligned} \Lambda_v &= \left[\sum_i \frac{W_i \cdot \mathbf{M_i}}{\left(\sigma_1 \|\vec{\nabla} I(\vec{r}_i, t)\|^2 + \sigma_2\right)} + {\Lambda_p}^{-1} \right]^{-1}, \\ \mu_v &= -\Lambda_v \cdot \sum_i \frac{W_i \cdot \vec{b}_i}{\left(\sigma_1 \|\vec{\nabla} I(\vec{r}_i, t)\|^2 + \sigma_2\right)}, \end{aligned} \tag{3.11}$$

where σ_1 and σ_2 are, respectively, the standard deviation of $\vec{n}_1$ and $\vec{n}_2$, and Λ_p is the covariance for the Gaussian distribution $\mathbf{P}(\vec{v})$.

In Eq. (3.11) inversion is not anymore a problem, since Λ_p ensures the invertibility of the matrix. In addition, the $\mathbf{M}$ and $\vec{b}$ matrices are scaled by a compressive non-linearity, which acts as an adaptative gain control mechanism. It will tend to normalize the magnitude of the quadratic terms. From these expressions, we see that, for low contrast regions, the main source of errors will be noise of derivative measurements ($\vec{n}_2$). In contrast, failures ($\vec{n}_1$) of the constraint equation (Eq. 3.3) will be the dominant source of errors in high contrast regions.

Implementation Details

Eq. (3.11) is the one selected to generate the dense Optical Flow field for our application. The value of μ_v is the one that will determine the most suitable velocity vector over the neighborhood. The trial has been implemented in MATLAB® (see in appendix D). The commented source code is found in the appendix as well as the explanation of the approximation used for the spatial and temporal differentiation.

In the implementation, the classical scheme for Optical Flow computation has been followed. It is typically composed by:

1. Previous smoothing to increase the signal to noise ratio.
2. Computation of the image measurements. In this case, computation of the image derivatives, basis of the selected algorithm.
3. Integration of the image measurements to compute the Optical Flow using the algorithm.

A neighborhood of 5x5 pixels has been used to impose the "intersection of constraints" of the *Lucas & Kanade* algorithm. In order to make emphasize the pixel over which the velocity vector is computed, a Gaussian weight distribution is used to perform the summation in Eq. (3.11).

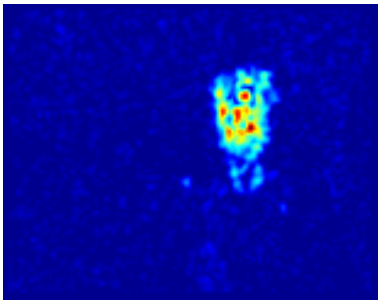
In the following, some results of the Optical Flow computation can be seen.

3.2.3 Statistical Motion Detection

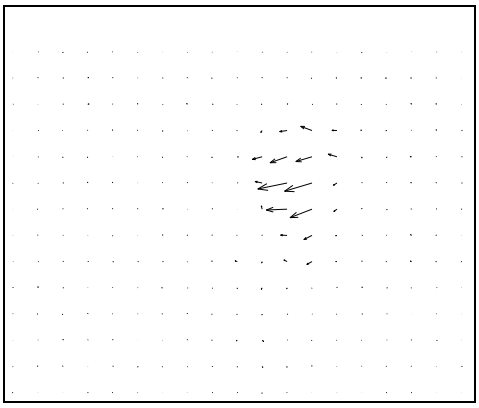
Once we have the estimated motion flow, we are interested in finding the moving regions in the image. A sequence of motion estimation frames is a sequence of matrices where each element in the matrix is a $2D$ element. The elements are vectors indicating direction and magnitude. Since we are only interested in the existence or not of motion, the relative motion magnitude is important.

Examples of motion detection and the basis of the present work can be found in [1, 35], where a comparison of each frame with a reference is worked out. Here, we will apply the statistical test used there but this time on estimated Optical Flow. This is done in order to be independent of a reference frame, since a moving camera will be used and we will not work on a static plane (contrary to classic surveillance applications).

(a) Subject moving the head.



(b) Optical Flow module.



(c) Downsampled Optical Flow

Figure 3.2: Motion Estimation

Motion Data Thresholding

One idea could be to compute the motion module all over the image, and consider as moving region all the pixels whose motion vector is larger than a given threshold. It is a very simple way to discriminate between moving and static areas, but it has some drawbacks:

- The threshold has to be set empirically at a certain value, which implies a certain lose of automation and in some way an increment of the supervision necessity.
- Weakness with respect to noise, since it is not taken into account its distribution.
- Low magnitude motion, but uniform and well defined could be missed due

to a high threshold.

In a well controlled environment, it is a cheap and easy technique of extracting motion areas. This approach performs sufficiently well if we analyze sequences where moving objects have a well contrasted motion vector in relation to noise. In addition, filtering can be used to clean out noise and allow a better region discrimination. Median filtering is well known to clean out impulsional noise, furthermore it does not provoke blurring on the data and consequently area limits are not smoothed. Any way, a determined pixel can propagate its value on the neighborhood, which is an area as big as the filtering kernel is. Examples of the use of median filtering in motion estimation and in other image features can be found in [8].

Fig. 3.3 shows an example of motion module data thresholding. Threshold selection is arbitrary and if it is not correctly adjusted for the sequence, a little difference from the optimal can lead to a biased detection. In addition, if the selected threshold is too high, holes in the regions and non-detections will appear.

Motion Data Statistical Detection

In order to perform a much more flexible detection, a kind of adaptative threshold according to the camera noise would be quite useful. The present procedure based on the *Aach* idea for change detection [1], and used in [35], models the noise and determine a threshold for each pixel of the image. This is done by taking into account the statistical behavior of each pixel's neighborhood.

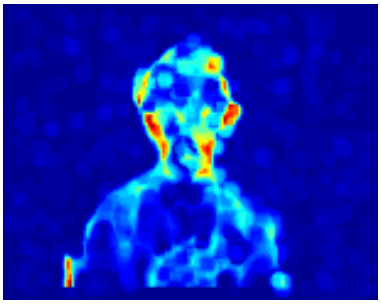
In the following, the statistical test will not be applied anymore on the difference of two images as *Aach* did. Since a more flexible tool is desired, we will apply it to estimated Optical Flow.

In order to discriminate movement magnitudes that are due to noise from those that are caused by a moving object it is necessary to define two hypothesis:

- H_0: there are no moving objects at image position $[i, j]$.
- H_1: complementary of H_0.

The indexes $[i, j]$ represent the pixel coordinates in an image. We take $\vec{v}$ the Optical Flow vector defined in sec. 3.2.2, and we define $v_{x[i,j]}$ and $v_{y[i,j]}$ as the horizontal and vertical components of it at the coordinates $[i, j]$. v_x and v_y will be two images. Each one containing the sign and magnitude of the velocity in the corresponding direction. In the application, we will work separately in the v_x and in v_y components. The resulting areas detected will be the union of the detected in

(a) Subject talking and moving.



(b) Median Filtered (5x5) Optical Flow Module.

(c) Moving area mask performed over a median filtered (5x5) version of the module(thresholded at 0.04).

(d) Moving area mask performed over a median filtered (5x5) version of the module(thresholded at 0.02).

Figure 3.3: Motion Region detection using thresholding.

each case. It could be also possible to work directly with the speed vector, using the two components joint distribution to model noise. Like this, direction could be also taken into account to discrimine noise. Since the literature, used as a basis, takes into account only one variable in the distribution, for the first trial we have worked on each component only with a one dimension distribution.

In the following, the explanation will be only performed with notation v_x, since both components have the same behavior.

$P(v_{x[i,j]} \mid H_0)$ is defined as the probability that $v_{x[i,j]}$ is different from zero, given the hypothesis H_0. In sec. 3.2.2 or [29], error distribution in motion estimation is assumed to be Gaussian. This is true if the input image sequences are affected by Gaussian noise, since it is a linear approximation of the Optical Flow,

and linear operations on a Gaussian give a Gaussian. If we suppose this noise to be uncorrelated, then $P(v_{x[i,j]} \mid H_0)$ can be modeled as:

$$P(v_{x[i,j]} \mid H_0) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{1}{2}\left(\frac{v_{x[i,j]}}{\sigma}\right)^2\right)}, \tag{3.12}$$

which is the Gaussian distribution with 0 mean and σ is the noise std. affecting motion information.

The objective is to find regions, and not only pixels. This is why the statistic is performed over a neighborhood and not on a single pixel. The test should detect those areas with a more or less uniform motion contribution over the area. This decision increases the reliability of the statistical description.

$P(v_{x[i,j]} \mid H_0)$ is a function of $\left(\frac{v_{x[i,j]}}{\sigma}\right)^2$. If we normalize v_x by σ, Eq. (3.12) turns into a normal ($N(0,1)$) distribution. Since we are interested in computing the statistical test over a neighborhood, for a Gaussian noise distribution on the velocity vector we obtain that from:

$$\bar{\Delta}^2_{[i,j]} = \sum_{w^n{}_{[i,j]}} \left(\frac{v_{x[i,j]}}{\sigma}\right)^2, \tag{3.13}$$

where $w^n{}_{[i,j]}$ is a window of width n centered at pixel coordinates $[i,j]$. $P(\bar{\Delta}^2_{[i,j]} \mid H_0)$ obeys a $\chi^2 - distribution$ (see app. E) with as many degrees of freedom as pixels inside the window. Noticed that this evaluation over a neighborhood is equivalent to the application of a low-pass filter to the squared difference image. On one hand this choice reduces the noise effect, on the other hand it causes a blurring effect or binary dilation on the resulted mask.

With the known distribution of Eq. 3.13, the decision between 'moving' and 'still' can be achieved by a significance test. For this purpose, we specify a significance level α and compute a corresponding threshold t_α according to

$$\alpha = P(\bar{\Delta}^2_{[i,j]} > t_\alpha \mid H_0). \tag{3.14}$$

For each pixel position $[i,j]$ in the difference image, we compute $\bar{\Delta}^2_{[i,j]}$, and whenever it is over the threshold t_α, we mark the pixel at position $[i,j]$ as moving. The significance level α is also an important parameter. It represents the probability of rejecting H_0, although it is true. In the present application we will set the parameter α to set the probability of rejection.

Experience has shown that Gaussian distribution is not a good model for camera noise. In sec. 3.2.2 (see also [29]) Gaussian noise distribution is assumed, but

experimental results have shown that in fact it is closer to a Laplace distribution. We can show this behavior with our measurements. In the following figure, experimental results are shown from a sequence of motion estimation of a static scene.

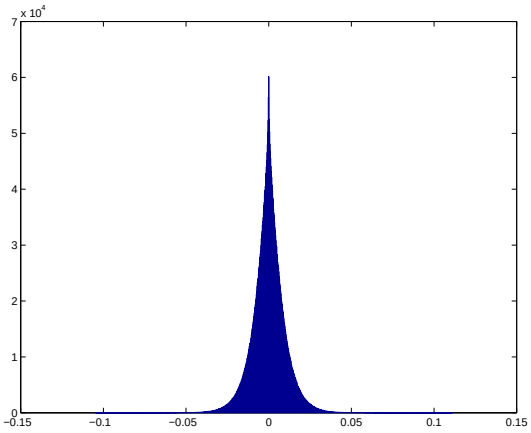


Figure 3.4: Noise histogram obtained empirically from 211 motion frames on the v_x component.

In order to use the same test, another evaluation must be used over the neighborhood to keep the $\chi^2 - distribution$. This is:

$$\bar{\Delta}_{[i,j]} = \sum_{w^n_{[i,j]}} \frac{\sqrt{2}}{\sigma} \left| v_{x[i,j]} \right| , \tag{3.15}$$

where in this, if $v_{x[i,j]}$ is a Laplace distribution, $\bar{\Delta}_{[i,j]}$ is a $\chi^2 - distribution$ with twice as many degrees of freedom as there are samples inside the local window w.

From Eq. 3.15 it turns out that now the evaluation is:

$$\alpha = P(\bar{\Delta}_{[i,j]} > t_\alpha \mid H_0). \tag{3.16}$$

To perform the test, the parameter α is set in order to define the probability that the pixel in $[i, j]$ has changed given the hypothesis H_0. In this case, the evaluation will be performed over the result of $\Gamma_q(a, z)$ where $\Gamma_q(a, z) = 1 - \Gamma_p(a, z)$.

So, every value of $\bar{\Delta}_{[i,j]}$ is evaluated in the following way:

$$sig\_level = 1 - \Gamma_p \left(\frac{r}{2}, \frac{x}{2} \right) , \tag{3.17}$$

where r is the number of degrees of freedom, and to allow the evaluation, a factor 2 must be added to $\bar{\Delta}_{[i,j]}$ in the Laplacian case, $x = 2 \cdot \bar{\Delta}_{[i,j]}$, in order to be like χ^2 function (in App. E).

The sig_level value is thresholded, and all pixels below the threshold are considered as moving pixels. The sig_level value determines the probability of that pixel to be noise given the hypothesis H_0. It follows that for low values of the threshold, only those pixels that are very probable to be moving will be marked.

3.2.4 Implementation

General Procedure

The practical implementation (see Fig. 3.5), follows a classical scheme of a first off-line parameter calibration phase and a second phase where the detection of moving areas is performed.

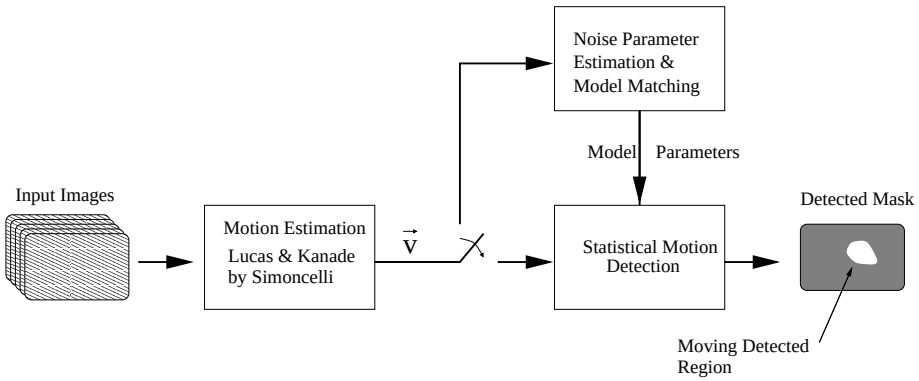


Figure 3.5: Moving Region Detection Algorithm Scheme.

From Fig. 3.5 is described the procedure:

1. The *Simoncelli's* version of the *Lucas & Kanade* algorithm is performed. As explained in sec. 3.2.2 and 3.2.2 (see code in App. D), we obtain from this step a dense, regularised and normalized optical flow.

2. When the system is started, the output sequence of the motion estimation of a static scene is supposed to be noise, since in reality it should be zero. From this sequence, the noise distribution is estimated. From the realized tests (see sec. 3.2.5), it has been found to be nearly a Laplacian distribution. Consequently, the σ and mean for the best matching Laplacian distribution are computed.

3. Once the noise distribution of the motion has been estimated off-line, the statistical test is applied to the sequence of Optical Flow Estimation using the computed distribution parameters in the initialization of the system.

Distribution model Estimation

To correctly perform the posterior test, we must find the best model for the empirical distribution.

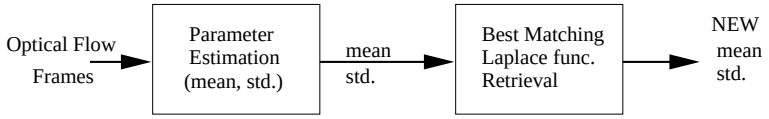


Figure 3.6: Statistical Noise Parameters Retrieval.

Fig. 3.2.4 shows the procedure steps. To estimate the best model, the σ and the mean from the incoming sequence are estimated. The computed parameters are used to initialize a Laplacian distribution model, this parameters are variated in order to find the best match distribution according to a Minimum Square Error (MSE) or a Minimum Absolute Error (MAE).

$$SE = \sum_i \left(H(i) - f_{model}(i)\right)^2 \qquad \text{or} \qquad AE = \sum_i |H(i) - f_{model}(i)|. \quad (3.18)$$

In the present work, any implementation of a smart convergent technique has been done to find the distribution model, since it was not the objective. For the test, an exhaustive search algorithm was used, and the search was performed on a determinate domain around the estimated σ from the data. See App. F for details.

Statistical Detection

In the statistical Detection step, mainly is being carried out the statistical test exposed in sec. 3.2.3. Any away, in addition some other features have been added, in order to ensure the robustness. Some post-processing features have been included to clean out any non-relevant detected point, and also it has been included the part to find the isolated detected areas, and compute its centroid.

In Fig. 3.7, we see the followed algorithm. It is composed by several steps, which can be described as:

1. Statistical test on the horizontal an vertical component of the Optical Flow separately. That gives as output two masks of detected movement.
2. Logical OR of the previous results.
3. Morphological improvement cleaning out possible Holes in regions, and eliminating non-relevant small spots on the mask. The used techniques are **Hole Filling** and **Erosion-Reconstruction Filtering**. See App. F and App. G.

4. Retrieval of all the 8 connected regions.
5. Computation of the centroid of the regions.

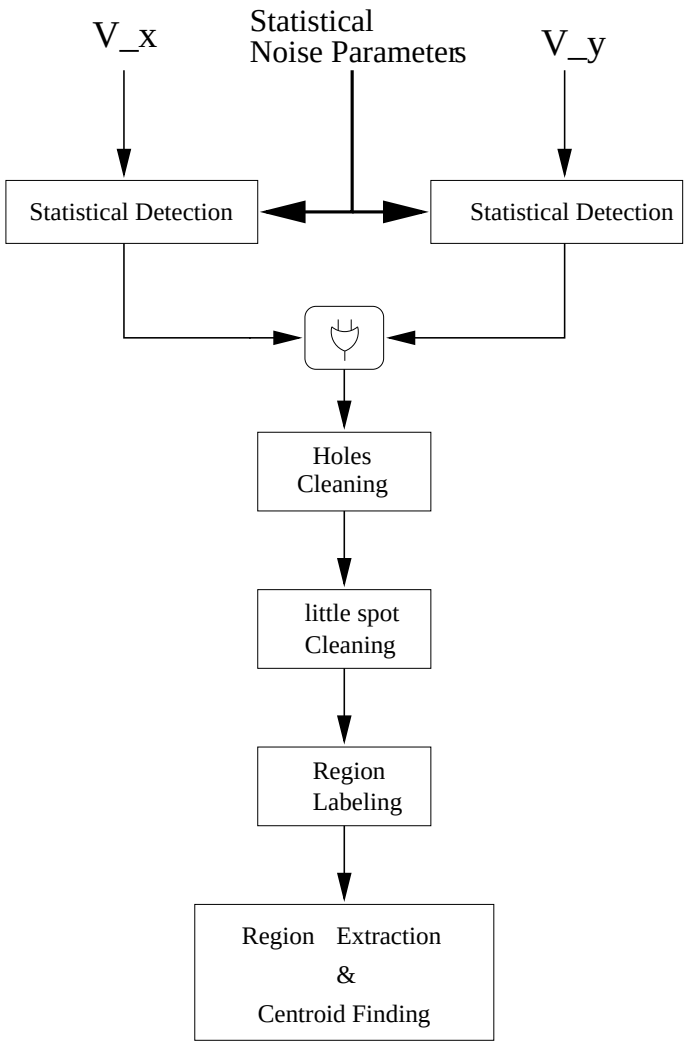


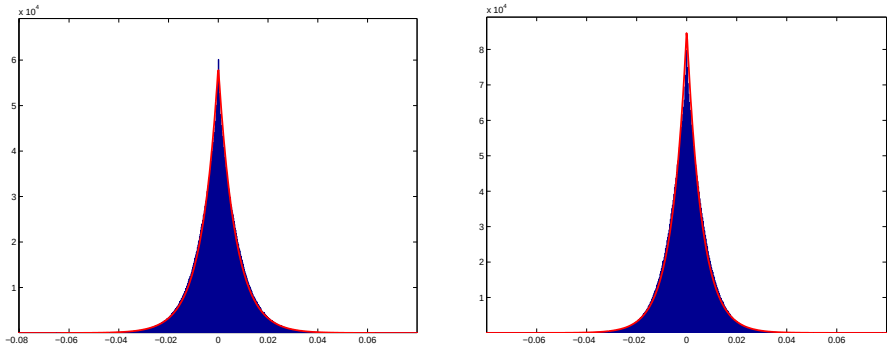
Figure 3.7: Statistical Motion Detection Scheme.

3.2.5 Results

The following results have been obtained from the MATLAB® code in App. F.

Noise estimation and modeling.

In Fig. 3.8 shows the match between the noise histogram and the Laplace distribution model. In this, 0 mean is assumed and the noise computed σ is used for each velocity direction. Notice that may not necessary be equal in both directions. This can be due to different mechanical stability of the camera in both directions, or due to some internal feature in the electronic circuitry.



(a) Noise Histogram for v_x $\sigma = 0.0098$ $\mu = 1.107e-5$ in blue. Laplace model with $\sigma = 0.0098$ $\mu = 0$ in red.

(b) Noise Histogram for v_y $\sigma = 0.0086$ $\mu = -1.5016e-5$ in blue. Laplace model with $\sigma = 0.0086$ $\mu = 0$ in red.

Figure 3.8: Experimental results on v_x and v_y noise distribution.

Despite the similarity between empirical observation and Laplace model, a more general model has been checked in order to see the most likely statistical model that matches. Since it is very similar to the Laplacian distribution, we have checked the *General Gaussian Distribution* (GGD) [31], which comprises the Laplace Distribution (parameter p=1) and the Gaussian Distribution (parameter p=2).

$$GGD(x,p,\sigma,\mu) = \frac{1}{2\Gamma\left(1+\frac{1}{p}\right)\sqrt{\sigma^2\frac{\Gamma\left(\frac{1}{p}\right)}{\Gamma\left(\frac{3}{p}\right)}}} e^{\left(-\left(\frac{|x-\mu|}{\sqrt{\sigma^2\frac{\Gamma\left(\frac{1}{p}\right)}{\Gamma\left(\frac{3}{p}\right)}}}\right)^p\right)}, \tag{3.19}$$

where p is the GGD parameter, μ the mean and σ the standard deviation.

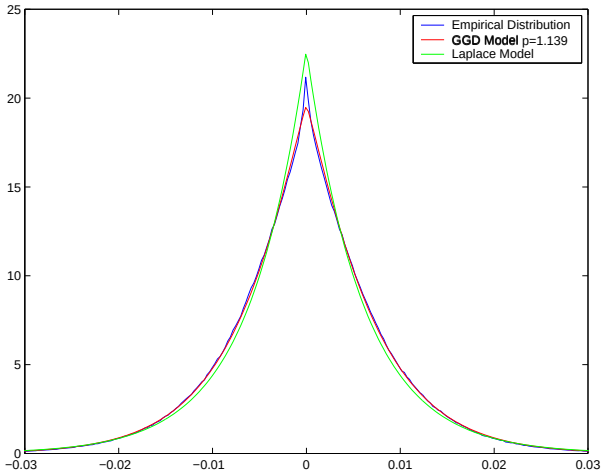


Figure 3.9: Comparison among Empirical, Laplace and GGD for v_y velocity component.

The result gives that the closest distribution is a GGD with parameter $p = 1.139$ for the vertical component, it is very close to Laplacian, but it is not a pure Laplacian. In the horizontal component v_x a closer to Laplacian parameter has been found, $p = 1.090$. Since it is not exactly a Laplacian, and we are approximating it by a Laplacian, we check which would be the variance that makes both distributions (the empirical and the model) best match according to MSE. In our test sequence we have obtained a slightly increase in the variance (not much since both are very similar).

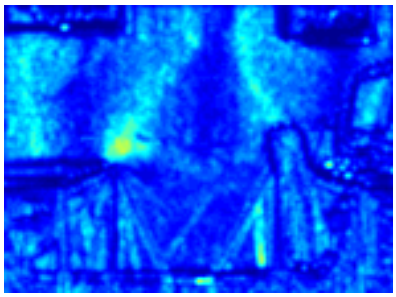
$$\sigma_{v_x} = 0.0105 \qquad (\text{before } 0.0098)$$
$$\sigma_{v_y} = 0.0095 \qquad (\text{before } 0.0086)$$

Detection Results

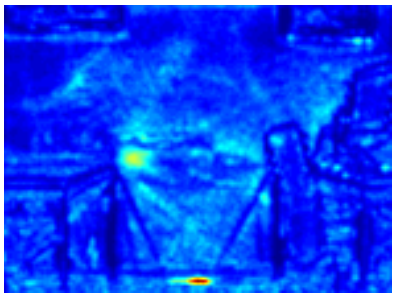
We have applied this test with the computed parameters to the sequence used to estimate the noise. The observed results has shown that although a general statistic on the sequence turns out to be a GGD (very close to a Laplacian), locally it can have a different behavior than the general. Since the environment is not ideal and illumination is not uniform.

In the figure above (b) (c) can be observed the non uniform distribution of σ for each pixel over the sequence. The most affected areas are the shadow areas and dark objects. This could be due to the performance of the camera, or due to illumination drifts. In fact, during the sequence, some changes suddenly appear reflected in the temporal derivative, when motion is computed. This is a proof of the existence of illumination drifts.

(a) Static Scene to measure noise.



(b) Noise σ for each pixel on the x component through time.



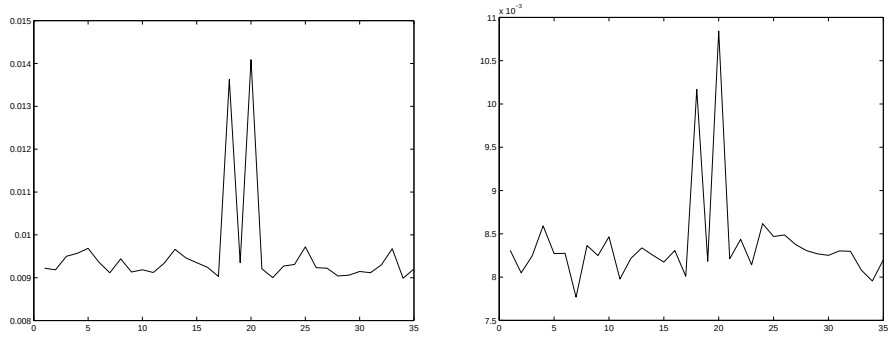
(c) Noise σ for each pixel on the y component through time.

Figure 3.10: Standard deviation of the pixels through time.

In Fig. 3.11, in (a) and (b) one can see the evolution through time of the variance of pixel motion into the frame. Both are graphics that represents σ against the frame number. Notice the high increment of the std. deviation in the 18th frame and see Fig. 3.12.
In this frame, something is changing.

In Fig. 3.12 we show the results of applying the test to the training sequence. We can see that it performs well since is correctly detected as non moving. Some little white spots can be seen due inhomogeneities of σ all over the frame. A special case is the 18th frame. As it has been seen in Fig. 3.11 in the there is a really high increment in the σ (something has a displacement) mean over the frame. It is clearly reflected in (l) from Fig.3.12.

Fig. 3.12 also shows that with the increment of the window size, it is more sensitive to very low motion but with uniform magnitude.



(a) Noise σ for each frame on the x component through time.

(b) Noise σ for each frame on the y component through time.

Figure 3.11: Standard deviation of the frames through time.

- In a **1x1 window** test, no detection is observed. It is the ideal situation, since there is no moving object in the scene, it does not detect if there is any other variation, but it will not be so sensitive to low motion areas.

- In a **3x3 window** test, the detection is almost the desired. A little few of spots can be seen through the sequence, but it can be easily cleaned with a morphological binary filter. The advantage, is that it will be more sensitive to low motion, but the drawback will be major sensitivity to changes in illumination like in the 18th frame of the sequence test (h) from Fig. 3.12.

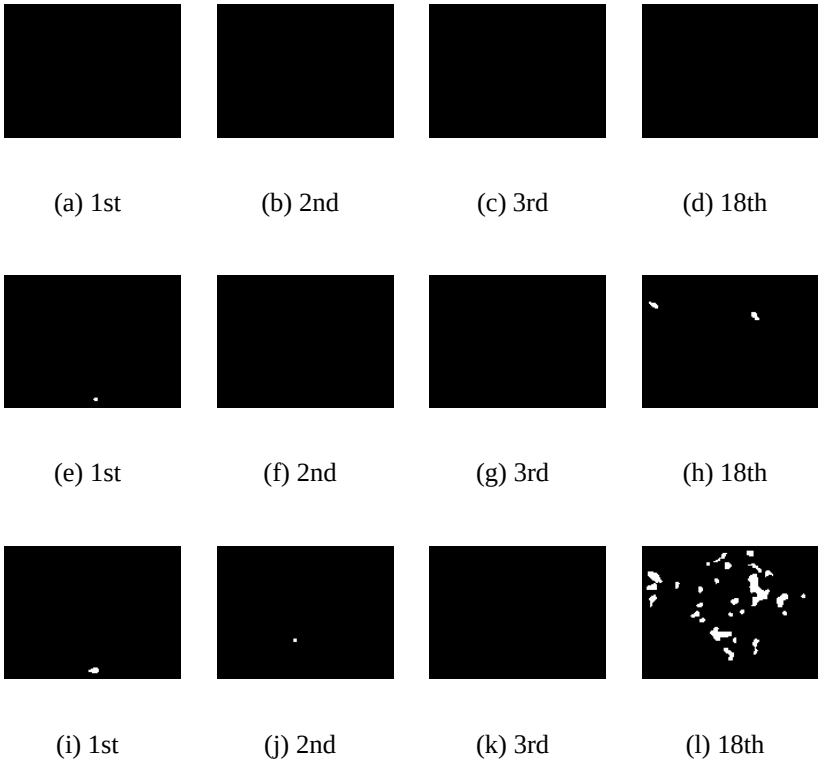
- In a **5x5 window** test, the sensitivity to slow uniform motion is increased at the expense of larger sensitivity to slight motion (see frame (l) in the following figure).

Another effect of increasing the window size will be a relative blurring of the area's contours. Anyway, since the objective is only to detect the origin of motion, a few pixels of blurring on the boundaries of motion areas are not important.

The results applying the algorithm test (see app. F) with different window test sizes can be seen in Fig. 3.13 and 3.14.

As it is seen on these figures, the algorithm performs well and detects the moving regions. In the first one (Sequence A in Fig. 3.13), only the head is moving. In the second (Sequence B in Fig. 3.14) the main motion is in the face (the subject is talking), but the body is also (though slower).

The effect of the window size is clear. The bigger it is the more sensitive to slow motion. If we increase the size, the blurring effect will increase and the sensitivity to illumination drifts and others will increase, correspondingly. The



(a) 1st (b) 2nd (c) 3rd (d) 18th

(e) 1st (f) 2nd (g) 3rd (h) 18th

(i) 1st (j) 2nd (k) 3rd (l) 18th

Figure 3.12: Results on the application of the test on the noise training sequence.(a) (b) (c) (d) are detected using a 1 element window. (e) (f) (g) (h) are detected using a 3x3 elements window. (i) (j) (k) (l) are detected using a 5x5 elements window.

selection of a 3x3 or 5x5 window is a good compromise.

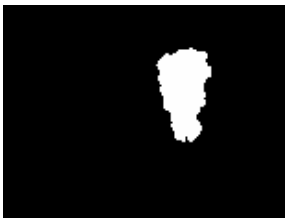
3.2.6 Conclusions

We have seen that from dense Optical Flow information, we can determine which are the moving areas in a sequence of images. Consequently, we can obtain an orientation of what can be interesting to look at in a scene.

The present application, performs a previous study of the noise distribution. Noise is modeled searching the most similar distribution. In our case we have proved that the best way to match the distribution is using the GGD model. Anyway, a statistical test based on the Laplacian is used because the GGD model found was really very close to the the Laplacian distribution, and the χ^2 test for a Laplacian was well known. We conclude that it is a good approxi-

(a) Sequence A

(b) 1x1 window



(c) 3x3 window

(d) 5x5 window

Figure 3.13: Results obtained on real sequences with different window sizes. All them are obtained using the estimated σ above, and with a fixed probability of H_0 rejection (although it is true) of 1e-12

mation, since the results obtained have shown to detect correctly the moving areas.

The present solution, compared with others that use the comparison between the instantaneous frame and a reference one, shows to be more adequate in the context of an Active Vision system. This means, in a system where the camera is not static, and where it acts as a dynamic element in the system, focusing, zooming, rotating... In a situation like this, there is not a fixed view, the environment changes dynamically. Other solutions like panoramic reference frames (*Sprites*) have the same problem, since they need also refresh of the reference frame, and they detect presence and not movement.

In movement detection application the contours may not coincide with the contours of the moving object, this is since it is evident that not the whole object may be moving (see Fig. 3.13 where only the head is moving and the body is not detected). But the objective is not to segment with this tool, this tool is mainly to guide where it must be segmented. In fact, is an advantage only detecting the moving regions, in this way, all the attention will be concentrated where the action takes place. An other advantage of the present, is the possibility to compensate the

(a) Sequence B

(b) 1x1 window

(c) 3x3 window

(d) 5x5 window

Figure 3.14: Results obtained on real sequences with different window sizes. All them are obtained using the estimated σ above, and with a fixed probability of H_0 rejection (although it is true) of 1e-12

motion induced in the Optical Flow estimation when the camera is moving. The computer that controls the camera, knows perfectly which is the movement of the camera all the time, since it is the computer who orders it to move, and which can be the motion induced on the estimation. With that, the motion induced by the camera should be compensated.

Chapter 4

Conclusions

4.1 Achievements

In this work we have studied two approaches for the *Smart Video-Conferencing System* part of Motion Detection and Segmentation. The application of both in a sequence of images can be seen in Fig. 4.1. In it, we look for the a focus of attention and in it we perform a segmentation in a certain area around the focus of attention point.

It is possible to implement a motion detection based on a Statistical Test of a dense Optical Flow. An uncommitted visual front-end can be as well implemented for a meaningful object retrieval. In a posterior step, each of the found regions can be analyzed in a higher level fashion. In sec. 2.5 and 3.2.6 con be found the conclusions in each part of study.

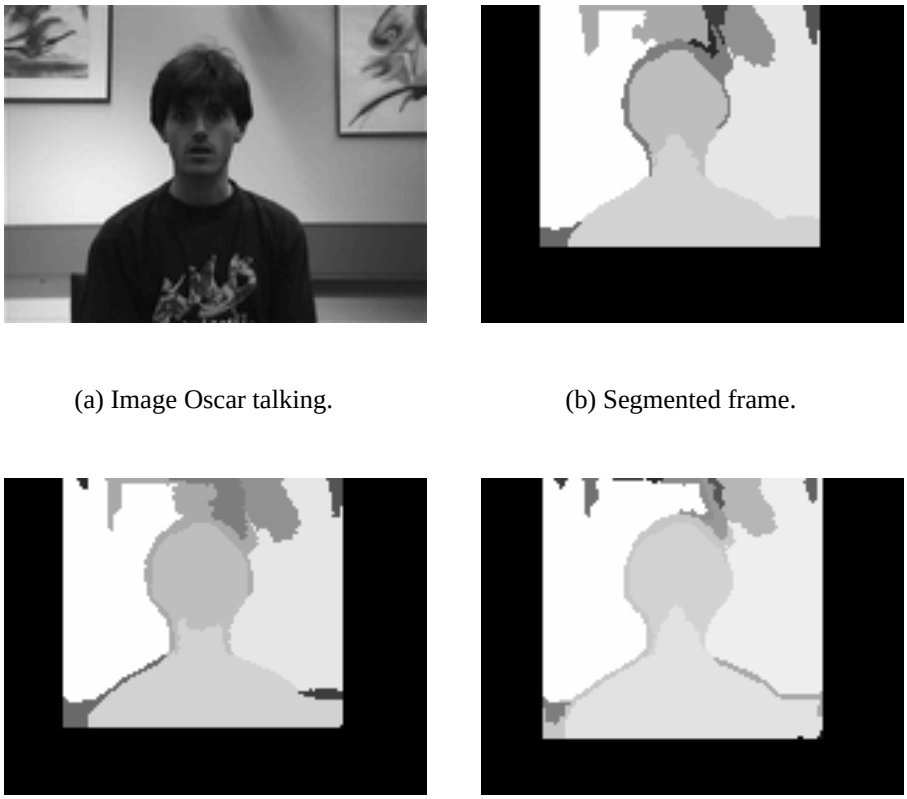
Both approaches have shown to be promising techniques, and to have many possibilities of application. Their use is not limited to a video-conferencing system, they have many possibilities in everything related with *Computer-vision*. In example, the use of them in surveillance, could be as useful to locate the place of the action, as well to perform the segmentation in the right place.

4.2 Possible Extensions and Future Work

From this work much further work can be done in the future.

Focus of attention

- The others features like sound and colors, can be implemented.



(a) Image Oscar talking. (b) Segmented frame.

(c) Segmented frame. (d) Segmented frame.

Figure 4.1: Segmentation on the sequence Oscar talking. The focus of attention is searched (the speaking subject) and a segmentation is performed in a squared area centered in it.

- Some technique can be used to combine the three or more features to find the focus of attention more reliably, like the use of a kalman filter.

Motion Finding

- The Optical Flow velocity due to de camera motion should be compensated, since the displacement of the camera is known by the system, and it could be modeled and substracted from the estimation.

- Dynamic noise estimation. Using as initialization the technique studied, an estimation of the noise evolution in real time could be implemented performing the noise estimation on the areas detected as non moving.

- Study of the application of the "χ^2 Test" on the General Gaussian Distribution (GGD), to use the best statistical model instead of the Laplacian approximation.

Scale-Space Segmentation

- Study of a possible extension of the scale-space in RGB. Now, only the luminance information is used. It is clear that colors bring much useful information to differentiate between two regions. So, the use of colors should be included.

- Study of more band-pass features. Studies on the HVS indicate that it uses Wavelet Gabor Function Analysis. It could be a point to start.

- Implementation of a post-processing technique relying on the gray level to clean out very small regions (1-2-3 pixel size), very common in the final segmentation after the tree reconstruction. (Now, a morphological technique is being used).

Appendix A

Greens Function

A Greens function is an integrating kernel which can be used to solve an inhomogeneous differential equation with boundary conditions. It serves roughly an an analogous role in partial differential equations as does Fourier analysis in the solution of ordinary differential equations.

Consider the $1D$ case.

Lets $\tilde{L}$ be a differential operator such that

$$\tilde{L} = \tilde{D}^n + a_{n-1}(t)\tilde{D}^n - 1 + ... + a_1(t)\tilde{D} + a_0(t), \tag{A.1}$$

with $a_i(t)$ continuous for $i = 0, 1, ...n - 1$ on the interval I, and assume we wish to find the solution $y(t)$ to the equation

$$\tilde{L}y(t) = h(t), \tag{A.2}$$

where $h(t)$ is a given continuous on I. To solve Eq. above, we look for a function $g : C^n(I) \mapsto C(I)$ such that $\tilde{L}\left(g(h)\right) = h$ where

$$y(t) = g\left(h(t)\right). \tag{A.3}$$

This is a convolution equation of the form

$$y = g * h, \tag{A.4}$$

so the solution is

$$y(t) = int_{t_0}^{t} g(t - x)h(x)\,\mathrm{d}x, \tag{A.5}$$

and the function $g(t)$ is called the **Greens function** for $\tilde{L}$ on I. Now, note that if we take $h(t) = \delta(t)$, then **Greens function** $g(t)$ can be defined by:

$$\tilde{L}g(t) = \delta(t). \tag{A.6}$$

For an arbitrary linear differential operator $\tilde{L}$ in $2D$, the **Greens function** $G(\vec{r}, \vec{r}')$ is defined by analogy with the $1D$ case by

$$\tilde{L}G(\vec{r}, \vec{r}') = \delta(\vec{r} - \vec{r}'). \tag{A.7}$$

Appendix B

Implementation Details in Scale-Space Segmentation

program.m

```
function [tot_masc,new_regions]=program(im,ini,cleaning,s_per_oct,slices)
%
% This program performs the segmentation of the im-
age im using scale-space.
%
% im: image
% ini: level from the computed stack from where to be-
gin the segmentation
% (the downward projection).
% cleaning: 0 no morphological cleaning / 1 yes.
% s_per_oct: number of samples per octave.
% slices: number of levels till the top of the stack.
%
% tot_masc: regions mask. The segments in gray level.
% new_regions: cell array as long as there are seg-
ments in the image. In
% each cell there is a matrix with the pixel positions.
%
%

disp('Building Tree...');
[family,dimim]=buildtree(im,s_per_oct,slices);

[x level]=size(family);
```

```
%figure(1);
%colormap(gray);
%imagesc(im);

leveln=level-ini+1;
%dimim=size(im);
disp('Segmenting...');
regions=segment(family,leveln,dimim);
disp('Ordering...');
[new_regions,tot_masc]=order_regions(regions,dimim);

clear regions;

%morphologic alternate sequential filter

if(cleaning)
disp('Cleaning...');
tot_masc=clean_small_reg(tot_masc,1);
itera=2;
  change=0;
  old_tot_masc=tot_masc;
  while(change==0)

    disp('Cleaning...');
    tot_masc=clean_small_reg(tot_masc,itera);
    if(old_tot_masc==tot_masc)
      change=1;
    else
      itera=itera+1;
      old_tot_masc=tot_masc;
    end
  end

%Reordering of the regions and regrouping.

    disp('Regrouping...');
    r=1;
    for l=1:length(new_regions)
      [a,b]=find(tot_masc==l);
      dima=size(a);
      if(dima(2)==1)
        a=a.';
        b=b.';
```

```
      end
      if(length(a)>0)
        [Labeled,NUM]=bwlabel(full(sparse(a.',b.',1,dimim(1),
dimim(2))),8);

        if(NUM>1)
          for ind=1:NUM
            [a,b]=find(Labeled==ind);
           dima=size(a);
            if(dima(2)==1)
              a=a.';
              b=b.';
            end
            regions{r}=[a; b];
            r=r+1;
          end
        else
          regions{r}=[a; b];
          r=r+1;
        end
      end
    end

    disp('Ordering...');
    [new_regions,tot_masc]=order_regions(regions,dimim);
    clear regions;

end

number_of_found_regions=length(new_regions)
%figure(f+1);
%colormap(gray);
%imagesc(totmasc);

function out=clean_small_reg(segmented_im,steps)

if(steps<=0)
  error('steps: bad steps number');
end

dilated=segmented_im;

for r=1:steps
```

```
dilated=dilate_gray4(dilated);
end


out=erec_gray4(segmented_im,dilated,1);
```

buildtree.m

```
function [family,dim]=buildtree(im,s_per_oct,slices)
%
%
% Function that performs the analysis of the im-
age structure
% im: images
% s_per_oct: number of samples per octave.
% slices: number of levels till the top of the scale-
space
%
% family: cell array, in each cell there is the rela-
tion of
% parent and children.
% dim: is the dimensions of scale.

disp('   Building Scale-Space');
[pre_scale,edges_scale]=scalspc_lh(im,slices,1,0.90,s_per_oct);

scale=pre_scale;
dim=size(scale)

noson=[];
volumes=[];
%exspc1=ones(dim(1),dim(2));
%exspc2=ones(dim(1),dim(2));
diff=double(max(max(scale(:,:,1))))-
double(min(min(scale(:,:,1))));

disp('   Linking Process');
for n=2:dim(3)
  disp('Remaining...');disp(dim(3)-n+1);
  [family_full,noson,volumes]=tree_level(ones(dim(1),dim(2)),ones(dim(1
noson,n-1,dim,double(scale(:,:,n-
```

```
1)),double(scale(:,:,n)),diff,volumes,
edges_scale(:,:,n),s_per_oct);
  disp('   fathers_number:');disp(length(family_full));

% compressing data to store it.
  family_half{1}=sparse(family_full(:,:,1));
  family_half{2}=sparse(family_full(:,:,2));
  clear family_full;

  family{n-1}=family_half;
  clear family_half;

end
clear noson;
```

scalspc_lh.m

```
function [bigout_l,bigout_h]=scalspc_lh(im,N,mode,sigma,s_per_oct)
%
%
% function that generates the scale-space of the orig-
inal image
% (the low-pass and the band pass.
%
% im: image
% N: number of levels till the top of the stack.
% mode: indicates if generating N stack lev-
els (mode=1) or
% generating till a certain width of the trans-
form (mode=0).
% sigma: width of the transform
% s_per_oct: number of levels per octave.
%
% bigout_l: low-pass stack. basis of the struc-
ture of the image.
% bigout_h: stack of the detected edges through scale.

   %figure(1);
   %imshow(im);
```

```
   dim=size(im);

%image padding.
   im2=ones(512,512).*mean2(im);
   im2(1:dim(1),1:dim(2))=im;
   bigout_l=im;

   bigout_h=im;

%generation of the levels
   tim=fft2(im2);
   %old_filtered=tim;
   old_filtered=tim.*fftshift(tgaussian(0.5,[512 512],s_per_oct));
   k=1;
   while ((k<=N & mode) | (k>1 & tgauss(1,2)>=sigma & ~mode) | k==1)
      tgauss=fftshift(tgaussian(k,[512 512],s_per_oct));
      filtered=tim.*tgauss;
      out1=abs(ifft2(filtered));
      %out2=real(ifft2(old_filtered-filtered));
      filtered2=tim.*fftshift(tgaussian(k+0.5,[512 512],s_per_oct));
      out2=real(ifft2(old_filtered-filtered2));
      %old_filtered=filtered;
      old_filtered=filtered2;

      k=k+1;
      %figure(k);
        %imshow(out,[min(min(out)) max(max(out))]);
      bigout_l(:,:,k)=out1(1:dim(1),1:dim(2));
      bigout_h(:,:,k)=find_zero_cros(out2(1:dim(1),1:dim(2)));
   end


function masc=find_zero_cros(im)
%
%this function returns a binary mask of the esti-
mated edges
%having used the Laplacian (DOG approx.).
%

   m=size(im,1);
   n=size(im,2);
```

```
   b=im;

   rr = 2:m-1; cc=2:n-1;

   thresh=0;

   masc=logical(zeros(size(im)));

   % Look for the zero crossings:  +-, -
+ and their transposes
   % We arbitrarily choose the edge to be the nega-
tive point
   [rx,cx] = find( b(rr,cc) < 0 & b(rr,cc+1) > 0 ...
      & abs( b(rr,cc)-b(rr,cc+1) ) > thresh );   % [-
+]
   masc((rx+1) + cx*m) = 1;
   [rx,cx] = find( b(rr,cc-1) > 0 & b(rr,cc) < 0 ...
      & abs( b(rr,cc-1)-b(rr,cc) ) > thresh );   % [+ -
]
   masc((rx+1) + cx*m) = 1;
   [rx,cx] = find( b(rr,cc) < 0 & b(rr+1,cc) > 0 ...
      & abs( b(rr,cc)-b(rr+1,cc) ) > thresh);   % [-
+]'
   masc((rx+1) + cx*m) = 1;
   [rx,cx] = find( b(rr-1,cc) > 0 & b(rr,cc) < 0 ...
      & abs( b(rr-1,cc)-b(rr,cc) ) > thresh);   % [+ -
]'
   masc((rx+1) + cx*m) = 1;

   % Most likely this covers all of the cases. Just check to see
   % if there are any points where the LoG was pre-
cisely zero:
   [rz,cz] = find( b(rr,cc)==0 );
   if ~isempty(rz)
      % Look for the zero crossings: +0-, -
0+ and their transposes
      % The edge lies on the Zero point
      zero = (rz+1) + cz*m;   % Linear in-
dex for zero points
      zz = find(b(zero-1) < 0 & b(zero+1) > 0 ...
         & abs( b(zero-1)-b(zero+1) ) > 2*thresh);     % [-
```

```
0 +]'
       masc(zero(zz)) = 1;
      zz = find(b(zero-1) > 0 & b(zero+1) < 0 ...
         & abs( b(zero-1)-b(zero+1) ) > 2*thresh);     % [+ 0 -
]'
       masc(zero(zz)) = 1;
      zz = find(b(zero-m) < 0 & b(zero+m) > 0 ...
         & abs( b(zero-m)-b(zero+m) ) > 2*thresh);     % [-
0 +]
       masc(zero(zz)) = 1;
      zz = find(b(zero-m) > 0 & b(zero+m) < 0 ...
         & abs( b(zero-m)-b(zero+m) ) > 2*thresh);     % [+ 0 -
]
      masc(zero(zz)) = 1;
   end
```

tree_level.cpp

```
////////////////////////////////////////////////////////////////
//
//
//
//   The following C/C++ Code performs the tasc corre-
sponding to the
//   linking. It builds the struc-
ture which will be used to perform
//   the segmentation after.
//
//   This code is embeded into a matlab pro-
gram. So, it uses the
//   C/C++ matlab interface. It has been done like that to allow
//   to continue working in matlab, the Mat-
lab code equivalent to
//   this function was extremelly memory consumming.
//
//
//
////////////////////////////////////////////////////////////////
#include <stdio.h>
#include <stdlib.h>
```

```
#include <math.h>
#include "mex.h"
#include "matrix.h"
#include "mat.h"
#include <ctype.h>
#include <vector>
#include "tree_level.h"

#define MAX_ITER 20 //Max number of itera-
tion  to check criteria
#define WG 0.4
#define WI 1.0      //Weight values of the criteria
#define WM 0.4
#define NO_EDGE_DETECT 1
        //1 do not use edge detec-
tion, 0 use edge detection.
//#define OCTAVE_DIV 4.0

void mexFunction(int nlhs, mxArray *plhs[], int nrhs,
 const mxArray *prhs[])
{
  double *o_family,*o_newnoson,*o_newvolumes,diff=0.0,OCTAVE_DIV=0;
  Image i_chl,i_fath,i_scale1,i_scale2,i_edge;
  Container i_noson,i_volumes;
  int level=0,dimens_fam[3],sum_fathers=0;
  double *dimens,width,heigth;
  vector<int> *family[2];
  vector<int> newnoson[2];
  vector<double> newvolumes[4];

  //MATFile *arxiu;

  if(nrhs!=11){
    mexErrMsgTxt("Bad number of input arguments");
  }
  if(nlhs!=3){
    mexErrMsgTxt("Bad number of output arguments");
  }

  //mexPrintf("Loading Data...\n");
```

```
  // Data reading


   i_chl.image=mxGetPr(prhs[0]);
   i_chl.width=mxGetN(prhs[0]);
   i_chl.heigth=mxGetM(prhs[0]);
   //mexPrintf("i_chl Heigth: %d, Width: %d\n",
i_chl.heigth,i_chl.width);

   i_fath.image=mxGetPr(prhs[1]);
   i_fath.width=mxGetN(prhs[1]);
   i_fath.heigth=mxGetM(prhs[1]);
   //mexPrintf("i_fath Heigth: %d, Width: %d\n",
i_fath.heigth,i_fath.width);

   i_noson.position=mxGetPr(prhs[2]);
   i_noson.length=mxGetN(prhs[2]);
   i_noson.heigth=mxGetM(prhs[2]);
   //mexPrintf("i_noson Heigth: %d, Width: %d\n",
i_noson.heigth,i_noson.length);

   if(i_noson.heigth!=0 && i_noson.heigth!=2){
     mexPrintf("Heigth: %d, Width: %d\n",i_noson.heigth,
i_noson.length);
     mexErrMsgTxt("Bad dimensions in noson");
   }

   level=(int)(*(mxGetPr(prhs[3])));
   //mexPrintf("Level: %d\n",level);

   dimens=mxGetPr(prhs[4]);

   i_scale1.image=mxGetPr(prhs[5]);
   i_scale1.width=mxGetN(prhs[5]);
   i_scale1.heigth=mxGetM(prhs[5]);
   //mexPrintf("i_scale1 Heigth: %d, Width: %d\n",
i_scale1.heigth,i_scale1.width);

   i_scale2.image=mxGetPr(prhs[6]);
   i_scale2.width=mxGetN(prhs[6]);
```

```
   i_scale2.heigth=mxGetM(prhs[6]);
   //mexPrintf("i_scale2 Heigth: %d, Width: %d\n",
i_scale2.heigth,i_scale2.width);
   diff=(*(mxGetPr(prhs[7])));

   i_volumes.position=mxGetPr(prhs[8]);
   i_volumes.length=mxGetN(prhs[8]);
   i_volumes.heigth=mxGetM(prhs[8]);

   i_edge.image=mxGetPr(prhs[9]);
   i_edge.width=mxGetN(prhs[9]);
   i_edge.heigth=mxGetM(prhs[9]);

   OCTAVE_DIV=(* (mxGetPr(prhs[10])));

   //mexPrintf("done\n");

   //mexPrintf("Computing Parent-
Children relationships...\n");

   //function*********************************
   sum_fathers=tree_level(&i_chl,&i_fath,&i_noson,&i_volumes,
level,newnoson,family,dimens_fam,&i_scale1,&i_scale2,diff,
newvolumes,&i_edge,OCTAVE_DIV);
   //function*********************************
   //mexPrintf("done\n");

   //mexPrintf("Creating matlab variables...\n");
   if(newvolumes[0].size()>0){
     plhs[2]=mxCreateDoubleMatrix(4,newvolumes[0].size(),mxREAL);
   }
   else{
     plhs[2]=mxCreateDoubleMatrix(0,0,mxREAL);
   }

   if(newnoson[0].size()>0){
     plhs[1]=mxCreateDoubleMatrix(2,newnoson[0].size(),mxREAL);
   }
   else{
     plhs[1]=mxCreateDoubleMatrix(0,0,mxREAL);
   }
```

```
   plhs[0]=mxCreateNumericArray(3,dimens_fam,mxDOUBLE_CLASS,mxREAL);

   o_newvolumes=mxGetPr(plhs[2]);
   o_newnoson=mxGetPr(plhs[1]);
   o_family=mxGetPr(plhs[0]);

   int k=0;
   for(int i=0;i<sum_fathers;i++){

     if(family[0][i][0]!=-1 && family[0][i][0]!=0){

       for(int j=0;j<dimens_fam[0];j++){
         o_family[j+k*dimens_fam[0]]=(double)(family[0][i][j]);
         o_family[k*dimens_fam[0]+j+dimens_fam[0]*dimens_fam[1]]=
(double)(family[1][i][j]);
       }
       k++;

     }
   }

//    if((k*dimens_fam[0])!=(dimens_fam[0]*dimens_fam[1])){
//      mexPrintf("he fet: %d, hau-
ria d'haver fet: %d\n",
k*dimens_fam[0],dimens_fam[0]*dimens_fam[1]);
//      mexErrMsgTxt("error ja saps on");
//    }

   // arxiu=matOpen("dades.mat","w");
//    mxSetName(plhs[0],"fam");
//    matPutArray(arxiu,plhs[0]);
//    matClose(arxiu);

   if(newnoson[0].size()>0){
     for(int i=0;i<newnoson[0].size();i++){
       o_newnoson[2*i]=(double)newnoson[0][i];
       o_newnoson[2*i+1]=(double)newnoson[1][i];
     }
   }
```

```
   if(newvolumes[0].size()>0){
     for(int i=0;i<newvolumes[0].size();i++){
       o_newvolumes[4*i]=newvolumes[0][i];
       o_newvolumes[4*i+1]=newvolumes[1][i];
       o_newvolumes[4*i+2]=newvolumes[2][i];
       o_newvolumes[4*i+3]=newvolumes[3][i];
     }

   }

   //mexPrintf("done\n");


   delete[] family[0];
   delete[] family[1];

}

int tree_level(Image *chl,Image *fath,Container *no-
son,Container
*i_volumes,int level,vector<int> newno-
son[],vector<int> *family[],
int dimens_fam[],Image *scale1,Image *scale2,double diff,
vector<double> newvolumes[],Image *edge,double OCTAVE_DIV){


  int sum_fathers=0,k=0,father=0,max_son=0,fathers_wo_son=0;
  int *fathers[2], *number_son;
  double limwindow=0,norma=0,color=0,dif_meassure=0,old_max_dif=0,
oldsim=0,distance=0,d=0,sim=0,sigmap=0,sigmac=0,searchrad=0,
volume_max=0;
  double ci=0,cg=0,cm=0,wi=0,wg=0,wm=0,no_seen_edge=0;
  vol **vol_mat, **new_vol_mat;

  //Recovering information from noson.

  if(noson->length>0){
    for(int i=0;i<noson->length;i++){

      if(noson->position[2*i]>chl->width ||
noson->position[2*i+1]>chl->width || noson-
```

```
>position[2*i+1]<=0
|| noson->position[2*i]<=0)
        mexErrMsgTxt("Error in noson index");

      chl->image[(int)(noson->position[2*i]-1)+
(int)(noson->position[2*i+1]-1)*chl->heigth]=0;

    }
  }
  //mexPrintf("Hi ha %d nofills\n",noson->length);


  //Recovering information from i_volumes

  if(i_volumes->length>0){

    vol_mat=new (vol *) [chl->heigth];

    for(int i=0;i<chl->heigth;i++){
      vol_mat[i]=new vol [chl->width];
    }

    for(int i=0;i<i_volumes->length;i++){
      (vol_mat[(int)(i_volumes->position[i*4])-
1][(int)
(i_volumes->position[i*4+1])-1]).volume=
(i_volumes->position[i*4+2]);
      (vol_mat[(int)(i_volumes->position[i*4])-
1][(int)
(i_volumes->position[i*4+1])-1]).average=
(i_volumes->position[i*4+3]);
      if(i_volumes->position[i*4+3]>257 ||
i_volumes->position[i*4+3]<0)
 mexErrMsgTxt("Average value error");
    }

  }
  else{
    vol_mat=new (vol *) [chl->heigth];

    for(int i=0;i<chl->heigth;i++){
```

```
      vol_mat[i]=new vol [chl->width];
    }

    for(int i=0;i<chl->heigth;i++){
      for(int j=0;j<chl->width;j++){
        (vol_mat[i][j]).volume=1;
        (vol_mat[i][j]).average=scale1-
>image[i+j*scale1->heigth];
        if(vol_mat[i][j].average>257)
mexErrMsgTxt("Average value error");
      }
    }
  }


  //Building new volumes

  new_vol_mat=new (vol *) [fath->heigth];

  for(int i=0;i<chl->heigth;i++){
    new_vol_mat[i]=new vol [fath->width];
  }


  //Counting parents

  for(int i=0;i<fath->heigth;i++){
    for(int j=0;j<fath->width;j++){
      if((int)fath->image[i+j*fath->heigth]==0)
mexErrMsgTxt("Pares xungos");
      if((int)fath->image[i+j*fath->heigth]!=1 &&
(int)fath->image[i+j*fath->heigth]!=0)
        mexErrMsgTxt("Matriu Pares no binaria");
      sum_fathers=sum_fathers+(int)fath-
>image[i+j*fath->heigth];
    }
  }
  //mexPrintf("Hi ha %d Pares\n",sum_fathers);

  if(sum_fathers>(fath->heigth*fath->width)){
    mexPrintf("Hi ha %d Pares\n",sum_fathers);
```

```
        mexErrMsgTxt("Error massa pares");
  }


  //creatin the new structure for parents and filling

  fathers[0]=new int [sum_fathers];
  fathers[1]=new int [sum_fathers];



  for(int i=0;i<fath->heigth;i++){
    for(int j=0;j<fath->width;j++){
      if(fath->image[i+j*fath->heigth]==1){
        if(k>=sum_fathers) mexErrMsgTxt("Error fa-
thers out of range");
        fathers[0][k]=i;
        fathers[1][k]=j;
        k++;
      }
    }
  }

  family[0]=new vector<int> [sum_fathers];
  family[1]=new vector<int> [sum_fathers];

  number_son=new int [sum_fathers];

  for(int i=0;i<sum_fathers;i++){
    family[0][i].push_back(fathers[0][i]+1);
    family[1][i].push_back(fathers[1][i]+1);
    number_son[i]=0;
  }

  sigmap=0.455*exp(double((level+1))*log(2)/OCTAVE_DIV+
(OCTAVE_DIV-1.0)*log(2)/OCTAVE_DIV);
  sigmac=0.455*exp(double(level)*log(2)/OCTAVE_DIV+
(OCTAVE_DIV-1.0)*log(2)/OCTAVE_DIV);

  limwindow=0.5*sigmap;
  searchrad=3*sqrt(sigmap*sigmap-sigmac*sigmac);
```

```
  norma=exp(-(limwindow*limwindow)/(2*(sigmap*sigmap-
sigmac*sigmac)));

  if(old_max_dif>255){
    mexPrintf("old_max_dif = %d\n",old_max_dif);
    mexErrMsgTxt("Error: old_max_dif out of range");
  }


  for(int iter=0;iter<MAX_ITER;iter++){

    max_son=0;
    father=0;
    k=0;
    //mexPrintf("Family size: %d %d %d\n",
2,sum_fathers,family[0][0].size());
    //mexPrintf("Iter %d\n",iter);

    for(int i=0;i<chl->heigth;i++){
      for(int j=0;j<chl->width;j++){

        if(chl->image[i+j*chl->heigth]==1){

          color=scale1->image[i+j*chl->heigth];
          oldsim=0;

          //j_min=(((int)(j-searchrad))>0) ? ((int)(j-
searchrad)) : 0;
          //j_max=(((int)(j+searchrad))<fath->width) ?
((int)(j+searchrad)) : fath->width;

          int i_min=(((int)(i-searchrad))>0) ?
((int)(i-searchrad)) : 0;
          int i_max=(((int)(i+searchrad))<(fath-
>heigth-1)) ?
((int)(i+searchrad)) : (fath->heigth-1);


          for(int n=i_min; n<=i_max; n++){
```

```
            int m_min=(int)(j-sqrt(searchrad*searchrad-
(n-i)*(n-i)));
            int m_max=(int)(j+sqrt(searchrad*searchrad-
(n-i)*(n-i)));

            int j_min=(m_min>0) ? m_min : 0;
            int j_max=(m_max<(fath->width-
1)) ? m_max : (fath->width-1);

            for(int m=j_min; m<=j_max; m++){

              k=n*fath->width+m;


              //Checking if the father has chil-
dren or not

              if((new_vol_mat[fathers[0][k]][fathers[1][k]].average!=0
&& iter>0) || iter==0){

                //Checking if there are edges
                if(NO_EDGE_DETECT){
                  no_seen_edge=1;
                }
                else{
                  if(Is_there_an_edge(i,j,n,m,edge)==0){
                    no_seen_edge=1;
                  }
                  else{
                    no_seen_edge=0;
                  }
                }


                if(no_seen_edge){

                  //computation of crieria

                  distance=(i-fathers[0][k])*(i-
fathers[0][k])+
(j-fathers[1][k])*(j-fathers[1][k]);
```

```
                  if(sqrt(distance)<=limwindow){
                    d=1.0;
                  }
                  else{
                    d=exp(-distance/(2*(sigmap*sigmap-
sigmac*sigmac)))/norma;
                  }

                  ci=1.0-fabs(color-scale2-
>image[fathers[0][k]+
fathers[1][k]*scale2->heigth])/diff;
                  wi=WI;
                  if(ci<0 || ci>1){
                    mexPrintf("ci=%f\n",ci);
                    mexErrMsgTxt("ci out of range");
                  }


                  if(iter>0){
                    cg=new_vol_mat[fathers[0][k]][fathers[1][k]].
volume/volume_max;
                    wg=WG;
                    wm=WM;
                    cm=1.0-fabs(new_vol_mat[fathers[0][k]]
[fathers[1][k]].average-vol_mat[i][j].average)/diff;
                    if(cg<0 || cg>1){
                      mexPrintf("cg=%f\n",cg);
                      mexErrMsgTxt("cg out of range");
                    }
                    if(cm<0 || cm>1){
                      mexPrintf("cm=%f par-
ent_average=%f son_average=%f
diff=%d\n",cm,new_vol_mat[fathers[0][k]][fathers[1][k]].average,
vol_mat[i][j].average,diff);
                      mexErrMsgTxt("cm out of range");
                    }
                  }
                  else{
                    cg=0;
                    cm=0;
```

```
                    wg=0;
                    wm=0;
                  }

                  //mexPrintf("wg: %f   wm: %f   cg: %f  cm: %f\n"
,wg,wm,cg,cm);

                  sim=d*((wi*ci+wg*cg+wm*cm)/(wi+wg+wm));

                  //if(sim!=(d*ci)) mex-
Printf("wg: %f   wm: %f
cg: %f  cm: %f   sim: %f  d*ci: %f \n",wg,wm,cg,cm,sim,d*ci);
                  if(d<0 || d>1) mexErrMs-
gTxt("d out of range");
                  if(ci<0 || ci>1) mexErrMs-
gTxt("ci out of range");
                  //mexPrintf("sim: %f  d*ci: %f \n",sim,d*ci);

                  //Search for the best one
                  if(oldsim<sim){
                    oldsim=sim;
                    father=k;
                  }

                }
              }
            }
          }


          // for(k=0;k<sum_fathers;k++){ //canviar***********
          //      distance=(i-fathers[0][k])*(i-
fathers[0][k])+
(j-fathers[1][k])*(j-fathers[1][k]);

          //      if(sqrt(distance)<=limwindow){
          //        d=1.0;
          //      }
          //      else{
          //        d=exp(-distance/(2*(sigmap*sigmap-
sigmac*sigmac)))/norma;
```

```
          //      }

          //      sim=(d*(1.0-fabs(color-scale2->
image[fathers[0][k]+fathers[1][k]*scale2-
>heigth])/diff));

          //      if(oldsim<sim){
          //        oldsim=sim;
          //        father=k;
          //      }


          //    }//for(k=0;k<sum_fathers;k++)


          if(father>=sum_fathers)
 mexErrMsgTxt("Error: father out of range");

          //building of the data base.
          if(number_son[father]==max_son){
            max_son++;
            for(int t=0;t<sum_fathers;t++){
              family[0][t].push_back(0);
              family[1][t].push_back(0);
            }
          }

          family[0][father][number_son[father]+1]=i+1;
          family[1][father][number_son[father]+1]=j+1;
          number_son[father]++;
        }
      }//for j
    }//for i

    //mexPrintf("Old Part is OK!!!\n");


    //Generation of new values volume and average
    //mexPrintf("Family size: %d %d %d\n\n",2,sum_fathers,
family[0][0].size());
    volume_max=0;
```

```
    for(int i=0;i<sum_fathers;i++){

      //mexPrintf("Peto al pare %d\n",i);

      double volume_tot=0;
      double average_tot=0;

      if(number_son[i]!=0){
        int j=1;
        while(j<=number_son[i]){
          volume_tot+=(vol_mat[(family[0][i][j]-1)]
[(family[1][i][j]-1)]).volume;
          average_tot+=(vol_mat[(family[0][i][j]-1)]
[(family[1][i][j]-1)]).average*(vol_mat[(family[0][i][j]-
1)]
[(family[1][i][j]-1)]).volume;
          //mexPrintf("he fet fins a la %d\n",j);
          j++;

        }

        average_tot=average_tot/volume_tot;
        if(volume_tot>volume_max){
          volume_max=volume_tot;
        }
        //mexPrintf("Pero puc fer els volums\n");

        (new_vol_mat[fathers[0][i]][fathers[1][i]]).volume=volume_tot;
        (new_vol_mat[fathers[0][i]][fathers[1][i]]).average=average_tot
        //mexPrintf("Pero puc escriure els volums\n");
        if(iter==(MAX_ITER-1)){
          newvolumes[0].push_back((double)fathers[0][i]+1.0);
          newvolumes[1].push_back((double)fathers[1][i]+1.0);
          newvolumes[2].push_back(volume_tot);
          newvolumes[3].push_back(average_tot);
        }
      }
      else{
        (new_vol_mat[fathers[0][i]][fathers[1][i]]).volume=0;
        (new_vol_mat[fathers[0][i]][fathers[1][i]]).average=0;
        //mexPrintf("Pero puc es-
```

```
criure els volums nuls\n");
      }

      if(iter<(MAX_ITER-1)){
        number_son[i]=0;
        (family[0][i]).clear();
        (family[1][i]).clear();
        family[0][i].push_back(fathers[0][i]+1);
        family[1][i].push_back(fathers[1][i]+1);
      }

    }

    //mexPrintf("He pogut fer la part nova");

  }
  //mexPrintf("Family size out: %d %d %d\n\n",2,
sum_fathers,family[0][0].size());
  for(int i=0;i<sum_fathers;i++){

    if(number_son[i]==0){
      newnoson[0].push_back(fathers[0][i]+1);
      newnoson[1].push_back(fathers[1][i]+1);
      fathers_wo_son++;
      family[0][i][0]=-1;
      family[1][i][0]=-1;
    }
    else{
      if(family[0][i][1]==0 || family[1][i][1]==0)
        mexErrMsgTxt("Error in updating number_son");
    }
  }

  dimens_fam[0]=max_son+1;
  dimens_fam[1]=sum_fathers-fathers_wo_son;
  dimens_fam[2]=2;


  for(int i=0;i<fath->heigth;i++){
    delete[] new_vol_mat[i];
  }
```

```
  delete[] new_vol_mat;

  for(int i=0;i<chl->heigth;i++){
    delete[] vol_mat[i];
  }
  delete[] vol_mat;


  delete[] fathers[0];
  delete[] fathers[1];
  delete[] number_son;

  return(sum_fathers);

}

double Is_there_an_edge(int child_i,int child_j,int fath_n,
int fath_m,Image *edge){

  int search_m=0,search_n=0,k=0,kside1=0,kside2=0;
  double grav[2],p_grav[2],mg=1,found=0;

  search_n=child_i;
  search_m=child_j;

  if(search_n!=fath_n || search_m!=fath_m){

    //computation of the search vector (desplace-
ment vector)

    grav[0]=(double)(fath_n-search_n);
    grav[1]=(double)(fath_m-search_m);

    mg=sqrt(grav[0]*grav[0]+grav[1]*grav[1]);

    com

    //computation of the normal vector to the de-
splacement,
    //to look for edges in the immediate neighbors.
```

```
    grav[0]=grav[0]/mg;
    grav[1]=grav[1]/mg;

    p_grav[0]=grav[1];
    p_grav[1]=-grav[0];

    search_n+=(int)(rint(grav[0]));
    search_m+=(int)(rint(grav[1]));

    //mexPrintf("search_n=%d search_m=%d\n",search_n,search_m);

    if(search_n>=edge->heigth || search_n<0 || search_m>=edge-
>width
|| search_m<0){

      mexPrintf("search_n=%d search_m=%d",search_n,search_m);
      mexErrMsgTxt("Out of edges matrix range");
    }

    if(fabs(grav[0])==fabs(grav[1])){
      if(fabs(rint(grav[1]))!=1){
        mexErrMsgTxt("Rounding Error in grav");
      }
    }

    while(search_n!=fath_n && search_m!=fath_m){

      k=edge->heigth*search_m+search_n;

      if(((search_m+(int)(rint(p_grav[1])))>=0 && (search_m+
(int)(rint(p_grav[1])))<edge->width) && ((search_n+
(int)(rint(p_grav[0])))>=0 && (search_n+(int)(rint(p_grav[0])))
<edge->heigth)){

        kside1=edge->heigth*(search_m+(int)(rint(p_grav[1])))+
(search_n+(int)(rint(p_grav[0])));

      }
      else{
        kside1=k;
      }
```

```
      if(((search_m-(int)(rint(p_grav[1])))>=0 && (search_m-

(int)(rint(p_grav[1])))<edge->width) && ((search_n-
(int)(rint(p_grav[0])))>=0 && (search_n-
(int)(rint(p_grav[0])))<edge->heigth)){

        kside2=edge->heigth*(search_m-
(int)(rint(p_grav[1])))
+(search_n-(int)(rint(p_grav[0])));

      }
      else{
        kside2=k;
      }

      if(edge->image[k]!=0 || edge-
>image[kside1]!=0 ||
edge->image[kside2]!=0){
        found=1;
        break;
      }

      grav[0]=fath_n-search_n;
      grav[1]=fath_m-search_m;

      mg=sqrt(grav[0]*grav[0]+grav[1]*grav[1]);

      grav[0]=grav[0]/mg;
      grav[1]=grav[1]/mg;

      p_grav[0]=grav[1];
      p_grav[1]=-grav[0];

      search_n+=(int)rint(grav[0]);
      search_m+=(int)rint(grav[1]);

      if(search_n>=edge->heigth || search_n<0 || search_m>=
edge->width || search_m<0){

        mexPrintf("search_n=%d search_m=%d",search_n,search_m);
```

```
        mexErrMsgTxt("Out of edges matrix range");
      }

      if(fabs(grav[0])==fabs(grav[1])){
        if(fabs(rint(grav[1]))!=1){
          mexErrMsgTxt("Rounding Error in grav");
        }
      }

    }
  }
  else{
    found=0;
  }

  return(found);

}
```

segment.m

```
function regions=segment(family,level,dimim)
%
% function that follows back the structure of im-
age stored in
% the cell array family.
%
% level: indicates from which level it must be-
gin te segmentation.
% dimim: indicates the dimensions of the image.
%
% regions: this function returns a cell ar-
ray were each cell is a
%  matrix with the pixel positions of the segment.
%



%reconstruction of the image
for n=level:-1:1
```

```
  family_half=family{n};

  temp1(:,:,1)=full(family_half{1});
  temp1(:,:,2)=full(family_half{2});
  clear family_half;

   dim=size(temp1);

   if(n==level)
      regions{dim(2)}=[];
      disp('   Total regions number:');
      nregions=dim(2)
   end

   for m=1:nregions
      if (n==level)
         tempa=shiftdim(temp1(2:dim(1),m,:),2);
         [nox,noy,a]=find(tempa(1,:));
         [nox,noy,b]=find(tempa(2,:));
         regions{m}=[a; b];
         clear tempa;
      else
         tempb=(regions{m}).';
         if(m==1)
            tempa=(shiftdim(temp1(1,:,:),1));
         end
         [nox,tempinter,noy]=intersect(tempa,tempb,'rows');
         tempaa=temp1(2:dim(1),tempinter,:);
         [nox,noy,a]=find(tempaa(:,:,1));
         [nox,noy,b]=find(tempaa(:,:,2));
         dima=size(a);
         if(dima(2)==1)
            a=a.';
            b=b.';
         end
         regions{m}=[a; b];
      end
    end
    clear temp1;
end
```

```
%Separing unconnected areas.
extra_r=1;
for r=1:nregions
  tempr(:,:)=regions{r};
  [Labeled,NUM]=bwlabel(full(sparse(tempr(1,:).',tempr(2,:)',1
,dimim(1),dimim(2))),8);
  clear tempr;

  if(NUM>1)
    for l=1:NUM
      [a,b]=find(Labeled==l);
      dima=size(a);
      if(dima(2)==1)
        a=a.';
        b=b.';
      end
      if(l==1)
        regions{r}=[a; b];
      else
        regions{nregions+extra_r}=[a; b];
        extra_r=extra_r+1;
      end
    end
  end
end

disp('   Total regions after spliting unconnected:');
disp(length(regions));
```

Appendix C

Probability Distributions of Optical Flow

Following the statement of section (3.2.2), from Eq. (3.10):

$$\mathbf{P}\left(\vec{v} \mid \vec{\nabla} I\right), \tag{C.1}$$

and the model:

$$\vec{v}(\vec{r}) \cdot \vec{\nabla} I(\vec{r}, t) + \frac{\partial I(\vec{r}, t)}{\partial t} = n_2 + \vec{n}_1 \cdot \vec{\nabla} I(\vec{r}, t), \tag{C.2}$$

the Eq. (3.11) can be derived.

Eq. (C.2) describes the conditional probability $\mathbf{P}(\frac{\partial I(\vec{r},t)}{\partial t} \mid \vec{v}, \vec{\nabla} I)$.
From Bayes:

$$\mathbf{P}(\vec{v} \mid , \vec{\nabla} I, \frac{\partial I(\vec{r}, t)}{\partial t}) = \frac{\mathbf{P}(\frac{\partial I(\vec{r},t)}{\partial t} \mid \vec{v}, \vec{\nabla} I) \cdot \mathbf{P}(\vec{v})}{\mathbf{P}(\frac{\partial I(\vec{r},t)}{\partial t})}, \tag{C.3}$$

For the distribution $\mathbf{P}(\vec{v})$ a zero-mean Gaussian with covariance Λ_p is chosen. The resulting distribution of the provability in Eq. (C.1) is Gaussian and the mean and covariance can be extracted as:

$$\begin{aligned} \Lambda_v &= \left[\vec{\nabla} I \cdot \left(\left(\vec{\nabla} I\right)^T \cdot \Lambda_1 \cdot \vec{\nabla} I\right) \cdot \left(\vec{\nabla} I\right)^T + \Lambda_p^{-1}\right]^{-1}, \\ \mu_v &= -\Lambda \cdot \vec{\nabla} I \cdot \left(\left(\vec{\nabla} I\right)^T \cdot \Lambda_1 \cdot \vec{\nabla} I + \Lambda_2\right)^{-1} \cdot \frac{\partial I(\vec{r}, t)}{\partial t}, \end{aligned} \tag{C.4}$$

where Λ_1 is chosen as a diagonal matrix, with diagonal entry σ_1 and $\Lambda_2 = \sigma_2$. Then:

$$\begin{aligned} \Lambda_v &= \left[\frac{\mathbf{M}}{\left(\sigma_1 \|\vec{\nabla} I(\vec{r},t)\|^2 + \sigma_2 \right)} + {\Lambda_p}^{-1} \right]^{-1}, \\ \mu_v &= -\Lambda_v \cdot \frac{\vec{b}}{\left(\sigma_1 \|\vec{\nabla} I(\vec{r},t)\|^2 + \sigma_2 \right)}. \end{aligned} \tag{C.5}$$

Applying the formula over a neighborhood results in the following smoothed version:

$$\begin{aligned} \Lambda_v &= \left[\sum_i \frac{W_i \cdot \mathbf{M_i}}{\left(\sigma_1 \|\vec{\nabla} I(\vec{r}_i,t)\|^2 + \sigma_2 \right)} + {\Lambda_p}^{-1} \right]^{-1}, \\ \mu_v &= -\Lambda_v \cdot \sum_i \frac{W_i \cdot \vec{b}_i}{\left(\sigma_1 \|\vec{\nabla} I(\vec{r}_i,t)\|^2 + \sigma_2 \right)}. \end{aligned} \tag{C.6}$$

Appendix D

Implementation Details in Motion Estimation

D.1 Algorithm Description and Numerical Approximations

Derivative Filter

In the function, all the derivatives have been approximated in the discrete domain by a smoothed version of the classical discrete differential linear operant (diff$[n] = [\ 1, -1\ ]$).

We can approximate continuous derivation $f(x)$:

$$f'(x) = \frac{\partial f(x)}{\partial x}, \tag{D.1}$$

by

$$f'[n] = f[n] * \text{diff}[n]. \tag{D.2}$$

Before performing differentiation it is desired to apply a low-pass filtering (or smoothing) in order to increase the $\frac{S}{N}$ and obtain a cleaner and more reliable Optical Flow. Linear filtering is performed through the convolution of $f[n]$ (the input data function) and the filter $h[n]$:

$$g[n] = h[n] * f[n]. \tag{D.3}$$

If

$$g'[n] = \text{diff}[n] * (h[n] * f[n]), \tag{D.4}$$

and the convolution is associative, $g'[n]$ can be recalculated:

$$g'[n] = (\text{diff}[n] * h[n]) * f[n]. \tag{D.5}$$

It follows that the final result can be calculated directly with the smoothed version of the differential kernel.

In Images, the computation of the gradient is independent in each direction (x and y). To perform the smoothed kernel in both directions, the filtering kernel must be separable. If the kernel is separable, there is no difference between filtering with the complete filter function ($2D$ function) and filtering in each direction with the respective component. Then, if the kernel is separable, we can take each directional component, convolve it with its correspondent differential kernel and we get the smoothed differential kernels.

The kernel $[\ -\frac{1}{12}, \frac{8}{12}, 0, -\frac{8}{12}, \frac{1}{12}\ ]$, which is the implemented one, is the result of the convolution of the differential filter $[\ 1, -1\ ]$ and the Gaussian filter $[\ \frac{1}{12}, \frac{9}{12}, \frac{9}{12}, \frac{1}{12}\ ]$.

"Intersection of Constraints"

In section (3.2.2), we applied a smoothness constraint in order to reduce the singularity problem of the matrix inversion. An "intersection of constraints" is performed through the combination of motion information over a neighborhood. The result of that is a smoother evolution of the Optical Flow over the image.

In *Simoncelli*'s variation (section 3.2.2), the invertibility problem does not exist. The assumption errors according to the reality are modeled statistically, and invertibility is ensured. Anyway, another problem still remains. The aperture problem arises if the computations are performed over a single pixel. The problem is that only the gradient information is considered, and consequently perpendicular motion to the gradient direction is missed. So, a neighborhood scope is needed again, in order to increase the aperture. As in the case of *Lucas & Kanade*, a constraint of uniformity over the neighborhood is introduced. The effect, as we said many times before, is data smoothing. To apply this constraint any weighting function can be taken, and in our case a Gaussian function has been used. The effect is to perform an emphasis on the pixel (center of the evaluation window). The used weighting function is the following:

$$W_i = \begin{bmatrix} 0.0039 & 0.0156 & 0.0234 & 0.0156 & 0.0039 \\ 0.0156 & 0.0625 & 0.0938 & 0.0625 & 0.0156 \\ 0.0234 & 0.0938 & 0.1406 & 0.0938 & 0.0234 \\ 0.0156 & 0.0625 & 0.0938 & 0.0625 & 0.0156 \\ 0.0039 & 0.0156 & 0.0234 & 0.0156 & 0.0039 \end{bmatrix} \tag{D.6}$$

Finally, *Simoncelli* said that changing the σ parameters does not affect much the final result. We have taken the same he used: $\sigma_1 = 0.08,\ \sigma_2 = 1.0,\ \sigma_p = 2.0$.

D.2 Algorithm Code

```
function [V,V_no_filt]=motionbig(frames,a,sigma1,sigma2,sigmap)
%
% [V,V_no_filt]=motionbig(frames,a,sigma1,sigma2,sigmap)
%
% Computes an Optical Flow estimation using the Simoncelli
% variation of the Lucas and Kanade algorithm.
%
% V: is the optical flow of the image frames(:,:,3) cleaned of
% points (points turned to 0) where the eigenvalues of ma-
trix z
% are lower than a.
%
% V_no_filt: the same as V but without cleaning
%
% frames: 3D variable composed by the 5 im-
ages of the sequence
% that are used in the computation of the O.F.
%
% a: min. value for the eigenvalues of z.
%
% sigma1, sigma2, sigmap: std. deviation of the Simon-
celli's
% statistical model.
%

   %Approximation for the derivative function.
   kernel=[-1/12 8/12 0 -8/12 1/12];
   dim=size(frames);

   frames=double(frames);

   %Spatial derivative in the x direction.
   dxframe_3_1=8.*[zeros(dim(1),1) frames(:,1:dim(2)-1,3)];
   dxframe_3_2=-1.*[zeros(dim(1),2) frames(:,1:dim(2)-
2,3)];
   dxframe_3_3=-8.*[frames(:,2:dim(2),3) zeros(dim(1),1)];
   dxframe_3_4=1.*[frames(:,3:dim(2),3) zeros(dim(1),2)];
```

```
   dx=(dxframe_3_1+dxframe_3_2+dxframe_3_3+dxframe_3_4)/12;

   %Spatial derivative in the y direction.
   dyframe_3_1=8.*[zeros(1,dim(2)) ;frames(1:dim(1)-
1,:,3)];
   dyframe_3_2=-1.*[zeros(2,dim(2)) ;frames(1:dim(1)-
2,:,3)];
   dyframe_3_3=-8.*[frames(2:dim(1),:,3); zeros(1,dim(2))];
   dyframe_3_4=1.*[frames(3:dim(1),:,3); zeros(2,dim(2))];

   dy=(dyframe_3_1+dyframe_3_2+dyframe_3_3+dyframe_3_4)/12;

   %Temporal derivative.
   dt=(1.*frames(:,:,1)-8.*frames(:,:,2)+8.*frames(:,:,4) ...
        -1.*frames(:,:,5))/12;

   %weight window to impose the "intersec-
tion of constraints"
   w2=[0.0039 0.0156 0.0234 0.0156 0.0039
     0.0156 0.0625 0.0938 0.0625 0.0156
     0.0234 0.0938 0.1406 0.0938 0.0234
     0.0156 0.0625 0.0938 0.0625 0.0156
     0.0039 0.0156 0.0234 0.0156 0.0039];

   V=zeros(dim(1),dim(2),2);
   V_no_filt=zeros(dim(1),dim(2),2);

   for n=3:dim(1)-3
     for m=3:dim(2)-3

       z=[sum(sum(w2.*dx(n-2:n+2,m-2:m+2).^2./(sigma1*( ...
        dx(n-2:n+2,m-2:m+2).^2+dy(n-2:n+2,m-2:m+2).^2)+ ...
        sigma2)+1/sigmap)) sum(sum(w2.* dx(n-2:n+2,m-
2:m+2) ...
        .*dy(n-2:n+2,m-2:m+2)./(sigma1*(dx(n-2:n+2,m-
2:m+2) ...
        .^2+dy(n-2:n+2,m-2:m+2).^2)+sigma2)));sum(sum(w2.* ...
        dx(n-2:n+2,m-2:m+2).*dy(n-2:n+2,m-
2:m+2)./(sigma1*( ...
        dx(n-2:n+2,m-2:m+2).^2+dy(n-2:n+2,m-
2:m+2).^2)+sigma2))) ...
        sum(sum(w2.*dy(n-2:n+2,m-2:m+2).^2./(sigma1*( ...
```

```
        dx(n-2:n+2,m-2:m+2).^2+dy(n-2:n+2,m-
2:m+2).^2)+sigma2) ...
        +1/sigmap))];

       B=[sum(sum(w2.*dx(n-2:n+2,m-2:m+2).*dt(n-2:n+2,m-
2:m+2) ...
        ./(sigma1*(dx(n-2:n+2,m-2:m+2).^2+dy(n-2:n+2,m-
2:m+2).^2) ...
        +sigma2))); sum(sum(w2.*dy(n-2:n+2,m-2:m+2).* ...
        dt(n-2:n+2,m-2:m+2)./(sigma1*(dx(n-2:n+2,m-
2:m+2).^2+ ...
        dy(n-2:n+2,m-2:m+2).^2)+sigma2)))];

       vals=sort(eig(z));

       %Optical Flow computation
       pre_v=-inv(z)*B;

       %checking of the reliability of the esti-
mates according
       %to the threshold a.
       if(abs(det(z))>0 & vals(1)>=a & vals(2)>=a)
         v=pre_v;
       else
         v=zeros(2,1);
       end

       V(n,m,1)=v(1);
       V(n,m,2)=v(2);

       V_no_filt(n,m,1)=pre_v(1);
       V_no_filt(n,m,2)=pre_v(2);

     end
   end
```

Appendix E

$\chi^2 - distribution$

The following distribution is the $\chi^2 - distribution$ with r degrees of freedom:

$$f_r(x) = \begin{cases} \frac{x^{\left(\frac{r}{2}-1\right)}e^{\left(-\frac{x}{2}\right)}}{\Gamma\left(\frac{r}{2}\right)2^{\frac{r}{2}}} & for\ 0 \leq x < \infty \\ 0 & for\ x < 0. \end{cases} \tag{E.1}$$

The cumulative distribution is the integration of the Eq. above:

$$F_r\left(\chi^2\right) = \int_0^{\chi^2} \frac{x^{\left(\frac{r}{2}-1\right)}e^{\left(-\frac{x}{2}\right)}}{\Gamma\left(\frac{r}{2}\right)2^{\frac{r}{2}}}\,\mathrm{d}x\ = Gammap\left(\frac{r}{2}, \frac{\chi^2}{2}\right), \tag{E.2}$$

where $Gammap(a, z)$ is a regularised gamma function. $Gammap(a, z)$ is also an incomplete Gamma function, which one is defined as:

$$Gammap(a, z) = \frac{1}{\Gamma(a)} \int_0^z x^{(a-1)}e^{(-x)}\,\mathrm{d}x. \tag{E.3}$$

Some important details are:

- Independently distributed χ_j^2 then if

$$\sum_{j=1}^{k} \chi_j^2, \tag{E.4}$$

  the result is another χ_j^2 distribution but with as many degrees of freedom as the addition of all the degrees of freedom of all the χ_j^2.

- If x is a Gaussian distribution, $\left(\frac{x-\mu}{\sigma}\right)^2$ is a χ^2 distribution with $r = 1$ (degrees of freedom) [1].
- If x is a Laplace distribution, $\sqrt{\frac{2}{\sigma^2}}\,|x-\mu|$ is a χ^2 distribution with $r = 2$ (degrees of freedom) [1].

Appendix F

Implementation Details in Statistical Motion Detection

F.1 Algorithm Description

In the following there are described the differed functions used to perform the tests. The described functions can be found in sec. F.2.

estima_motion.m

This function performs the motion estimation of the input sequence. From the output Optical Flow data, some statistical parameters are estimated:

- General mean.
- General standard deviation
- Standard deviation for every frame and velocity component
- Mean for every frame and velocity component.

dist_find.m

This function gives the parameter that makes the best match between the data histogram, and the GGD [31]. Two choices are available. The exponent parameter can be variated or the σ, letting one or the other fixed.

Since it is not necessary a very precise search, any efficient search algorithm has not been used. It is performed an exhaustive search with a determinate resolution steep.

troba.m

In this section two functions are presented, `troba()` and `statist()`.

`troba()` carries out with the general structure of the algorithm. It calls the function `statist()` that performs the test in each motion direction, and then it cleans and classifies the found regions. In the following figure is described the process.

Some techniques have been included in order to improve the quality of the results. After the union of both results of the statistical detection, it is performed a **Holes cleaning** and a **Little spot Cleaning**. These processes are performed using morphological tools, the code can be found at app. G.

- **Holes cleaning** is performed using *reconstruction by erosion* (φ^{rec}) [7], using a plane white mask (white except the image boundaries, which are in black) as a marker, and the original motion region mask as a reconstruction model.
- **Little Spot Cleaning** is performed using *erosion* followed of *reconstruction by dilation* ($\gamma^{rec}\left(X;\varepsilon_B(X)\right)$ where X is the image to filter) [7]. The technique is based on a erosion first by some structural element of the original image, and a posterior reconstruction by dilation using the eroded image as a marker and the original one as the reconstruction model. The used element to erode is a 5x5 binary simetric element. In the implementation, according to the iteration property of the erosion, we have realised the 5x5 element filtering by two 3x3 element consecutive filtering. See App. G for details in morphology.

The following steps **Region Labelling** and **Region Extraction & Centroid Finding**, are based in an scanning of the image to look for the isolated 8-connected regions, their extraction and the computation of the centroid of each region. This centroid is the average of the x and y components of all the pixels forming the region.

`statist()` performs the χ^2 statistical test. The theoretical basis can be found in sec. 3.2.3

F.2 Algorithm Code

estima_motion.m

```
function [out,out_no_pad,sigma,m,sa,sb,ma,mb]=estima_motion(mm)
```

```
%
% [out,out_no_pad,sigma,m,sa,sb,ma,mb]=estima_motion(mm)
%
% Function that performs the motion estima-
tion of a sequence
% and give some statistical values.
%
% out: motion estimation sequence output.
%
% out_no_pad: motion estimation sequence out-
put with no zero
% padding around.
%
% sigma: general av. std computed.
%
% m: general mean computed.
%
% sa: vector composed by all the std values of v_x of the
% sequence.
%
% sb: vector composed by all the std values of v_y of the
% sequence.
%
% ma: vector composed by all the mean values of v_x of the
% sequence.
%
% mb: vector composed by all the mean values of v_y of the
% sequence.
%

dim=size(mm);
sigma=0;
m=0;

for n=3:dim(3)-2

  %Optical Flow Computation
  [v,v_no_filt]=motionbig(mm(:,:,n-2:n+2),1,0.08,1.0,2.0);

  a=v(5:dim(1)-4,5:dim(2)-4,1);
  b=v(5:dim(1)-4,5:dim(2)-4,2);

  %Statistical Parameter computation
```

```
  sa(n-2)=std2(a);
  sb(n-2)=std2(b);
  ma(n-2)=mean2(a);
  mb(n-2)=mean2(b);

  sigma=std2(a).^2/2+std2(b).^2/2+sigma;
  m=mean2(a)/2+mean2(b)/2+m;

  out(:,:,:,n-2)=v;
  out_no_pad(:,:,1,n-2)=a;
  out_no_pad(:,:,2,n-2)=b;

end

sigma=sqrt(sigma/(dim(3)-4));
m=m/(dim(3)-4);
```

dist_find.m

```
function out=dist_find(count,mu,sigma,x,mode,expon)
%
% out=dist_find(count,mu,sigma,x,mode,expon)
%
% Function that gives the parameter that makes best match count
% (empirical distribution) with the GGD func-
tion or the Laplacian.
%
% out: depending on variable 'mode' gives the parame-
ter p of the
% GGD or the sigma.
%
% count: empirical noise distribution.
%
% mu: mean of the model distribution.
%
% sigma: std. of the model distribution.
%
% mode: 1 if it is desired to match 'count' with the GGD with
% a fixed sigma (in that case expon does not matter.
%       0 if it is with 'expon' fixed (in that case sigma does
% not matter).
%
% expon: Parameter that determines the exponent in the GGD.
```

```
%

dims=size(count);

if(dims(1)>dims(2))
  count=count.';
end

dimsx=size(x);

if(dimsx(1)>dimsx(2))
  x=x.';
end

par=1.5;

errlist=[];
parlist=[];

count=count/sum(count);

if(mode==1)

  %search of the best parameter p with resolution 1/1000

  for p=par:-0.001:0.5

    ggd=exp(-(abs(x-mu)/sqrt(sigma.^2*gamma(1/p)/gamma(3/p))) ...
.^p)/(2*gamma(1+1/p)*sqrt(sigma.^2*gamma(1/p)/gamma(3/p)));

    ggd=ggd/sum(ggd);

    err=sum((ggd-count).^2);
    %err=sum(abs(ggd-count));

    errlist=[errlist err];
    parlist=[parlist p];

  end
else

  %search of the best parameter sigma with resolu-
tion 1/10000
```

```
  p=expon;
  ex_sigma=sigma;
   for sigma=ex_sigma*2:-0.0001:ex_sigma/2

    ggd=exp(-(abs(x-mu)/sqrt(sigma.^2*gamma(1/p)/gamma(3/p))) ...
.^p)/(2*gamma(1+1/p)*sqrt(sigma.^2*gamma(1/p)/gamma(3/p)));

    ggd=ggd/sum(ggd);

    err=sum((ggd-count).^2);
    %err=sum(abs(ggd-count));

    errlist=[errlist err];
    parlist=[parlist sigma];

  end

end

[y,I]=min(errlist);

out=parlist(I);

%displaying graphs

if(mode==1)
  p=out;
else
  sigma=out
end

ggd=exp(-(abs(x-mu)/sqrt(sigma.^2*gamma(1/p)/gamma(3/p))) ...
.^p)/(2*gamma(1+1/p)*sqrt(sigma.^2*gamma(1/p)/gamma(3/p)));

ggd=ggd/sum(ggd)*length(ggd);

p=1;

lap=exp(-(abs(x-mu)/sqrt(sigma.^2*gamma(1/p)/gamma(3/p))) ...
.^p)/(2*gamma(1+1/p)*sqrt(sigma.^2*gamma(1/p)/gamma(3/p)));

lap=lap/sum(lap)*length(lap);
```

```
figure(1);
plot(x,count*length(count));
hold on;
plot(x,ggd,'r');
plot(x,lap,'g');
drawnow;
hold off
```

troba.m

```
function [area,centroids,totalspeed]=troba(speedx,speedy,
        square,sigma,thres,av)
%
% [area,centroids,totalspeed]=troba(speedx,speedy,square,
% sigma,thres)
%
% Computes the centroids of the motion areas in the image.
%
% area: Is a cell array containing the pixels that belong
% to certain area.
%
% centroids: Matrix containing the centre of the main
% motion areas in the image.
%
% totalspeed: Binary image containing the detected motion
% areas.
%
% speedx: Matrix containing the x component of the Opti-
cal Flow.
% speedy: Matrix containing the y component of the Opti-
cal Flow.
%
% square: The test window is square window of (2*square+1)
% side length.
%
% sigma: The std. deviation of the estimated noise in the
% motion estimation (it has two components sigma(1) -
vertical
% sigma-, sigma(2) -horizontal sigma-.
%
% Noise distribution is assumed to be Gaussian.
%
```

```
% thres: Test threshold.
%

%Statistical test to find the motion areas
speedx=statist(speedx,square,sigma(1),thres,av);
speedy=statist(speedy,square,sigma(2),thres,av);

totalspeed=speedx | speedy;

dims=size(totalspeed);

b=zeros(dims(1),dims(2));
b(2:dims(1)-1,2:dims(2)-1)=ones(dims(1)-2,dims(2)-2);

%Cleanning of holes
totalspeed=erec_gray(totalspeed,b,1);

totalspeed_old=totalspeed;

%Little spot cleaning
totalspeed=erode_gray(totalspeed);
totalspeed=erode_gray(totalspeed);
totalspeed=drec_gray(totalspeed_old,totalspeed,1);

%Labelling of the separated areas
[L,NUM]=bwlabel(totalspeed,8);

count=1;
entra=0;

%Computation of the area centroids
if(NUM)
  for i=1:NUM
    [a(:,1),a(:,2)]=find(L == i);
    llarg(i)=length(a);
    if(length(a)>=(0.5/100*dims(1)*dims(2)))
      entra=1;
      area{count}=a;
      centroids(count,:)=round(sum(a)/length(a));
      count=count+1;
    end
    clear a;
  end
```

```
  if(entra==0)
    [fil,col]=max(llarg);
    [a(:,1),a(:,2)]=find(L == col);
    area{1}=a;
    centroids(1,:)=round(sum(a)/length(a));
  end
else
  area={};
  centroids=[];
end

function out=statist(speed,win,sigma,threshold,av)
%
% out=statist(speed,win,sigma,threshold,av)
%
% Function that returns a binary mask with the esti-
mated motion
% areas.
%
% out: Binary Mask
%
% speed: v_x component matrix.
%
% win:  The test window is a square win-
dow of (2*win+1) side
% length.
%
% sigma: std. of the Laplacian distribution noise.
%
% threshold: Determines which is the maximum probabil-
ity accepted
% that a pixel is not changed due to noise given Ho. 1-
threshold
% is the probability to be a moving object.
%
% av: mean of the Laplacian distribution noise.

dim=size(speed);

delta=zeros(dim(1),dim(2));
sig_level=delta;
out=delta;
```

```
for n=win+1:dim(1)-win
  for m=win+1:dim(2)-win

    delta(n,m)=sum(sum(abs(speed(n-win:n+win,m-win:m+win)-
av))) ...
        *sqrt(2)/sigma;

    sig_level=1.0-gammainc(delta(n,m),(2*win+1).^2);

    if(sig_level<=threshold)
      out(n,m)=1;
    end
  end
end
```

Appendix G

Morphological Tools

Some Morphological Tools have been used to different purposes. This morphological Tools have been used on $2D$, on gray-scale images, and on binary images. A general gray-scale version has been implemented for both purposes. Much theory can be found in [7] [16].

G.1 Main operators: Dilation, Erosion, Reconstruction by Dilation & Reconstruction by Erosion

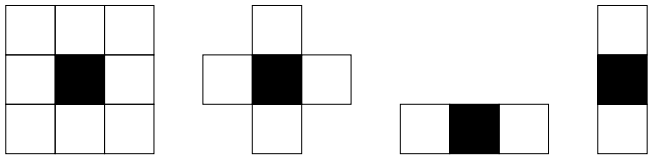
Dilation

Taking Dilation as defined like:

$$\delta(X) = X \bigoplus B = \bigcup_{p \in X \ \& \ b \in B} p + \vec{b} \tag{G.1}$$

where B is the structural element, X is the image, p represents a pixel and $\vec{b}$ is a structural flat element in a certain direction.

The structural elements used are:



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

dilate_gray.m Dilation with the 3x3 element.

```
function dilated=dilate_gray(image)
%
% dilated=dilate_gray(image)
%
% This function returns the dilated version of 'image'.
% It performs the dilation with a 3x3 simetric structurant
% Element.
%

dim=size(image);

im(:,:,1)=image;
im(:,:,2)=[zeros(1,dim(2)); image(1:dim(1)-1,:)];
im(:,:,3)=[zeros(1,dim(2)); [image(1:dim(1)-1,2:dim(2)) ...
zeros(dim(1)-1,1)]];
im(:,:,4)=[zeros(1,dim(2)); [zeros(dim(1)-1,1) ...
image(1:dim(1)-1,1:dim(2)-1)]];
im(:,:,5)=[zeros(dim(1),1) image(:,1:dim(2)-1)];
im(:,:,6)=[image(:,2:dim(2)) zeros(dim(1),1)];
im(:,:,7)=[[image(2:dim(1),2:dim(2)) zeros(dim(1)-
1,1)]; ...
zeros(1,dim(2))];
im(:,:,8)=[[zeros(dim(1)-1,1) image(2:dim(1),1:dim(2)-
1)]; ...
zeros(1,dim(2))];
im(:,:,9)=[image(2:dim(1),:); zeros(1,dim(2))];

dilated=max(im,[],3);
```

dilate_gray4.m Dilation with the 'cross' element.

```
function dilated=dilate_gray4(ima)
%
% dilated=dilate_gray4(ima)
%
% This function returns the dilated version of 'ima'.
% It performs the dilation with a  simet-
ric 'cross' structurant
% Element.
%
```

```
dim=size(ima);

im(:,:,1)=ima;
im(:,:,2)=[zeros(1,dim(2)); ima(1:dim(1)-1,:)];
im(:,:,3)=[zeros(dim(1),1) ima(:,1:dim(2)-1)];
im(:,:,4)=[ima(:,2:dim(2)) zeros(dim(1),1)];
im(:,:,5)=[ima(2:dim(1),:); zeros(1,dim(2))];

dilated=max(im,[],3);
```

dilate_gray2(ima,dir)

```
function dilated=dilate_gray2(ima,dir)
%
% dilated=dilate_gray2(ima,dir)
%
% This function returns the dilated version of 'ima'.
% It performs the dilation with a  simetric lin-
ear structurant
% Element 1x3 or 3x1.
%
% ima: image.
% dir: 1 if vertical direction.
% dir: 2 if horizontal direction.

dim=size(ima);

im(:,:,1)=ima;
if(dir==1)
  im(:,:,2)=[zeros(1,dim(2)); ima(1:dim(1)-1,:)];
  im(:,:,3)=[ima(2:dim(1),:); zeros(1,dim(2))];
else
  im(:,:,2)=[zeros(dim(1),1) ima(:,1:dim(2)-1)];
  im(:,:,3)=[ima(:,2:dim(2)) zeros(dim(1),1)];
end

dilated=max(im,[],3);
```

Erosion

Taking Erosion as defined like:

$$\varepsilon(X) = X \Theta B = \bigcup_{p \in X \ \& \ b \in B} p - \vec{b} \qquad \text{(G.2)}$$

where B is the structural element, X is the image, p represents a pixel and $\vec{b}$ is a structural flat element in a certain direction.

The structural elements used are the same than the used for the dilation, see section above.

erode_gray(image)

```
function eroded=erode_gray(image)
%
% eroded=erode_gray(image)
%
% This function returns the eroded version of 'image'.
% It performs the erosion with a 3x3 simetric structurant
% Element.
%

dim=size(image);

im(:,:,1)=image;
im(:,:,2)=[255.*ones(1,dim(2)); image(1:dim(1)-1,:)];
im(:,:,3)=[255.*ones(1,dim(2)); [image(1:dim(1)-1, ...
2:dim(2)) 255.*ones(dim(1)-1,1)]];
im(:,:,4)=[255.*ones(1,dim(2)); [255.*ones(dim(1)-1, ...
1) image(1:dim(1)-1,1:dim(2)-1)]];
im(:,:,5)=[255.*ones(dim(1),1) image(:,1:dim(2)-1)];
im(:,:,6)=[image(:,2:dim(2)) 255.*ones(dim(1),1)];
im(:,:,7)=[[image(2:dim(1),2:dim(2)) 255.*ones(dim(1)-
1, ...
1)]; 255.*ones(1,dim(2))];
im(:,:,8)=[[255.*ones(dim(1)-1,1) image(2:dim(1),1:dim(2)-
1)]; ...
 255.*ones(1,dim(2))];
im(:,:,9)=[image(2:dim(1),:); 255.*ones(1,dim(2))];

eroded=min(im,[],3);
```

erode_gray4(ima)

```
function eroded=erode_gray4(ima)
%
% eroded=erode_gray4(ima)
%
% This function returns the eroded version of 'image'.
```

```
% It performs the erosion with a  simet-
ric 'cross' structurant
% Element.
%

dim=size(ima);

im(:,:,1)=ima;
im(:,:,2)=[255.*ones(1,dim(2)); ima(1:dim(1)-1,:)];
im(:,:,3)=[255.*ones(dim(1),1) ima(:,1:dim(2)-1)];
im(:,:,4)=[ima(:,2:dim(2)) 255.*ones(dim(1),1)];
im(:,:,5)=[ima(2:dim(1),:); 255.*ones(1,dim(2))];

eroded=min(im,[],3);
```

Reconstruction by Dilation

This operation is based on the marking of a certain image F, with a certain $2D$ marker g, and dilate g as many times as is necessary till it is reconstructed with F as a model. This means that g will be similar to F but some information will be missing, according to how was the initial mark g.

We can notate this as, according to the previous notation:

$$\delta_1^F(g) = \delta(g) \wedge F \rightarrow \delta_\lambda^F(g) = \left(\delta_1^F(g)\right)^\lambda, \tag{G.3}$$

where δ indicates the dilation operation, g is the mark, F the model image. δ_1^F means dilation with reconstruction 1 iteration, and in the second term, λ means the number of times.

drec_gray.m

```
function rec_out=drec_gray(image,rec_out,steps)
%
% function rec_out=drec_gray(image,rec_out,steps)
%
% rec_out: reconstructed image.
% image: model image for the reconstruction.
% rec_out: (the input one) image to reconstruct.
% steps: nothing, not used, put any value.

if(steps<=0)
  error('steps: bad number of steps');
```

```
end

change=1;
dim=size(image);
temp = zeros(dim(1),dim(2));

while(change)
     rec_out_old=rec_out;
     dcue=rec_out;
     dcue=dilate_gray(dcue);
     temp = (image <= dcue);
     rec_out(temp) = image(temp);
     rec_out(~temp) = dcue(~temp);
     dif=double(rec_out)-double(rec_out_old);
     change=max(max(dif));
end
```

Reconstruction by Erosion

In Reconstruction by Erosion, we find the homologue operation to the Reconstruction by Dilation, but with Erosion.

$$\varepsilon_1^F(g) = -\delta_1^{-F}(-g) \rightarrow \varepsilon_\lambda^F(g) = \left(\varepsilon_1^F(g)\right)^\lambda \tag{G.4}$$

erec_gray.m

```
function rec_out=erec_gray(ima,rec_out,steps)
%
% function rec_out=erec_gray(image,rec_out,steps)
%
% rec_out: reconstructed image.
% image: model image for the reconstruction.
% rec_out: (the input one) image to reconstruct.
% steps: nothing, not used, put any value.


if(steps<=0)
  error('steps: bad number of steps');
end

change=1;
dim=size(ima);
```

```
temp = zeros(dim(1),dim(2));
while change
   rec_out_old=rec_out;
   dcue=rec_out;
   dcue=erode_gray(dcue);
   temp = (ima>= dcue);
   rec_out(temp) = ima(temp);
   rec_out(~temp) = dcue(~temp);
   dif=double(rec_out_old)-double(rec_out);
   change=max(max(dif));
end
```

G.2 Tools Based on Morphological Operators

Finding Maxima in an image.

```
function max_image=max_find(im)
%
% This function returns a binary mask with the max-
ima of the image.
%
%
%


im=im/(max(max(im))-min(min(im)))*2^32;

imcue=double(im)-1;
rec_out=drec_gray(im,imcue,1);
max_image=logical(double(im)-double(rec_out));
```

Finding Minima in an image.

```
function min_image=min_find(im)
%
% This fucntion returns a binary mask with the min-
ima of the image
%
%
%
```

```
im=im/(max(max(im))-min(min(im)))*2^32;

imcue=double(im)+1;
rec_out=erec_gray(im,imcue);
min_image=logical(double(rec_out)-double(im));
```

Finding Ridges in an image (Finding edges in the Gradient image).

```
function max_image=edge_find(im,dir)
%
% The function returns a mask with the ridges of
% the image (the masc is zero
% where there is no ridge, and the amplitude of
% the ridge where it is).
%
% im: image
% dir: direction: 0 vertical gradient ridges, 1
% horizontal gradient ridges,
% 2 both direction ridges.


max_image=zeros(size(im));

if((max(max(im))-min(min(im)))>0)
  im=(im/(max(max(im))-min(min(im))))*2^32;
  imcue=double(im)-1;
else
  imcue=im;
end



if(dir==0 | dir==2)
  rec_out1=drec_gray2(im,imcue,1,0);
  max_image=logical(double(im)-double(rec_out1));
end

if(dir==1 | dir==2)
  rec_out2=drec_gray2(im,imcue,1,1);
  max_image=max_image | logical(double(im)-
double(rec_out2));
end
```

Filtering Morphologically an image.

It has been used in the cleaning of the segmented regions masc a *alternated sequential filter* [7] of appertures and closings with reconstruction. To eliminate the small regions like alone pixels that last in the end of the segmentation process.

See app. B the code is included in the **program.m**.

```
if(cleaning)
disp('Cleaning...');
tot_masc=clean_small_reg(tot_masc,1);
itera=2;
  change=0;
  old_tot_masc=tot_masc;
  while(change==0)

    disp('Cleaning...');
    tot_masc=clean_small_reg(tot_masc,itera);
    if(old_tot_masc==tot_masc)
      change=1;
    else
      itera=itera+1;
      old_tot_masc=tot_masc;
    end
  end
```

Filling Holes.

Filling holes is a classic aplication of the morphological filters [7]. In the following there is a little piece of code that performs such a tasc.

Let *totalspeed* be a binary mask that we want to clean, so:

```
dims=size(totalspeed);

b=zeros(dims(1),dims(2));
b(2:dims(1)-1,2:dims(2)-1)=ones(dims(1)-2,dims(2)-2);

%Cleanning of holes
totalspeed=erec_gray(totalspeed,b,1);
```

Bibliography

[1] Kaup A., Aach T. and Mester R.; *Statistical model-based change detection in moving video.*; Signal Processing, 31:165-180, 1993.

[2] Ayer S.; *Sequential and competitive methods for estimation of multiple motions*; PhD thesis, Swiss Federal Institute of Technology, Lausanne, Switzerland, 1995.

[3] Barron J.L., Fleet D.J. and Beauchemin S.S.; *Performance of optical flow techniques*; International Journal of Computer Vision, Vol. 12, No. 1, pp.43-77, February 1994.

[4] Bullier J.; *Architecture du systeme visuel*; In Proc. of the NSI'97, LA PERCEPTION VISUELLE, Aspects Multi-Disciplinaires, Centre Paul Langevin Aussois, May 1997. NSI and ACTH.

[5] Burt P., Hong T.H. and Rosenfeld A.; *Segmentation and estimation of image region properties through cooperative hierarchical computation*, IEEE Trans. Sust., Man, Cybern., vol. SMC-11, pp. 802-809, 1981.

[6] Cai D., Deangelis G. C., Freeman R. D.; *Spatiotemporal Receptive Field Organization in Lateral Geniculate Nucleus of Cats and Kittens*; The Journal of Neurophysiology Vol .78 No. 2 August 1997, pp. 1045-1061.

[7] Casas J. R., Marqués F., Salembier P.; *Processament d'Imatge (PIM)*; Transparències del Curs; Departament de Teoria del Senyal i Comunicacions, Universitat Politècnica de Catalunya, Barcelona.

[8] Castagno R.; *Video Segmentation Based on Multiple Features for Interactive and Automatic Multimedia Applications*; PhD thesis, Swiss Federal Institute of Technology, Lausanne, 1998.

[9] Castagno R., Ebrahimi T., Kunt M.; *Video Segmentation Based on Multiple Features for Interactive Multimedia Applications*; IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, no. 5, September 1998.

[10] Florack L. M. J, Haar Romeny B. M. ter, Koenderink J. J., Viergever M. A.; *Linear scale-space*; Journal of Mathematical Imaging and Vision, 4(4):325-251, 1994.

[11] Gu L., Bone D.; *Skin Colour Region Detection in MPEG Video Sequences*; Image Analysis and Processing, 1999. Proceedings. International Conference on , 1999.

[12] ter Haar Romeny B. M.; *Introduction to Scale-Space Theory: Multiscale Geometric Image Analysis*; Technical Report No. ICU-96-21, Utrecht University.

[13] Henkel R. D.; *Segmentation with Synchronising Neural Oscillators*; ZKW Bericht Nr. 4/94; Zentrum Für Kognitionswissenschaften, Universität Bremen 1994.

[14] Henkel R. D.; *Segmentation in Scale Space*; Proc. of. 6th Int Conf. on Computer Analysis of Images and Pattern, CAIP '95, Prague 1995.

[15] Jackway P.T. and Deriche M.; *Scale-Space properties of multiscale morphological dilation-erosion*; IEEE Trans. Pattern Analysis and Machin Intelligence, 18(1):38-51, 1996.

[16] Jain, Anil K.; *Fundamentals of digital image processing*; Englewood Cliffs, NJ Prentice Hall cop. 1989.

[17] Koenderink J. J.; *The structure of images*; Biological Cybernetics, 50:363-370, 1984.

[18] Lifshitz L. M., Pizer S. M.; *A multi-resolution hierarchical approach to image segmentation based on intensity extrema*; IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 12. No. 6 June 1990.

[19] Lindeberg T.; *Scale-Space for Discrete Signals*; Pattern Analysis and Machine Intelligence, IEEE Transactions on , Volume: 12 Issue: 3, March 1990.

[20] Lindeberg T., Eklundh J. O.; *Scale Detection and Region Extraction from a Scale-Space Primal Sketch*; Computer Vision, 1990. Proceedings, Third International Conference on , 1990.

[21] Lindeberg T.; *Scale-Space Theory in Computer Vision*; Kluwer Academic Publishers, The Netherlands, 1994.

[22] Lucas B. D., Kanade T.; *An iterative image registration technique with an application to stereo vision*; In Proceedings of the 7th International Joint Conference on Artificial Intelligence, pag. 674-679, Vancouver, 1981.

[23] Moscheni F.; *Spatio-Temporal Segmentation and Object Tracking: An Application to Second Generation Video Coding*; PhD thesis, Swiss Federal Institute of Technology, Lausanne, 1997.

[24] Moscheni F., Bhattacharjee S., Kunt M.; *Spatio-temporal Segmentation Based on Region Merging*; IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 9, September 1998.

[25] Moscheni F., Dufaux F., Kunt M.; *Object Tracking Based on Temporal and Spatial Information*; Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on , Volume: 4 , 1996.

[26] Osher S. and Sethian S; *Fronts propagating with curvature dependent speed: algorithms based on the Hamilton-Jacobi formalism*. J. Comutational Physics, 79:12-49, 1988.

[27] Perona P. and Malik J.; *Scale-Space and edge detection using anisotropic diffusion*. IEEE Trans. Patern Analysis and Machine Intelligence, 12(7):629-639, July 1990. 16, 20.

[28] Singh A.; *Optical Flow Estimation - A Unified Perspective*; IEEE Computer Society Press, 1991.

[29] Simoncelli E. P., Adelson E. H., Heeger D. J.; *Probability Distributions of Optical Flow*; Computer Vision and Pattern Recognition, 1991; Proceedings CVPR '91, IEEE Computer Society Conference on, 1991.

[30] Tarroux Ph.; *Perception active*; In Proc. of the NSI'97, LA PERCEPTION VISUELLE, Aspects Multi-Disciplinaires, Centre Paul Langevin Aussois, May 1997. NSI and ACTH.

[31] Varanasi M. k. and Aazhang B.; *Parametric Generalized Gaussian Density Estimation*; J. Acoust. Soc. Am. vol.86, October 1989.

[32] Vincken K.; *Probabilistic Multi-scale Image Segmentation by the Hyperstack*; PhD thesis, Utrecht University, 1995.

[33] Weikert J.; *Scale-Space properties of nonlinear diffusion filtering with a diffusion tensor*. Technical Report 110, Laboratory of Technomathematics, Univ. of Kaiserslautern, Germany, 1994.

[34] Ziliani F.; *Focus of Attention: An Image Segmentation Procedure Based on Statistical Change Detection*; Internal Report at LTS, Swiss Federal Institute of Technology, Lausanne, 1998.

[35] Ziliani F., Cavallaro A.; *Image Analysis for Video Surveillance Based on Spatial Regularization of a Statistical Model-Based Change Detection*; Image Analysis and Processing, 1999. Proceedings. International Conference on , 1999.

[36] Ziliani F., Björn J.; *Unsupervised Segmentation Using Modified Pyramidal Linking Approach* Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on ,1998.

[37] Ziliani F., Moscheni F.; *Motion Tracking Based on Kalman Filtering for Recursive Spatio-Temporal Segmentation and Object Tracking*; Technical Report, Signal Processing Laboratory, Swiss Federal Institute of Technology, Lausanne.

[38] Ziliani F., Moscheni F.; *Kalman Filtering Motion Prediction For Recursive Spatio-Temporal Segmentation and Object Tracking*; Technical Report, Signal Processing Laboratory, Swiss Federal Institute of Technology, Lausanne.